RESEARCH Open Access

Llamol: a dynamic multi-conditional generative transformer for de novo molecular design

Niklas Dobberstein[1*], Astrid Maass[1] and Jan Hamaekers[1]

Abstract
Generative models have demonstrated substantial promise in Natural Language Processing (NLP) and have found application in designing molecules, as seen in General Pretrained Transformer (GPT) models. In our efforts to develop such a tool for exploring the organic chemical space in search of potentially electro-active compounds, we present Llamol, a single novel generative transformer model based on the Llama 2 architecture, which was trained on a 12.5M superset of organic compounds drawn from diverse public sources. To allow for a maximum flexibility in usage and robustness in view of potentially incomplete data, we introduce Stochastic Context Learning (SCL) as a new training procedure. We demonstrate that the resulting model adeptly handles single- and multi-conditional organic molecule generation with up to four conditions, yet more are possible. The model generates valid molecular structures in SMILES notation while flexibly incorporating three numerical and/or one token sequence into the generative process, just as requested. The generated compounds are very satisfactory in all scenarios tested. In detail, we showcase the model's capability to utilize token sequences for conditioning, either individually or in combination with numerical properties, making Llamol a potent tool for de novo molecule design, easily expandable with new properties.

Scientific contribution
We developed a novel generative transformer model, Llamol, based on the Llama 2 architecture that was trained on a diverse set of 12.5 M organic compounds. It introduces Stochastic Context Learning (SCL) as a new training procedure, allowing for flexible and robust generation of valid organic molecules with up to multiple conditions that can be combined in various ways, making it a potent tool for de novo molecular design.

Keywords Molecular generation, Machine learning, Transformers, De novo molecular design

Introduction
In fields like energy storage materials or medicinal chemistry, substances are key to technological advancement and progress: the success of these applications hinges on the specific properties of the materials. However, the processes of discovery and development of new materials often face practical and/or principal obstacles, such as unavailability of compounds or precursors, high production costs, and the need for extensive trials on the practical side, or limited data and/or experience, as well as biased expectations of designers and developers on the other hand. Generative models, a powerful category in machine learning, have the potential to address both of these issues simultaneously, as they can help focus our efforts a priori only on the *most likely* candidates.

*Correspondence:
Niklas Dobberstein
niklas.dobberstein@scai.fraunhofer.de
[1] Virtual Material Design, Fraunhofer Institute for Algorithms and Scientific Computing, Schloss Birlinghoven, 53757 Sankt Augustin, Germany

© The Author(s) 2024. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

Many architectures related to creation of novel data points were developed in recent years, most notably Recurrent Neural Networks (RNN) [1], Generative Adversarial Networks (GAN) [2], Variational Autoencoders (VAE) [3] and Transformers [4]. The transformer architecture, especially, has revolutionized the fields of Natural Language Processing (NLP) [5] and other domains like computer vision [6]. The introduction of the General Pretrained Transformer (GPT) architecture led to significant advancements in generative natural language applications. Generative models have also been applied in the fields of medicine and material science to create new molecules with *predefined* features, a process known as conditional generation [7, 8]. This application can significantly accelerate the discovery of new candidate molecules. Although current generative models may not provide the optimal solution, they can greatly reduce the size of the chemical space that needs to be evaluated. Current estimates for the size of the chemical space containing drug-like molecules range from 10^{23} to 10^{60} [9]. Many approaches have successfully used VAEs [10–12], GANs [13], RNNs [14, 15] or Reinforcement Learning [16]. However, more recently, transformer models, specifically the GPT models [8, 17], have emerged as the new state-of-the-art in this domain, especially, in the field of conditional molecular generation [18–20]. A good summary of available models can be found in the survey from Du et. al. [21].

Bagal et al. [8] presented the MolGPT architecture from which a family of models, each one tailored to a specific task, could be derived. Inspired by their work, we set out to develop a *solitary* model that can handle many tasks simultaneously to support the search for low-cost, high-energy-density alternatives for energy storage materials in flow batteries. The model itself should not require complex training data; thus, it operates on SMILES [22] – a minimalist molecular representation that allows us to draw a mass of data from numerous sources – and easy to provide and directly to verify target properties that serve as conditions (primarily to facilitate the development process of the model).[1]

In this paper, we present a new, dynamic training approach termed “Stochastic Context Learning” (SCL) to train a single model for conditional generation, capable of generating molecules as SMILES while respecting a variable number of conditions. Our training dataset consists of approx. 12.5 million organic molecules, which is a superset of several public datasets (see Sect. 3.1). On this, we train a GPT-style transformer model, specifically a model based on Llama 2 [23], to generate new compounds based on one or more conditions/target properties. To achieve this, we assign a learnable embedding to each property value. This ensures that the model perceives not only the numerical value, but also the associated label.

To be able to assess the model’s performance directly, we chose three easily determined numerical properties: SAScore [24] (reflecting production cost), logP, and molecular weight (contributing to energy density), along with another optional condition: a user-defined core structure that has to be integrated into the final molecule. The latter is given as a SMILES string, which is a continuous sequence of tokens, hereafter referred to as a ’token sequence’.[2]

In the following sections, we detail the architecture, training data and process along with the results obtained for unconditional, single, and multi-conditional molecule generation.

Architecture

The architecture we utilized, as depicted in Fig. 1, is a modified version of the Llama2 architecture [23] as obtained from GitHub (https://github.com/karpathy/llama2.c). The hyperparameters can be found in Table 1, which we determined from previous experiments.

Our model consists of approximately 15 million parameters and is composed of eight decoder blocks. Each decoder block includes a masked multi-head self-attention layer, followed by a Feed Forward Network (FFN) that employs the SwiGLU [25] activation function. While the original Llama 2 architecture utilized Grouped-Query Attention (GQA) [26], we opted for the full multi-head attention mechanism given the comparatively smaller size of our model.

The masked multi-head self-attention layer [4], defined by Eq. 1, takes an embedded input sequence $X \in \mathbb{R}^{L\times d_{emb}}$ of length L, where each element represents an embedding vector with dimension d_{emb}. Through the attention mechanism, each head learns to attend to a different part of the sequence, resulting in an attention matrix $\text{head}_i \in \mathbb{R}^{L\times d_v}$. We utilize dot-product self-attention, which produces three matrices: Q_i and K_i with dimensions $L \times d_k$, and V_i with dimensions $L \times d_v$. These matrices are generated by applying linear transformations using weight matrices W_Q^i, W_K^i, and W_V^i, each with dimensions of $d_{emb} \times d_k$ and $d_{emb} \times d_v$, respectively, to the input sequence X for each attention head i.

[1] A condition, here, is a desired molecular property that we want to provide to the model. Based on this condition, the model should generate new molecules that satisfy the requested value.

[2] A token sequence can represent either a complete molecule or a molecular fragment, which may not necessarily be valid independently. However, a token sequence should become part of a valid molecule when incorporated into the generative process.

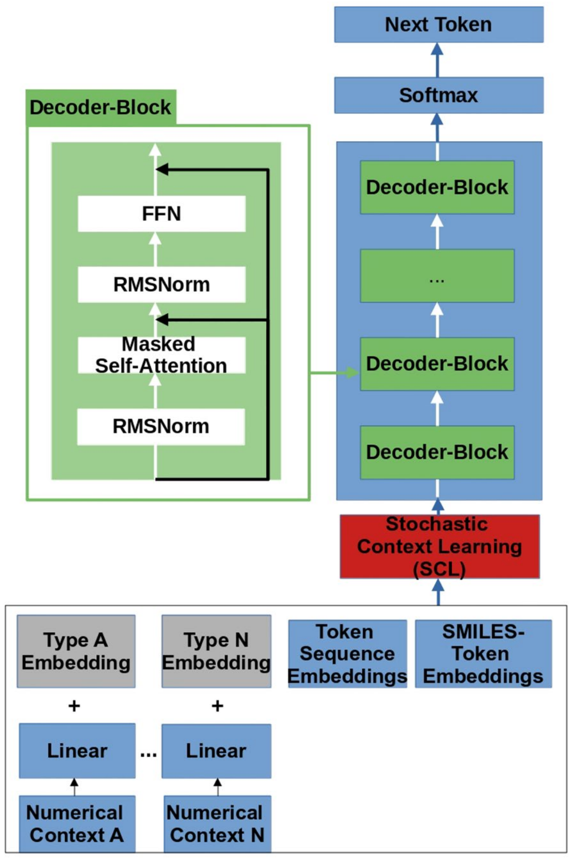


Fig. 1 The Llamol architecture visualized

Table 1 Hyperparameters used for the Llamol model

Hyperparameters	
Parameter/model	**Llamol**
Number of attention-heads n_{heads}	8
Number of decoder-blocks	8
Dropout probability	10%
Activation function	SwiGLU
Positional embeddings	RoPe
Embedding dimension d_{emb}	384
FFN hidden dim d_{ffn}	1024
Vocabulary size d_{voc}	591
Max SMILES length	256

In our specific case, we set d_k and d_v to be equal to d_{emb}/n_{heads}, resulting in $d_k = d_v = 384/8 = 48$. To keep the autoregressive property for our model, we mask out the upper right triangle by using the mask matrix $M \in \mathbb{R}^{L \times L}$ shown in Eq. 4. Then, these attention matrices are concatenated with each other along the d_v-dimension. Afterward, the resulting concatenated matrix is further transformed using another learnable weight matrix $W_O \in \mathbb{R}^{h \cdot d_v \times d_{emb}}$.

