# LigGPT: Molecular Generation using a Transformer-Decoder Model

Viraj Bagal,[†,‡] Rishal Aggarwal,[†] P. K. Vinod,[†] and U. Deva Priyakumar[*,†]

[†]*International Institute of Information Technology, Hyderabad 500 032, India*

[‡]*Indian Institute of Science Education and Research, Pune 411 008, India*

E-mail: deva@iiit.ac.in

**Abstract**

Application of deep learning techniques for the *de novo* generation of molecules, termed as inverse molecular design, has been gaining enormous traction in drug design. The representation of molecules in SMILES notation as a string of characters enables the usage of state of the art models in Natural Language Processing, such as the Transformers, for molecular design in general. Inspired by Generative Pre-Training (GPT) model that have been shown to be successful in generating meaningful text, we train a Transformer-Decoder on the next token prediction task using masked self-attention for the generation of druglike molecules in this study. We show that our model, LigGPT, outperforms other previously proposed modern machine learning frameworks for molecular generation in terms of generating valid, unique and novel molecules. Furthermore, we demonstrate that the model can be trained conditionally to optimize multiple properties of the generated molecules. We also show that the model can be used to generate molecules with desired scaffolds as well as desired molecular properties, by passing these structures as conditions, which has potential applications in lead optimization in addition to de novo molecular design. Using saliency maps, we highlight the interpretability of the generative process of the model.

## Introduction

It has been postulated that the total number of potential drug like candidates range from $10^{23}$ to $10^{60}$ molecules[1] of which, only about $10^{8}$ molecules have been synthesized[2]. Since it is difficult to screen a practically infinite chemical space, and there is a a huge disparity between synthesized and potential molecules, generative models are used to model a distribution of molecules for the purpose of sampling molecules that have desirable properties. Deep generative models have made great strides in modeling data distributions in general data domains such as Computer Vision[3,4] and Natural Language Processing (NLP)[5,6]. Such methods have also been adopted to model molecular distributions[7,8]. Such models learn probability distributions over a large set of molecules and therefore are able to generate novel molecules by sampling from these distributions[7,9]. The rapid adoption of deep generative model has also led to the development of benchmark datasets such as the Molecular Sets (MOSES)[10] and GuacaMol[9] datasets.

The representation of molecules in Simplified Molecular Input Line Entry System (SMILES)[11] notation as a string of characters enables the usage of modern NLP deep learning models for their computation[12]. Some of the earliest deep learning architectures for molecular generation involved the usage of Recurrent Neural Networks (RNNs) on molecular SMILES[13,14]. Such models have also previously been trained on large corpus of molecules and then focused through the usage of reinforcement learning[15,16] or transfer learning[13] to generate molecules of desirable properties and activity.

Auto-Encoder variants such as the Variational Auto-Encoder(VAE)[17–21] and Adversarial Auto-Encoder(AAE)[22–25] have also been employed for molecular generation. These models contain an encoder that encodes molecules to a latent vector representation and a decoder that maps latent vectors back to molecules. Molecules can then be generated by sampling from these latent spaces. However, SMILES based methods often lead to a discontinuous latent space[20] since similar molecules can have very different SMILES representations. To address this problem, SMILES strings have also been randomized as inputs for such models[26–28]. Junction Tree VAE (JT-VAE)[20] is also an alternative solution which represents molecules as graph tree structures. JT-VAE also ensures 100% validity of generated molecules by maintaining a vocabulary of molecular components that can be added at each junction of the molecule tree. Conditional Variational Auto-Encoders have also been used to generate molecules with desired properties[29].

Generative Adversarial Networks (GANs) have also gained traction for molecular design[30–34]. This is mainly because of their ability to generate highly realistic content[4]. GANs are composed of generators and discriminators that work in opposition of each other. While the generator tries to generate realistic content, the discriminator tries to distinguish between generated and real content. ORGAN[31] was the first usage of GANs for molecular generation. RANC[34] introduced reinforcement learning alongside a GAN loss to generate molecules of desirable properties. LatentGAN[30] is a more recent method which uses latent vectors as input and outputs. These latent vectors are mapped to molecules by the decoder of a pre-trained auto-encoder. This ensures that the model can work with latent representations and doesn't have to handle SMILES syntax. Most of these methods have been benchmarked on the MOSES and GuacaMol datasets for easy comparison.

Often, methods use Bayesian optimization[35,36], reinforcement learning[15,34] or other optimization methods[37,38] to generate molecules exhibiting desirable properties. Mol-CycleGAN[39] is a generative model that utilises the JT-VAE architecture and applies the CycleGAN[40] loss to generate molecules of required properties using given molecule templates. Only a few methods explicitly define the values of properties for generated molecules. Conditional RNNs[13,41], Deep learning enabled inorganic material generator (DING)[29] and Conditional Adversarially Regularized Autoencoder (CARAE)[25] are three such methods that sample molecules based on exact values. RNNs have also been previously used to generate molecules based on given scaffolds[42]. A graph based method has been designed that ensures the presence of desired scaffolds while generating molecules with exact property values[43].