$$\text{MMHA}(X) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h) \cdot W_O \tag{1}$$

$$\text{head}_i = \text{MaskedAttention}(X \cdot W_Q^i, X \cdot W_K^i, X \cdot W_V^i) \tag{2}$$

$$\text{Masked Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right) \cdot V \tag{3}$$

$$M = \overbrace{\begin{pmatrix} 0 & -\infty & -\infty & \ldots & -\infty \\ 0 & 0 & -\infty & \ldots & -\infty \\ \ldots & \ldots & \ldots & \ldots & -\infty \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}}^{L} \tag{4}$$

$$\text{FFN}(X) = \text{SwiGLU}(X, W_1, W_3) \cdot W_2 \tag{5}$$

The Llama 2 architecture employs several changes compared to the standard decoder architecture [4]. Firstly, we use rotary positional embeddings (RoPe) [27] to encode absolute and relative positional information directly into the attention matrix. Secondly, instead of applying layer normalization [28] after the self-attention and feed-forward layers, we employ RMSNorm [29] as a more efficient pre-normalization step. A feed-forward layer is described by Eq. 5, where $W_1, W_3 \in \mathbb{R}^{d_{emb} \times d_{ffn}}$ and $W_2 \in \mathbb{R}^{d_{ffn} \times d_{emb}}$ are learned weight matrices and $\odot$ represents the elementwise product of two vectors. After each feed-forward layer, we employ a dropout-layer [30] with the probability given in Table 1. The concatenate function, stacks the matrices row-wise $\text{Concat}(A, B) : \mathbb{R}^{a \times e} \times \mathbb{R}^{b \times e} \rightarrow \mathbb{R}^{(a+b \times e)}$.

Furthermore, we made significant alterations to the context ingestion process. The input to our model is a sequence X of shape $L \times d_{emb}$, which can be divided into two parts: $X = \text{Concat}(C, S)$. The first part, $C \in \mathbb{R}^{c \times d_{emb}}$, also later referred to as the "context", represents the given conditions and can be expressed as $C = Concat((t_1, t_2, \ldots, t_n)^T, t_{ts}) \in \mathbb{R}^{c \times d_{emb}}$. The embedded vectors $t_i \in \mathbb{R}^{d_{emb}}\ \forall i \in \{1, \ldots, n\}$ represent the n numerical conditions, which are provided to the model, in our case $n = 3$. On the other hand, $t_{ts} \in \mathbb{R}^{k \times d_{emb}}$ is a matrix of k embedded tokens, that is concatenated with the numerical conditions. The second part, $S \in \mathbb{R}^{s \times d_{emb}}$, just describes the molecule, as a SMILES, itself. c is just the length of the complete context and s is the length of the given SMILES, both are not fixed in length.

Typically, a token sequence includes multiple tokens, which translates to multiple embeddings in the context, while a numerical condition is represented by one embedding. In our case, the order was the following: First are the embedded numerical conditions, then the token sequence embeddings, and lastly the SMILES itself. During the training process, we learn SMILES embeddings by learning an embedding vector for each token.

In order to facilitate controlled property generation of molecules, we prepend the sequence with conditions, such as numerical values or a token sequence. In principle, the number of conditions has no limit as it is part of the input sequence of the transformer model, although empirical evaluation is needed to determine practical limits and scalability. Initial experiments with ten conditions showed promising results, suggesting the model's potential to handle a larger number of conditions. For the purposes of this paper, we limit the contexts to three numerical values and one token sequence. Each numerical value is assigned a type identifier, and a separate linear layer is used to transform them into the embedding dimension. The transformed values are then combined with the learned type encoding specific to each numerical property. In our implementation, we assigned a fixed type number to each property and mapped it to a learnable vector, which serves as the type encoding.

To provide positional information, we applied RoPe to every part of the context and sequence. Although adding positional information to numerical values is not necessary, we chose to include it for the sake of simplicity in implementation, without negatively impacting the model's performance.

Due to the type identifiers, this approach enables the model to differentiate between various conditions in a straightforward, yet effective manner. Consequently, we are free to mix or even omit conditions within the sequence. This property plays a crucial role in our training procedure, as in combination with the SCL method it allows the model to adapt dynamically and process all possible combinations of context.

The degree of creativity of the model's output can be controlled by the so-called temperature parameter, which is defined as a positive real number $t \in \mathbb{R}_+$, by dividing the output log probabilities by the said value. A temperature of $t = 1$ does not alter the model's output, whereas a lower temperature sharpens the output distribution, thus making it more deterministic. Conversely, a temperature greater than one leads to a higher level of variability.

Training

Dataset

The model was trained on a dataset of molecules, which was compiled from several public sources to create a large and diverse population. The resulting dataset, we call *OrganiX13*, includes SMILES strings of mostly organic and/or drug-like molecules taken from the sources listed in Table 2.

Table 2 Datasets used in the combined dataset

Dataset	Number of SMILES
ZINC 15 [31]	5M
QM9 [32, 33]	134k
ZINC 250k [34]	250k
RedDB [35]	31k
OPV [36]	91k
PubchemQC 2017/2020 [37, 38]	5.3M
CEP [39] subset [40]	20k
ChEMBL [41–44]	2.3M
Combined (OrganiX13)	13.1M (Total) / 12.5M (After removing duplicates)

All SMILES were canonicalized via RDKit, while keeping the stereochemistry intact and duplicates were, via string-based comparison, removed. Entries that could not be parsed properly by RDKit were also removed. Likewise, all molecules exceeding a limit of 256 tokens or ionic structures (salts) were rejected. After this preprocessing, the final dataset contains approximately 12.5 million SMILES.

Subsequently, we used RDKit to provide the numerical values for some quick-to-compute surrogate properties to investigate the training behavior and to enable the direct verification of the generated results. The properties chosen were the logP, SAScore, and molecular weight, as those properties also have an impact on the achievable energy density or cost of an electroactive material in the chosen aqueous flow battery application. In detail,

1. LogP is defined as the logarithm of the partition coefficient, which denotes the hydrophobic or lipophilic nature of a molecule. A positive logP value suggests that the molecule prefers non-polar solvents, whereas a negative value indicates that the molecule is soluble in water, a desirable property for aqueous flow battery systems, which correlates to energy density.
2. Molecular weight can be used as a proxy for its size. Again, to attain high energy densities, we would like to have control over the maximum size of the compounds generated. To ensure numerical similarity with the intervals of other properties, the molecular weights were divided by 100.
3. SAScore: The Synthetic Accessibility Score (SAScore) [24] estimates the ease or difficulty of creating a compound. Based on a frequency analysis of chemical moieties in the PubChem database, it assigns a score ranging from zero (easy) to ten (difficult), which is

supposed to reflect to some extent the cost of production.

The resulting dataset encompasses many SMILES strings that cover a broad range of about 12 units in logP, a range in SAScore from around 1 to 6, as well as a similar range in scaled molecular weights. This served as a basis for the subsequent training.

Procedure

Initially, we convert the SMILES representation into a sequence of tokens using a tokenizer. We used the BERT-tokenizer [45] in DeepChem [46], which employs a fixed vocabulary size of 591 tokens. It splits the SMILES at the character level, except for values enclosed in square brackets, which are treated as a single token.

These tokens are then passed through a separate lookup table, which maps them to a d_{emb}-dimensional embedding space. Prior to feeding the token embeddings into the decoder model, a context is prepended.

The numerical properties are processed as described in section 2.

If we use a token sequence as context, we perform these calculations dynamically in each batch during the training, allowing them to have varying token sequence sizes and content. During a training step, a token sequence represents a contiguous subsequence of the current tokenized SMILES. We start by randomly selecting a starting index from zero up to the current SMILES length, followed by determining a random ending index greater than the starting index but smaller than the current SMILES length.

In our case, we limited the context token to a maximum sequence length of 50 to avoid memory issues, which sufficed for our purposes. This sequence is then embedded using the same embedding layer as the input sequence and combined with an embedding specific to the token sequence, sharing the shape of the input embedding table. Additionally, a learned label embedding is added to these combined token sequence embeddings to indicate their relatedness.

Stochastic context learning (SCL)

Given an input sequence $X \in \mathbb{R}^{L \times d_{emb}}$ of length L, where each element is represented by a d_{emb}-dimensional vector, we divide it into two parts: $X = \text{Concat}(C, S)$. Our algorithm focuses on modifying the context part C. We represent this part as a combination of two parts. The first is $C_{num} = (t_1, t_2, \ldots, t_n)^T \in \mathbb{R}^{n \times d_{emb}}$, where n represents the maximum number of numerical conditions used in the training process (in this case, $n = 3$). The second is the token sequence $C_{ts} \in \mathbb{R}^{k \times d_{emb}}$, where k is the length of the token sequence, such that $C = Concat(C_{num}, C_{ts})$. The length k is not specific and can change for each input sequence X.

To begin, we set a deletion probability p_{del} to 15% during training. For each row in the C_{num} matrix, we check if it should be deleted with a probability of p_{del}. If it meets the criteria, we remove the row from the C_{num} matrix and consequently from the input sequence X, which then would be of shape $(L - i) \times d_{emb}$, where $0 \leq i \leq n$ is the number of deleted numerical conditions. Similarly, the same probability is used to control if the token sequence should remain in the context for the current sequence. In this case, the p_{del} probability says if the entire token sequence should be removed, not just one row. Occasionally, there may be a situation where all conditions in C_{num} and C_{ts} are eliminated. In such instances, the sequence becomes unconditioned.

For batched input sequences X_{batch} with shape $\mathbb{R}^{B \times L_{max} \times d_{emb}}$, the process works similarly. We iterate over each of the n numerical conditions and sample if it should be deleted with a probability of p_{del}. If a condition is selected for deletion, we remove the corresponding row from all entries in the batch of size B. A description for the batched algorithm is given in the Algorithm 1. We assume that every molecule in the batch has all n numerical properties. If a molecule only has a portion of the properties, we would simply pad the missing values. In our case, there was no need for padding, as all molecules had all the numerical properties. The batch is created out of B number of sequences X, each of those could have a different length L due to the variance in length in the token sequence condition and also the SMILES itself. To batch those together, we take the maximum sequence length L_{max} for all sequences that should be packed into the batch and pad the shorter SMILES by appending a pad-token to the length of L_{max}.

Thus, throughout the training process, the model has to handle different combinations of the provided conditions, which allows the model to learn unconditionally, single conditions, and also multiple conditions in one go. Thanks to the type of embeddings we add to every context element, we can change the number of properties that are provided to the model and still have the model distinguish which properties are provided.

Algorithm 1 Batched SCL algorithm

```
Require: Input sequence $X \in \mathbb{R}^{B\times L\times d_{emb}}$, $n$ maximum number of numerical
    conditions, B batch size, L sequence size and $d_{emb}$ is the embedding size.
 1: function SCL($X$)
 2:     $p_{del} \leftarrow 0.15$                                   ▷ Deletion probability
 3:     for $i \leftarrow 1$ to $n$ do
 4:         $r_1 \leftarrow$ RANDOM$(0,1)$                          ▷ Random number between 0 and 1
 5:         if $r_1 \le p_{del}$ then
 6:             Remove row $j$ in the L-axis from all samples in $X$   ▷ Delete numerical
    condition $j$
 7:         end if
 8:     end for
 9:     $r_2 \leftarrow$ RANDOM$(0,1)$                              ▷ Random number between 0 and 1
10:     if $r_2 \le p_{del}$ then
11:         Remove all entries from the token sequence in the L-axis from all samples
    in $X$                                                      ▷ Delete token sequence condition
12:     end if
13:     return $X$                                              ▷ Modified input sequence
14: end function
```

Loss

The model is trained to predict the next token by calculating the cross-entropy loss between the actual next token and the predicted probability for that token. Note that this loss is only calculated for the SMILES part of the given sequence, the prepended context is not considered in the loss. Since we only train with the autoregressive loss, the context does not have to be evaluated while training, making our approach very flexible to various conditions. This loss is then backpropagated through the model using the AdamW optimizer [47]. The cross-entropy loss is defined as follows (Eq. 6):

$$\mathrm{CrossEntropyLoss}(y,\hat{y}) = -\frac{1}{N}\sum_{n=1}^{N}\log\left(\frac{\exp(\hat{y}_{n,y_n})}{\sum_{i=1}^{d_{\mathrm{voc}}}\exp(\hat{y}_{n,i})}\right) \tag{6}$$

In this expression, N is the batch size, where $y \in \{0,1,2,\ldots,d_{\mathrm{voc}}\}^N$ and $\hat{y} \in \mathbb{R}^{N\times d_{voc}}$ correspond to the target tokens and the predicted log probabilities, respectively. The mean over the negative logarithms for the normalized predicted probabilities of the next token is calculated. Here, $\hat{y}_{n,y_n}$ specifically refers to the predicted log probability assigned to the correct target token y_n for the n-th sample in the batch.

The model was trained on a single Nvidia A100 GPU for two days and used about 35 GB VRAM while training. A constant learning rate of $\alpha = 10^{-4}$, with $\beta_1 = 0.9$ and $\beta_2 = 0.95$ was used for the AdamW optimizer. The dataset was randomly partitioned into two parts, a training set and a testing set. The training dataset consisted of 90%, while the testing dataset comprised 10% of the data. The model was trained using a batch size of 256 with gradient accumulation steps of 4 batches. Each sequence for the model starts with a “start of SMILES”-token ([CLS]) and ends with an “end of SMILES”-token ([SEP]). Shorter SMILES strings were padded with a “pad”-token ([PAD]) to match the length of the longest SMILES in that batch. The same padding process was applied to the token sequence in the context.

New SMILES are then sequentially generated by first starting with a “[CLS]”-token and then predicting the next tokens iteratively. The generation ends, when the model predicts the “[SEP]”-token or a specified token limit is reached.

Results and discussion

After the training, we used the model in different scenarios to generate new SMILES, e.g., without any constraints or with one or more constraints (including numerical and/or structural targets), while keeping the temperature parameter constant at *temperature* = 0.8. This value ensures a close but not too strict coupling to the underlying probability distributions, which proved helpful in our experiments.

The metric used to measure the performance of the models for a batch of generated compounds is the mean

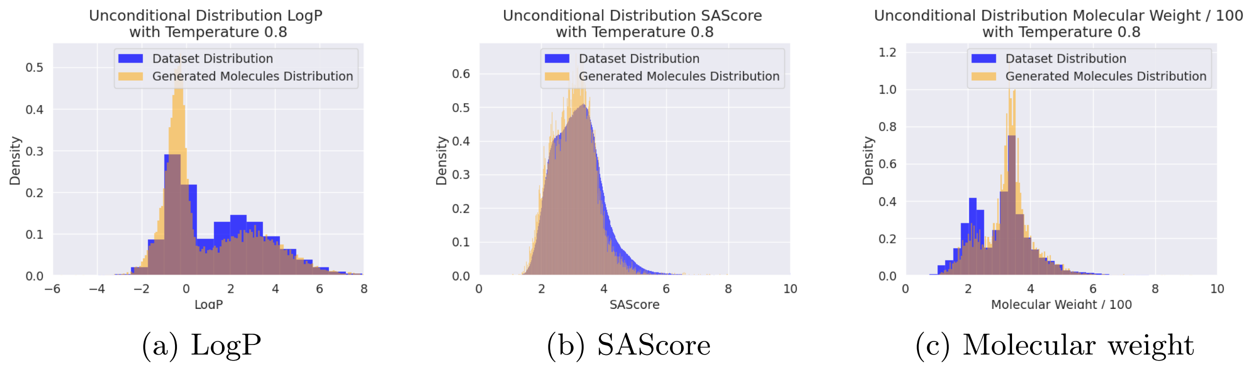


Fig. 2 Distribution of properties as obtained from the training dataset in comparison with the distributions from 20k unconditionally generated molecules

absolute deviation between requested and obtained numerical values, in addition to the percentage of novelty, uniqueness, and validity of the molecular structures generated. These are evaluated on the complete training dataset.

In more detail, these metrics are defined as follows:

1. Novelty: is defined as the percentage of newly generated molecules not present in the reference dataset. We use this, to ensure that the model is not memorizing the training data, but instead is inventing new compounds. We measure this by comparing the generated SMILES with the SMILES in the dataset. Please note: this is not equivalent to testing the molecular graphs for isomerism, i.e., alternative synonyms are not detected as redundant molecular structures by this procedure, but rather just a string comparison.
2. Uniqueness: The uniqueness is the ability of the model to generate unique molecules. We measure the percentage of unique molecules generated in a batch of 1k and 10k molecules under specific conditions.
   Again, identical molecules with synonymous SMILES remain undetected.
3. Validity: The ratio of validity is determined by the number of properly parsed SMILES (by RDKit [48]) versus the total number of generated SMILES in a batch.
4. Mean average deviation: Is defined as the following:

$$MAD = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \qquad (7)$$

   For each of the n generated SMILES strings, the target value of the respective property is denoted as y_i, while x_i represents the 'true', i.e. actually calculated property value. The model should minimize this quantity without being explicitly trained on it, which would indicate that the model incorporates the provided context into the generative process. This metric is also used to enable comparisons to other models [8, 49].

We compare our model to others of similar architecture and choice of conditions. The findings can be seen in Table 3.

Unconditional generation

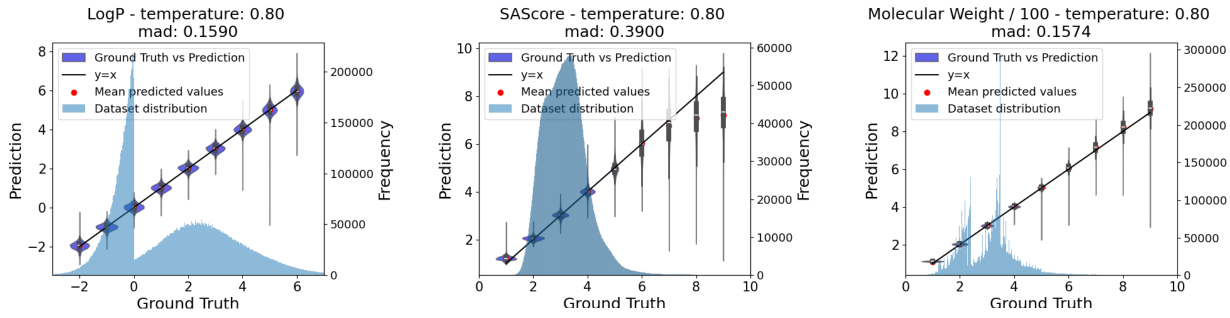
Without applying any conditions, we generated 20k SMILES and calculated the corresponding properties logP, SAScore, and molecular weight using RDKit. The resulting frequencies of distribution are very similar to the distributions obtained from a representative sample of training molecules, see Fig. 2a–c.

This indicates that the model has indeed learned the inherent distribution of the training dataset, without specifically training the model unconditionally.

The generated SMILES also achieve very comparable performance in terms of uniqueness, and validity to the other models as shown in Table 3. Our degree of novelty is inline or slightly better than that of models trained on the MOSES [50] dataset (1.9 Mio. molecules), but falls behind on the GuacaMol (1.59 Mio. molecules) [51]. We argue that this is mostly due to the significant difference in size of our dataset compared to the datasets mentioned above. Since our dataset covers more of the chemical space, this makes it more likely for the model to generate molecules present in the dataset.

Single Condition

In this experiment, we also assessed the model's ability to handle single-condition generation over wide ranges of target values.



(a) logP (b) SAScore (c) molecular weight

Fig. 3 Requested (x-axis) versus actual values (y-axis) for the diverse target properties: a) LogP, b) SAScore, c) Molecular weight. For each target value, a batch of 10k SMILES was generated; MAD is averaged over the entire range

Table 3 Table for comparing metrics for the three metrics at a temperature of 0.8 for 10k generated molecules. All metrics were evaluated with these 10k molecules, except uniqueness at 1k

Model	Condition type	Interval	Novelty [%] ↑	Uniqueness @ 1k [%] ↑	Uniqueness [%]↑	Validity [%] ↑	MAD ↓
Llamol	Unconditional		89.7	100.0	99.9	99.5	
	LogP	[-2, 7; 1]	86.5	100.0	99.9	99.3	0.159
	LogP	{2, 4, 6}	85.5	100.0	99.7	99.42	0.191
	SAScore	[1, 10; 1]	85.4	100.0	98.8	82.1	0.390
	SAScore	{2, 3, 4}	86.6	100.0	99.9	99.7	0.103
	Molecular weight	[1, 10; 1]	88.8	100.0	99.3	97.5	0.157
	Molecular weight	{2, 3, 4}	84.3	100.0	99.6	99.45	0.041
MolGPT	Unconditional (MOSES)		79.7		100.0	99.4	
	Unconditional (GuacaMol [51])		100.0		99.8	98.1	
	LogP[1]	{2, 4, 6}	100.0		99.8	97.1	0.23
	SAScore[1]	{2, 3, 4}	100.0		99.5	97.7	0.13
MolGPT (relative attention) [53]	Unconditional (MOSES)		87.9		100.0	99.2	
	Unconditional (GuacaMol		100.0		100.0	97.8	
	LogP[1]	{2, 4, 6}	100.0		100.0	96.9	N/A
	SAScore[1]	{2, 3, 4}	100.0		99.7	98.6	N/A
Transformer-Decoder Generator [54]	Unconditional (MOSES 30k[2])		97.38		99.92	91.15	
GraphGPT [49]	Unconditional (MOSES)		78.7		99.9	99.5	
	Unconditional (GuacaMol)		100.0		99.9	97.5	
	LogP[1]	{2, 4, 6}	100.0		99.8	96.9	0.22
	SAScore[1]	{2, 3, 4}	100.0		99.6	97.7	0.14

[1] Trained on the GuacaMol dataset

[2] The experiments of the Transformer-Decoder Generator (FSM-DDTR) were conducted on the MOSES dataset with 30k generated molecules instead of 10k

For each target value of the intervals listed in Table 3, rows 2 – 7, the procedure involved generating a sample of 10,000 molecules. As before, we determined the true property values of the generated molecules using RDKit and compared those to the targeted values. For each property, we ran two scenarios: The first one covered a broad interval of values that encompassed both in-distribution, as well as out-of-distribution values. The second run focused solely on the performance of a select few in-distribution target values. The interval notation $[a, b; c]$ signifies that the values were uniformly sampled from a discrete distribution of values, including a and b, with a step size of c. More precisely, [-2, 7; 1] denotes the set of values $\{-2, -1, 0, \ldots, 6, 7\}$. The notation $\{a_1, a_2, \ldots\}$

Table 4 Table for comparing multiple property conditions for 10k generated molecules to other models

Model	Novelty [%] ↑	Uniqueness [%]↑	Validity [%] ↑	LogP {2, 4, 6} MAD ↓	SAScore {2, 3, 4} MAD ↓	Molecular Weight {2, 3, 4} MAD ↓
Llamol	88.9	99.5	98.95	0.20	0.13	
	86.4	89.5	99.1	0.20		0.04
	86.3	99.3	99.5		0.10	0.04
	89.8	94.9	98.5	0.24	0.18	0.05
MolGPT	100.0	99.2	97.2	0.25	0.14	
GraphGPT	100.0	99.1	97.1	0.252	0.151	

indicates that the values were uniformly sampled from that specific set of values.

Despite the low probability of the model being trained solely on one property, it performs well in this task, as demonstrated in Figs. 3a–c. The model achieves low MAD values across the entire span of the respective target properties (rows 2, 4, and 6), although the MAD values obtained for the in-distribution series are generally and significantly lower (rows 3, 5, and 7). In fact, predicting logP values to an accuracy of 0.5 logP units (root mean square deviation) is commonly considered a satisfactory result [52].

Still, the out-of-sample performance is acceptable. Although the scatter increases, the general trend is well retained, with the only exception of the very high (>7) SAScores. In this case (row 4 in Table 3), we observe also a concomitant drop in the percentage of validity of the generated molecules (82.5 vs 99 %). Upon manual inspection, we find that not only compounds with highly bridged ring systems and/or accumulations of stereogenic centers were generated that are supposedly very demanding to synthesize but also rare and unstable atomic environments such as neighboring diradicals and/or carbenes that presumably prevent the proper parsing of structures.

In comparison to the other models in Table 3 our model archives a slightly lower MAD in the single condition case, without being specifically trained on that task, while simultaneously scoring the uniqueness and validity on par with the other models. Yet again, our level of novelty is significantly lower than that from MolGPT. As before, we suspect that this is due to the larger size of our training dataset and the restrictions on the generative process imposed by the constraints given by the context.

Multiple conditions

For each pairwise combination of target properties, we generated 10k SMILES, see Figs. 4, 5 and 6. The graph labels are in the same order as given in the captions of the respective figures.

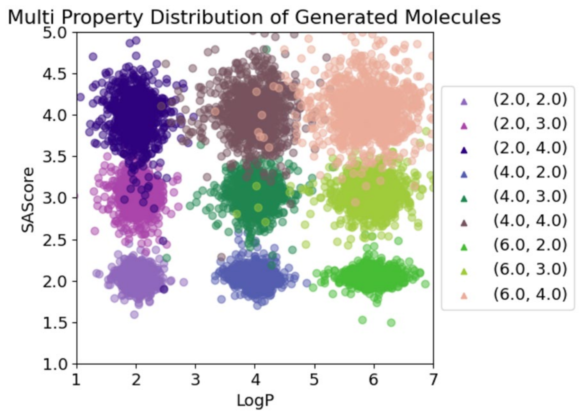


Fig. 4 LogP + SAScore

In general, the generated molecules' properties center closely around the desired values. Although all chosen values were well within the highly populated areas of the underlying distributions, some combinations turned out to be hard to satisfy, resulting in a more pronounced scatter.

Figure 4 shows the distribution of calculated logP values and SAScores. This pair works well for lower logP values, but for higher ones the variance in the SAScore axis rises significantly. This seems to indicate that in this case, the logP values have a slight priority in the generative process compared to SAScore. There are some outliers, but most of the generated molecules fulfill both conditions. We suspect that this could be an effect of the shortage of training data in that region, thus leading to more inaccurate results.

Next we compare the combination of logP and the molecular weight values, as shown in Fig. 5. Apparently, the molecular weight takes priority in the generation, as it displays a much smaller variance compared to the logP. However, the logP is still met accurately despite being under very strict size constraints. This comes as no surprise, due to the ease with which the molecular weight can be determined by counting the contributions of each

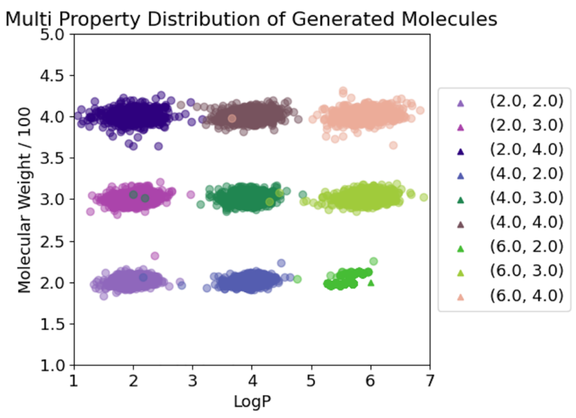


Fig. 5 LogP + Molecular weight

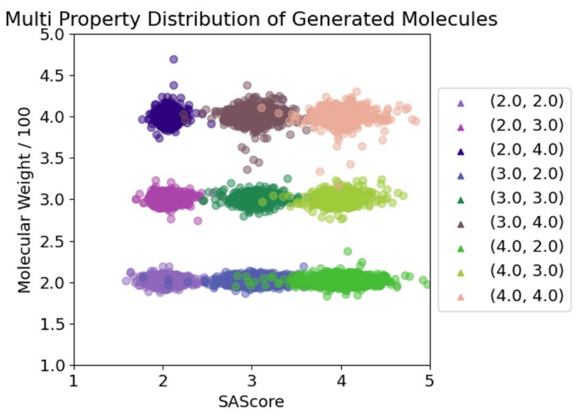


Fig. 6 SAScore + Molecular weight

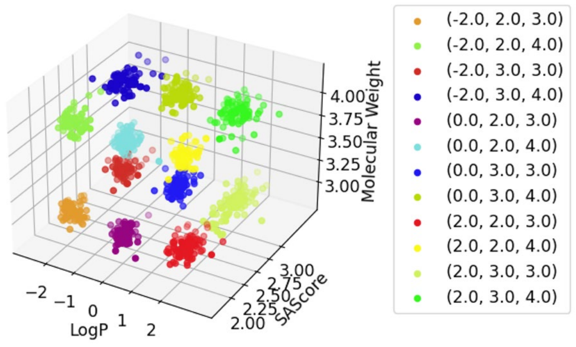


Fig. 7 LogP + SAScore + Molecular weight

atom, as opposed to the more extensive considerations demanded by logP values.

Lastly, Fig. 6 displays the combination of SAScore and molecular weight. Similar to the logP and molecular weight comparison, molecular weight still dominates the generative process. In comparison, the model can not uphold the SAScore in all cases. This is especially evident in the case, where the molecular weight is set to a low value of 1.5 which results in a high SAScore variance. Apparently, the model struggles to incorporate a sufficient number of challenging motifs into a small molecule, due to the limited size and range of available elements In contrast, when the weight is set to a higher value of 3.5, i.e. a larger molecule, we obtain a much lower variance.

Finally, in Fig. 7 we visualize the generated molecules that take into account all three properties. As is evident by the disjoint point clouds in the graph, the model learned to consider all three conditions and generate matching molecules. The labels in the legend should be read in the order of logP, SAScore, and molecular weight.

In Table 4 we compare the performance of our model to other models in multi-conditional generation where applicable. Each row in the table represents an experiment, with the columns representing the properties used as conditions. If a condition was not utilized, the cell was left empty. Our model slightly outperforms the others, in the case of logP + SAScore, in terms of MAD, uniqueness and validity, while simultaneously being able to handle other condition combinations effectively. Again, the novelty falls behind the other models.

Token sequence incorporation

A very common question in material design is to create analogs from a given starting molecule and add/modify structural features to customize the physical properties. For this reason, the model also accepts a SMILES string representing the desired molecular moiety that should be integrated as a building block in the newly generated structure as an additional condition. The requested SMILES string is converted to the canonical form used throughout, before it is tokenized. To measure the performance of our model for this task, we used the following criteria:

- Substructure Matches (SM): The substructure match measures the percentage of generated molecules that explicitly include the target moiety. As a first step, we convert the target structure into a SMARTS [55] pattern, which is essentially a regular expression to match specific atoms or substructures within a molecular structure. To make the criterion a little less strict, all information about bond orders is removed, only the connectivity itself is retained. Therefore, the pattern tolerates modifications in the details of the electronic structure (e.g. localized double bonds versus aromatic bonds), while still maintaining the overall topology. With this property, we measure how

Table 5 Table for comparing metrics on 1000 generated molecules for each context token sequence

	Token sequence SMILES	Unconditional Uniqueness at 1k [%] / SM [%]	LogP {-2, 0, 2} MAD / Uniqueness at 1k [%] / SM [%]	SAScore {2, 3, 4} MAD / Uniqueness at 1k [%] / SM [%]	Molecular Weight {2, 3, 4} MAD / Uniqueness at 1k [%] / SM [%]
1	c1ccccc1 (Benzene)	99.8 / 96.59	0.4 / 99.9 / 75.08	0.15 / 100.0 / 88.17	0.11 / 99.2 / 93.17
2	s1cccc1 (Thiophene)	94.08 / 70.88	0.5 / 97.36 / 53.05	0.15 / 98.39 / 53.33	0.13 / 95.97 / 60.52
3	CC1=CSC=C1 (3-Methylthiophene)	90.78 / 79.86	0.43 / 95.85 / 53.5	0.14 / 93.05 / 62.94	0.14 / 94.04 / 56.97
4	CCO (Ethanol)	99.9 / 61.83	0.17 / 99.8 / 65.73	0.09 / 100.0 / 65.06	0.07 / 99.9 / 54.92
5	CC=O (Acetaldehyde)	99.9 / 89.1	0.19 / 99.2 / 93.67	0.19 / 95.77 / 88.61	0.08 / 97.38 / 91.03
6	CC(=O)OC1=CC=CC=C1C(=O)O (Aspirin)	56.44 / 96.78	0.56 / 73.63 / 87.55	0.2 / 81.45 / 88.41	0.14 / 45.21 / 64.98
7	CC(=O)NC1=CC=C(C =C1)O (Paracetamol)	89.74 / 65.29	0.31 / 92.63 / 72.98	0.15 / 96.12 / 72.55	0.11 / 70.29 / 82.68
8	CN1C=NC2=C1C(=O)N (C(=O)N2C)C (Caffeine)	42.66 / 98.19	0.7 / 70.53 / 91.93	0.23 / 61.69 / 95.69	0.29 / 51.78 / 68.1
9	CN1CCC23C4C1CC5=C2 C(=C(C=C5)O)OC3C (C=C4) O (Morphine)	14.21 / 99.37	0.5 / 46.32 / 94.66	1.85 / 31.86 / 99.49	0.11 / 29.4 / 96.68
10	OC(=O)C(C)c1ccc(cc 1)CC(C) C (Ibuprofen)	33.2 / 44.5	1.03 / 63.65 / 87.46	0.28 / 48.28 / 69.39	0.12 / 30.4 / 66.97

often the target structure is retained during the generative process.

For this experiment, at a constant temperature of 0.8, batches of 1k molecules were generated for various context token sequences and evaluated using the mentioned metric. Table 5 lists different organic target structures (as the context token sequence in SMILES form) and the results obtained a) without applying any other conditions (columns: uniqueness at 1k / SM), and b) with another additional numerical condition (columns: LogP/SAScore/molecular weight at different target values each).

Overall, the model seems to perform very well, as we can recover the target structures at least once in most of the newly generated SMILES. However, especially when given a larger target structure such as Morphine, we observe that the generated structures become very repetitive.

Token sequence with a single numerical condition

The really useful application for customizing given structures is the simultaneous application of one or more additional criteria.

Thus, we study combinations of token sequence conditions together with single numerical conditions, see Table 5. Each combination was tested on 1000 generated molecules, with the numerical values uniformly sampled from the range specified in the table header for each property.

Generally, we observe that smaller token sequences, which naturally occur as building blocks in the training data (entries 1 - 5), are readily integrated into a variety of generated compounds. In contrast, larger compounds (entries 6 - 10) that emerge as independent, self-contained units within the dataset exhibit a significantly lower rate of uniqueness.

In most cases, we observed a decrease in the number of substructure matches for the molecules tested as compared to the previous run without numerical conditions. This is likely due to the model having to handle two possibly competing conditions simultaneously. The MAD values for logP and SAScore were also notably higher compared to generating without a token sequence but remain within acceptable limits. It is worth noting that when the two conditions conflicted, such as with Ibuprofen and negative logP values, this led to the presence of some significant outliers. Conversely, when the conditions aligned well, the errors were consistent with the previous results. More details on the graph for the Ibuprofen and logP relationship can be found in Appendix 5.1. We also observed that the MAD of the SAScore in the case of Morphine is significantly higher than in the other examples. This is mostly due to Morphine having a SAScore of about 5.2, and we requested lower values. In this case, the model prioritizes the token sequence in comparison to the SAScore, which leads to the higher MAD.

Surprisingly, the token sequence condition takes precedence in most cases over the criteria logP and SAScore, as evidenced by the elevated MAD scores. Yet again, the molecular weight seems to be prioritized over the token

Table 6 Table for comparing multiple property conditions for 1000 generated molecules using example token sequences

Token Sequence SMILES	SM[%]	Uniqueness at 1k [%]	LogP {-2, 0, 2} MAD	SAScore {2, 3, 4} MAD	Molecular Weight {2, 3, 4} MAD
C1=CSC=C1 (Thiophene)	42.32	91.72	0.45	0.15	
	37.05	94.88	0.45		0.13
	40.16	98.98		0.15	0.13
	24.02	87.73	0.49	0.18	0.15
CC=O (Acetaldehyde)	92.79	94.19	0.18	0.15	
	96.88	99.20	0.18		0.07
	91.68	98.40		0.14	0.08
	94.65	91.83	0.18	0.14	0.08
CC(=O)NC1=CC=C(C=C1)O (Paracetamol)	71.41	90.61	0.37	0.18	
	75.95	74.00	0.37		0.10
	82.05	81.17		0.35	0.11
	70.20	83.98	0.38	0.29	0.13
CN1C=NC2=C1C(=O)N(C(=O)N2C)C (Caffeine)	89.91	72.48	0.52	0.25	
	60.13	57.36	0.47		0.20
	70.48	49.36		0.35	0.23
	62.70	60.21	0.54	0.37	0.17

sequence, as evidenced by the very low MAD scores, particularly for larger molecules such as Morphine.

Token sequence with multiple numerical conditions

We also conducted experiments where multiple token sequences were tested under two conditions simultaneously. The results of these experiments can be found in Table 6. Each row in the table represents a specific experiment, with the columns representing the properties used as conditions. If a condition was not utilized, the cell was left empty.

The model consistently performs well under various conditions, as shown by the low MAD values. However, when conditions are overly restrictive in combination with the token sequence, it can lead to higher MAD values or lower rates of substructure matches. This is because the model prioritizes certain properties over others.

For instance, consider Paracetamol, where both logP and molecular weight conditions are applied. Due to the constraining effect of molecular weight on the molecule's size, decreasing the logP value significantly becomes challenging. In this case, the model prioritizes the molecular weight condition. We suspect this is because molecular weight is easier to validate and has more pronounced limitations compared to logP.

Nevertheless, the model effectively satisfies all three constraints in most cases, as evidenced by a high percentage of substructure matches and low MAD values for the properties in Table 6. Notably, when generating molecules with three properties, some MAD values are even lower than those observed in two-property generation. This could be attributed to the model being trained on a larger number of three-property batches, resulting in improved performance.

In general, all four conditions are respected during the generative process and make significant contributions to the resulting molecules.

Conclusion

Our aim was to provide a tool for exploring the relevant chemical spaces for a given application, in our case the subspace of organic, potentially electro-active compounds. We therefore adapted existing work and approaches to our needs and came up with a new training variant that allows for a solitary model very flexible in use, which was also trained on a data set of substantial size.

In detail, we

1. developed a GPT-style Transformer based on the Llama 2 architecture, showcasing strong perfor-

mance in both single and multi-conditioned generation, comparable to or slightly surpassing existing models, despite not being task-specific.
2. compiled and utilized a training dataset comprising 12.5 million organic molecules sourced from various origins, enhancing the model's ability to generate a variety of molecular structures.
3. implemented a new training method we call Stochastic Context Learning (SCL), enabling our model to handle various combinations of conditions efficiently for multi-conditional generation using a single model.

We were able to show that the training process was successful and the achieved accuracy very satisfactory. The model generalizes quite well, as target values requested outside the well-sampled areas still tend to fall in the desired ranges. At present, this provides us with promising chemical sub-spaces to screen for electro-active materials, e.g. by feeding the generated SMILES into a trained model for predicting the enthalpy of reaction (as a prerequisite for calculating the redox potential [56]).

The whole setup is very generic and easily adaptable to other applications. The latter motivates the number and choice of properties used as conditions for narrowing down the search space. In fact, for the model to be more useful in the search for energy-storage materials, in future we intend to provide a more meaningful, yet expensive property, such as the enthalpy of reaction.

Looking ahead, this research opens up exciting possibilities for further advancements in generative models and their applications in chemistry and related fields. Our modified architecture, combined with the SCL approach, holds great potential for generating novel and diverse organic molecules with precise control over desired properties.

In theory, a single model can learn a wide range of conditions and combinations by utilizing this approach during training. Therefore, we chose the SAScore (reflecting a materials' production cost), molecular size and logP (contributing to the energy density), as well as a desirable molecular core structure as optional target conditions. As an added benefit, a single model also comes at a reduced training cost. This method enables a more flexible and scalable training process, as it does not require every property to be available for all samples.

Outlook For future work, we intend to focus more on curating the dataset, as to not have these very concentrated distributions for all properties. We hope that by reducing redundant molecules, the model would generalize better, while also reducing training time in the process. Generally, we assume that the model could perform even better with more training data, as it seems to be underfitted even with our large dataset.

Furthermore, we also intend to expand the number of properties that are given to the model, as there are more useful conditions for practical applications, such as the HOMO-LUMO gap.

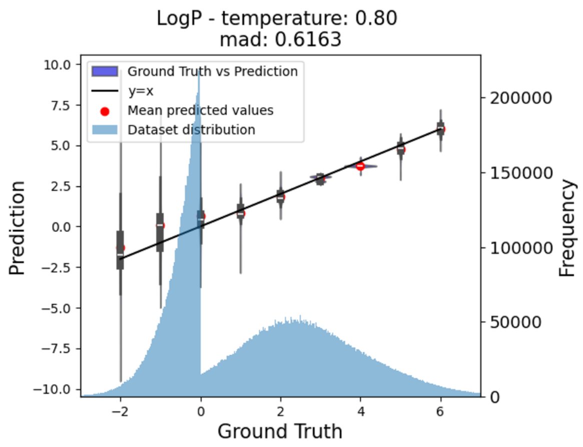


Fig. 8 Ibuprofen logP Graph - Generated vs Target

Appendix

A

In this chapter, we visualize the errors of generated molecules using the molecular fragment condition with a single numerical condition.

In the case of Ibuprofen with a naturally very positive logP of about 3.0, it is very difficult for the model to significantly reduce the logP to the desired negative values, while also keeping the fragment intact. This leads to an overall higher MAD, due to a small sample of large outliers that increased the mean by a significant margin. This can be seen in the Fig. 8.

B

We also conducted some experiments on special combinations of different conditions, as these also show the limitations of the model, either due to the incompatibility of these conditions or the lack of training data in those regions.

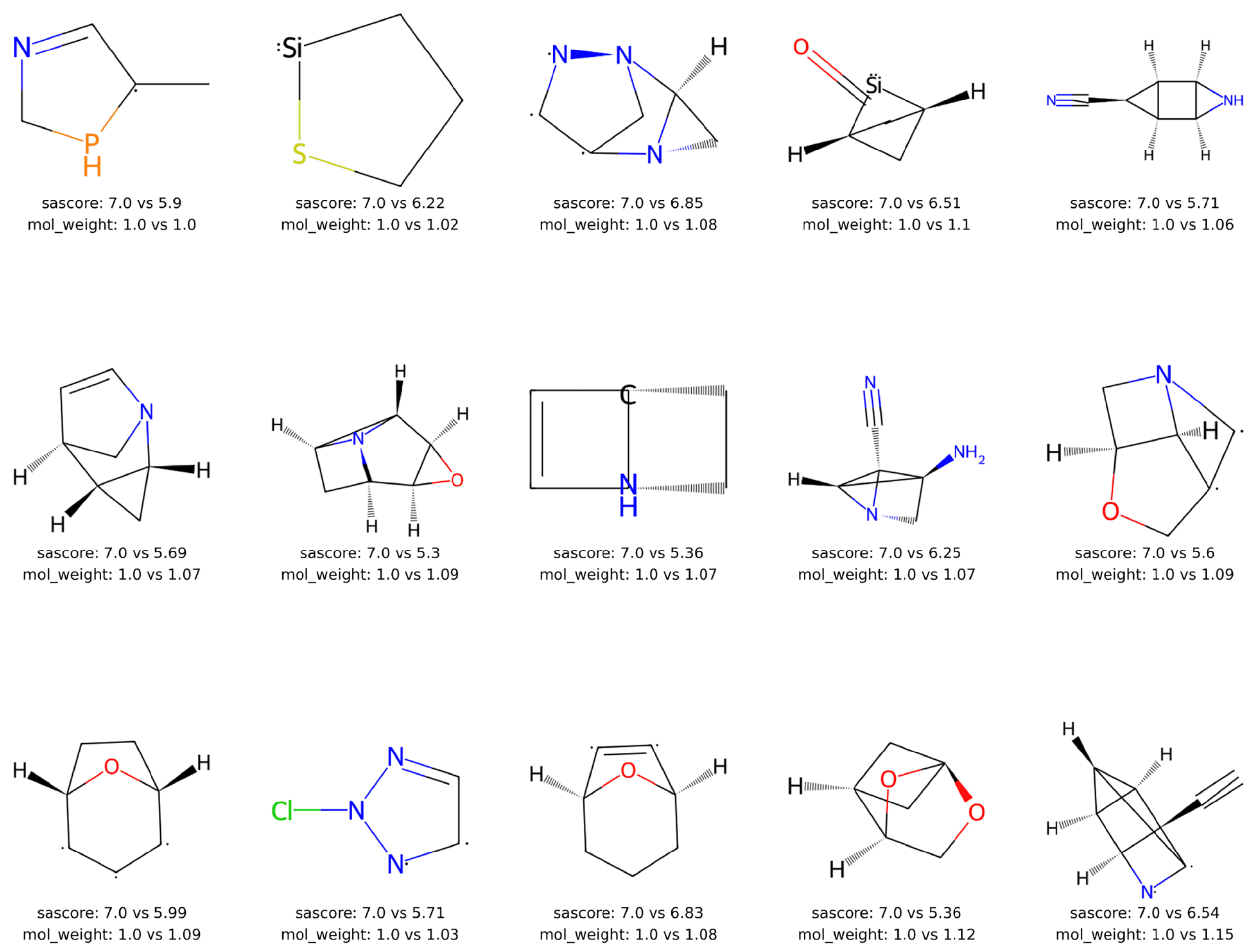


Fig. 9 Special Case: Generated molecules with low molecular weight and high SAScore as conditioning

We tested the combination of a low molecular weight (100) and a high SAScore (7), which can be seen in the Fig. 9. The generated molecules have both characteristics by being hard to produce due to the high number of connected, bridged, annealed or spiro-rings and ring strains associated with the high degree of interconnected rings and/or open-shell centers (radicals and/or carbenes), while keeping the molecular weight small. In this scenario, it also uses more uncommon elements to fit into both conditions.

C

In this section are a sample of the generated molecules for each property visualized. Figure 10 showcases examples that are generated with logP as a property from negative to positive values. Furthermore, the Fig. 11 show the change over different SAScores. Lastly, the Fig. 12 shows how the generated molecules get larger with a rising molecular weight.

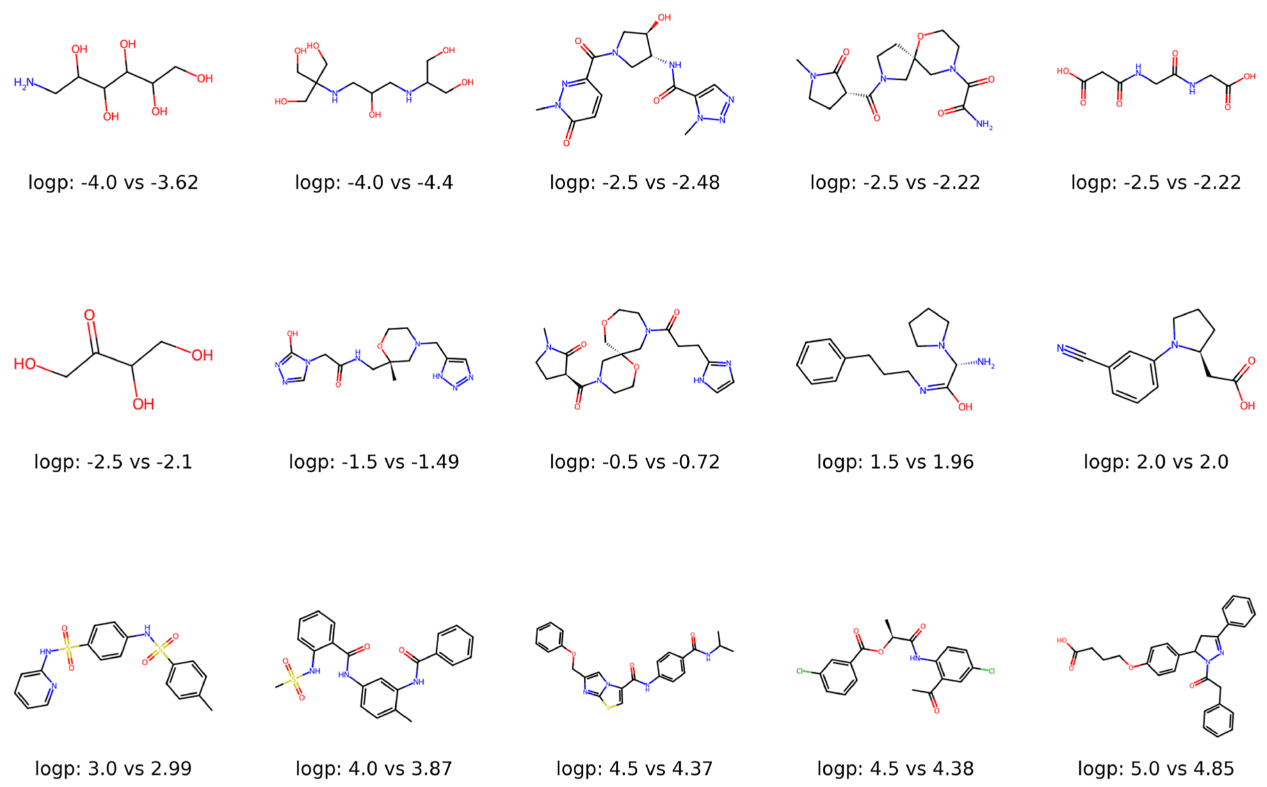


Fig. 10 A sample of the generated molecules with logP as conditioning

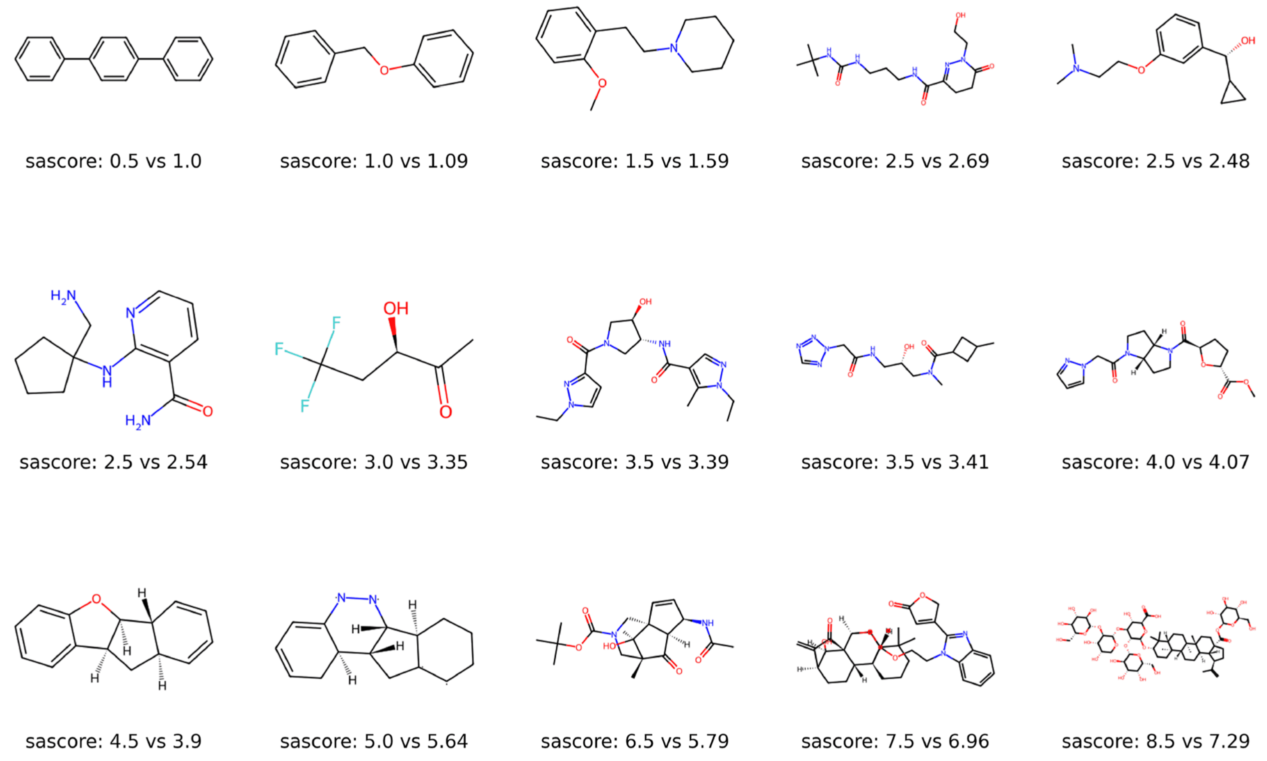


Fig. 11 A sample of the generated molecules with SAScore as conditioning

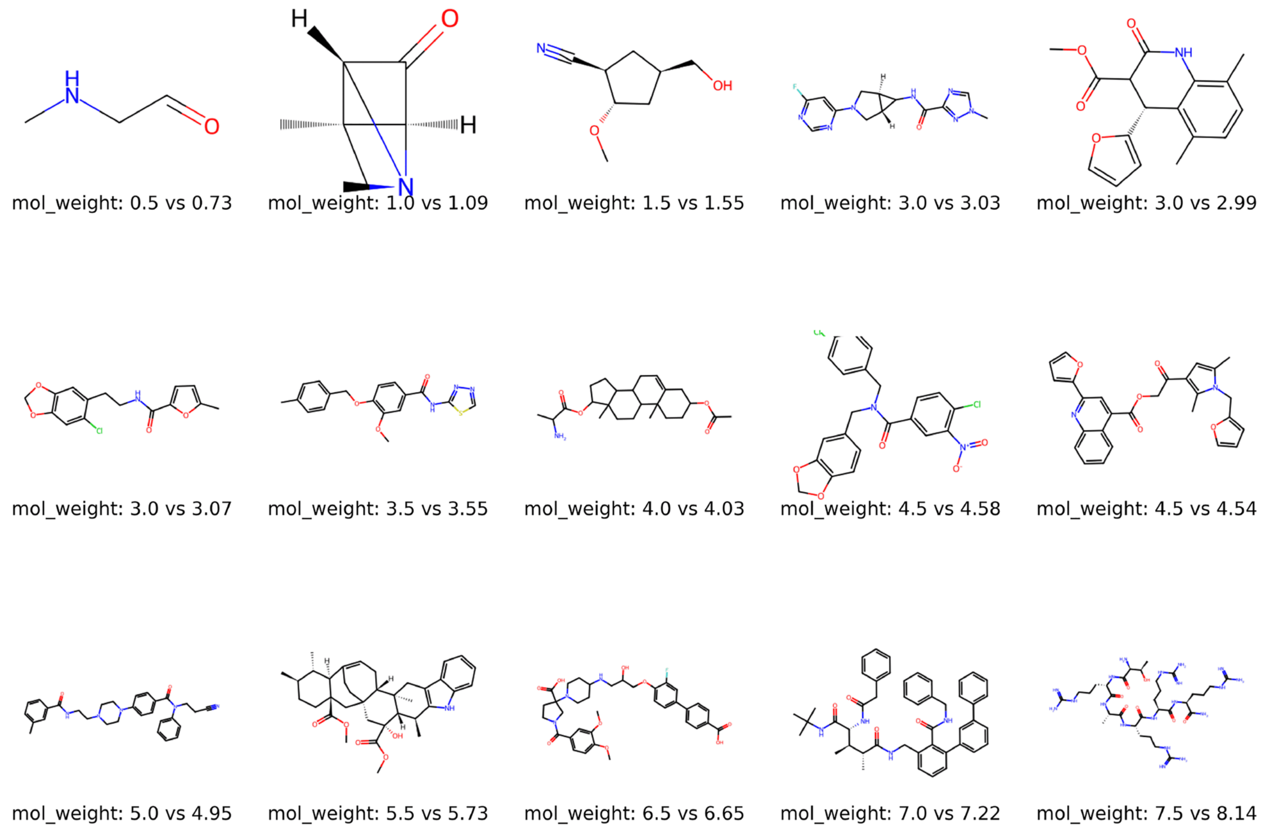


Fig. 12 A sample of the generated molecules with the molecular weight as conditioning

Author contributions
N.D. wrote the main manuscript. A.M. helped in writing and refining parts of the manuscript and gave input on the chemical application of the model. J.H. supervised the paper and helped with the mathematical background. All authors reviewed and refined the manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL.

Data availability
The model and training data is available at our Github: https://github.com/Fraunhofer-SCAI/llamol.

Declarations

Competing interests
The authors declare no competing interests.

Received: 9 January 2024 Accepted: 19 May 2024
Published online: 21 June 2024

References

1. Sherstinsky A (2020) Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena 404:132306. https://doi.org/10.1016%2Fj.physd.2019.132306
2. Goodfellow IJ et al (2014) Generative adversarial networks. http://arxiv.org/abs/1406.2661arXiv:1406.2661
3. Kingma DP, Welling M (2022) Auto-encoding variational bayes. http://arxiv.org/abs/1312.6114arXiv:1312.6114
4. Vaswani A, et al (2017) Attention is all you need. http://arxiv.org/abs/1706.03762arXiv:1706.03762
5. Brown TB, et al (2020) Language models are few-shot learners. http://arxiv.org/abs/2005.14165arXiv:2005.14165
6. Dosovitskiy A, et al (2021) An image is worth 16x16 words: Transformers for image recognition at scale. http://arxiv.org/abs/2010.11929arXiv:2010.11929
7. Urbina F, Lowden CT, Culberson JC, Ekins S (2022) MegaSyn: integrating generative molecular design, automated analog designer, and synthetic viability prediction. ACS Omega 7:18699–18713
8. Bagal V, Aggarwal R, Vinod PK, Priyakumar UD (2021) MolGPT: molecular generation using a transformer-decoder model. J Chem Inf Modeling 62:2064–2076. https://doi.org/10.1021/acs.jcim.1c00600
9. Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on gdb-17 data. J Comput-aided Mol Design 27:675–679
10. Richards RJ, Groener AM (2022) Conditional β-vae for de novo molecular generation. http://arxiv.org/abs/2205.01592arXiv:2205.01592
11. Lim J, Ryu S, Kim JW, Kim WY (2018) Molecular generative model based on conditional variational autoencoder for de novo molecular design. J Cheminf https://doi.org/10.1186/s13321-018-0286-7
12. Lee M, Min K (2022) Mgcvae: multi-objective inverse design via molecular graph conditional variational autoencoder. J Chem Inf Modeling 62:2943–2950. https://doi.org/10.1021/acs.jcim.2c00487
13. Cao ND, Kipf T (2022) Molgan: An implicit generative model for small molecular graphs. http://arxiv.org/abs/1805.11973arXiv:1805.11973
14. Grisoni F, Moret M, Lingwood R, Schneider G (2020) Bidirectional molecule generation with recurrent neural networks. J Chem Inf Modeling 60:1175–1183. https://doi.org/10.1021/acs.jcim.9b00943

15. Kotsias P-C et al (2020) Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. Nat Mach Intell 2:254–265. https://doi.org/10.1038/s42256-020-0174-5
16. S V, S S, et al (2022) Multi-objective goal-directed optimization of de novo stable organic radicals for aqueous redox flow batteries. Nat Mach Intell 4:720–730. https://doi.org/10.1038/s42256-022-00506-3
17. Chen Y et al (2023) Molecular language models: RNNs or transformer? Brief Functional Genom 22:392–400. https://doi.org/10.1093/bfgp/elad012
18. Wang J et al (2021) Multi-constraint molecular generation based on conditional transformer, knowledge distillation and reinforcement learning. Nat Mach Intell 3:914–922. https://doi.org/10.1038/s42256-021-00403-1
19. Wang Y, Zhao H, Sciabola S, Wang W (2023) cMolGPT: a conditional generative pre-trained transformer for target-specific de novo molecular generation. Molecules 28:4430. https://doi.org/10.3390/molecules28114430
20. Kim H, Na J, Lee WB (2021) Generative chemical transformer: neural machine learning of molecular geometric structures from chemical language via attention. J Chem Inf Modeling 61:5804–5814. https://doi.org/10.1021/acs.jcim.1c01289. (PMID: 34855384)
21. Du Y, Fu T, Sun J, Liu S (2022) Molgensurvey: a systematic survey in machine learning models for molecule design. http://arxiv.org/abs/2203.14500arXiv:2203.14500
22. Weininger D (1988) Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. J Chem Inf Comput Sci 28:31–36. https://doi.org/10.1021/ci00057a005
23. Touvron H, et al (2023) Llama 2: open foundation and fine-tuned chat models. http://arxiv.org/abs/2307.09288arXiv:2307.09288
24. Ertl P, Schuffenhauer A (2009) Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. J Cheminf https://doi.org/10.1186/1758-2946-1-8
25. Shazeer N (2020) Glu variants improve transformer. http://arxiv.org/abs/2002.05202arXiv:2002.05202
26. Ainslie J, et al (2023) Gqa: Training generalized multi-query transformer models from multi-head checkpoints. http://arxiv.org/abs/2305.13245arXiv:2305.13245
27. Su J, et al (2022) Roformer: Enhanced transformer with rotary position embedding. http://arxiv.org/abs/2104.09864arXiv:2104.09864
28. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. http://arxiv.org/abs/1607.06450arXiv:1607.06450
29. Zhang B, Sennrich R (2019) Root mean square layer normalization. http://arxiv.org/abs/1910.07467arXiv:1910.07467
30. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958
31. Sterling T, Irwin JJ (2015) ZINC 15 - ligand discovery for everyone. J Chem Inf Modeling 55:2324–2337. https://doi.org/10.1021/acs.jcim.5b00559
32. Ramakrishnan R, Dral PO, Rupp M, von Lilienfeld OA (2014) Quantum chemistry structures and properties of 134 kilo molecules. Scientific Data 1. https://doi.org/10.1038%2Fsdata.2014.22
33. Ruddigkeit L, van Deursen R, Blum LC, Reymond J-L (2012) Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. J Chem Inf Modeling 52:2864-2875. https://doi.org/10.1021%2Fci300415d
34. Gómez-Bombarelli R et al (2018) Automatic chemical design using a data-driven continuous representation of molecules. ACS Central Sci 4:268-276. https://doi.org/10.1021%2Facscentsci.7b00572
35. Sorkun E, Zhang Q, Khetan A, Sorkun MC, Er S (2022) RedDB, a computational database of electroactive molecules for aqueous redox flow batteries. Sci Data 9.https://doi.org/10.1038%2Fs41597-022-01832-2
36. John PCS et al (2019) Message-passing neural networks for high-throughput polymer screening. J Chem Phys 150:234111. https://doi.org/10.1063%2F1.5099132
37. Nakata M, Shimazaki T, Hashimoto M, Maeda T (2020) PubChemQC PM6: data sets of 221 million molecules with optimized molecular geometries and electronic properties. J Chem Inf Modeling 60:5891–5899
38. Nakata M, Shimazaki T (2017) PubChemQC project: a large-scale first-principles electronic structure database for data-driven chemistry. J Chem Inf Model 57:1300–1308
39. Hachmann J et al (2011) The Harvard clean energy project: Large-scale computational screening and design of organic photovoltaics on the world community grid. The Journal of Physical Chemistry Letters 2:2241-2251. https://doi.org/10.1021%2Fjz200866s
40. Duvenaud D, et al (2015) Convolutional networks on graphs for learning molecular fingerprints. http://arxiv.org/abs/1509.09292arXiv:1509.09292
41. Zdrazil B et al (2023) The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. Nucleic Acids Res 52:D1180–D1192. https://doi.org/10.1093/nar/gkad1004
42. Blackshaw J et al (2009) CHEMBL database release 31. https://doi.org/10.6019/chembl.database.31
43. Davies M et al (2015) Chembl web services: streamlining access to drug discovery data and utilities. Nucleic Acids Res 43:W612–W620. https://doi.org/10.1093/nar/gkv352
44. Jupp S et al (2014) The ebi rdf platform: linked open data for the life sciences. Bioinformatics 30:1338–1339. https://doi.org/10.1093/bioinformatics/btt765
45. Schwaller P et al (2019) Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. ACS Central Sci 5:1572–1583. https://doi.org/10.1021/acscentsci.9b00576
46. Ramsundar B, et al (2019) Deep Learning for the Life Sciences O'Reilly Media. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837
47. Loshchilov I, Hutter F (2019) Decoupled weight decay regularization. http://arxiv.org/abs/1711.05101arXiv:1711.05101
48. Landrum G, et al (2020) rdkit/rdkit: 2020_03_1 (q1 2020) release. https://doi.org/10.5281/zenodo.3732262
49. Lu H, Wei Z, Wang X, Zhang K, Liu H (2023) Graphgpt: A graph enhanced generative pretrained transformer for conditioned molecular generation. Int J Mol Sci 24. https://www.mdpi.com/1422-0067/24/23/16761
50. Polykovskiy D et al (2020) Molecular sets (moses): a benchmarking platform for molecular generation models. Front Pharmacol. https://doi.org/10.3389/fphar.2020.565644
51. Brown N, Fiscato M, Segler MH, Vaucher AC (2019) Guacamol: benchmarking models for de novo molecular design. J Chem Inf Modeling 59:1096–1108. https://doi.org/10.1021/acs.jcim.8b00839
52. Işık M et al (2020) Assessing the accuracy of octanol-water partition coefficient predictions in the sampl6 part ii log p challenge. J Comput-aided Mol Design 34:335–370
53. Haroon S, CA H, AS J, (2023) Generative pre-trained transformer (gpt) based model with relative attention for de novo drug design. Comput Biol Chem 106:107911. https://doi.org/10.1016/j.compbiolchem.2023.107911
54. Monteiro NR et al (2023) Fsm-ddtr: end-to-end feedback strategy for multi-objective de novo drug design using transformers. Comput Biol Med 164:107285. https://doi.org/10.1016/j.compbiomed.2023.107285
55. Daylight Theory: SMARTS - A language for describing molecular patterns – daylight.com. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html. [Accessed 03-11-2023]
56. Barker J, Berg L-S, Hamaekers J, Maass A (2021) Rapid prescreening of organic compounds for redox flow batteries: a graph convolutional network for predicting reaction enthalpies from smiles. Batteries Supercaps 4:1482–1490

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.