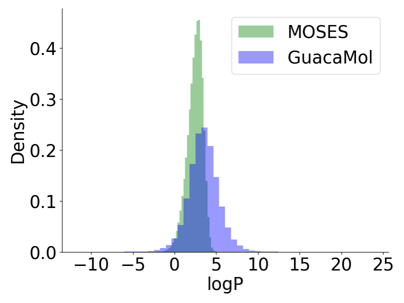
A novel NLP architecture called the Transformer[5] has shown state-of-the-art performance in language translation tasks. Transformers consist of encoder and decoder modules. The encoder module gains context from all the input tokens through self attention mechanisms. The decoder module gains context from both the encoder as well as previously generated tokens by attention. Using this context the decoder is able to predict the next token. The decoder module has also been previously used independently for language modeling task and is known as the Generative Pre-Training Transformer model (GPT)[44]. The GPT model has been shown to develop better language embedding that model longer-distance connections. Due to this, the embeddings have shown top performance when used for multiple language modelling tasks such as natural language inference, question answering, sentence similarity and classification[45].

To yield the added benefits of this architecture, we train a a GPT model, named LigGPT, to predict a sequence of SMILES tokens for molecular generation. To the best of our knowledge, this is the first work that has used the GPT architecture for molecular generation. For this, we use a SMILES tokenizer to break SMILES strings into a set of relevant tokens. Since predicted tokens are a result of attention applied to all previously generated tokens, we believe, that the model easily learns the SMILES grammar and therefore, can focus on higher level understanding of molecular properties. To this end, we also train our models conditionally to explicitly learn certain molecular properties. The model displays performance that is at par with other methods benchmarked on the MOSES and Guacamol datasets. Furthermore, we show that LigGPT controls user specified molecular properties and scaffolds with extremely high accuracy, leading to our conclusion that it learns a good representation of the chemical space. Due to this learned representation, the utility of LigGPT in molecular design processes such as lead optimization is also highlighted.

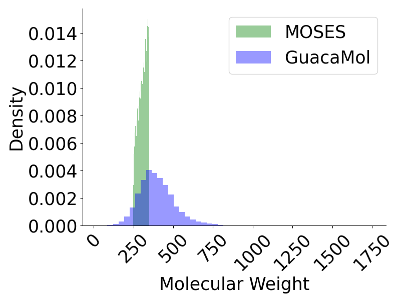
## Methods

In this section, we first present the datasets used for all the experiments. We discuss the properties used for conditional generation. This is then followed by the overview of the proposed model. Schematic of the training and generation pipeline is shown in this section. At last, the details of the experiments and the metrics used for the evaluation of different models are provided.

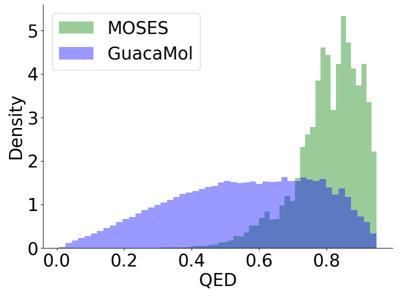
### Datasets



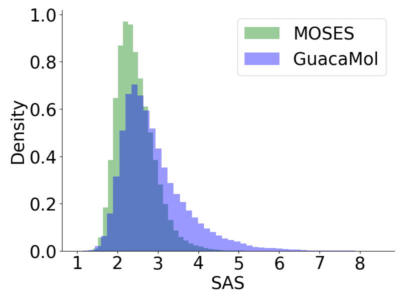
(a)



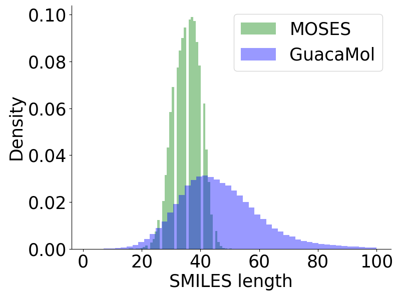
(b)



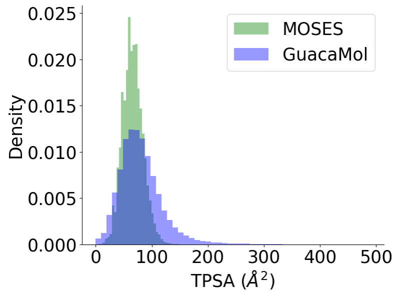
(c)



(d)



(e)



(f)

Figure 1: Probability distributions of properties (log P, molecular weight, QED, SAS, TPSA and SMILES length) of molecules in the MOSES and GuacaMol datasets.

In this work, we used two benchmark datasets, MOSES and GuacaMol for training and evaluation of our model. MOSES is a dataset composed of 1.9 million lead-like molecules with molecular weight ranging from 250 to 350 Daltons, number of rotatable bonds lower than 7 and and XlogP below 3.5. GuacaMol on the other hand is a subset of the ChEMBL 24[46] database that contains 1.6 Million molecules. We used the rdkit toolkit[47] to calculate

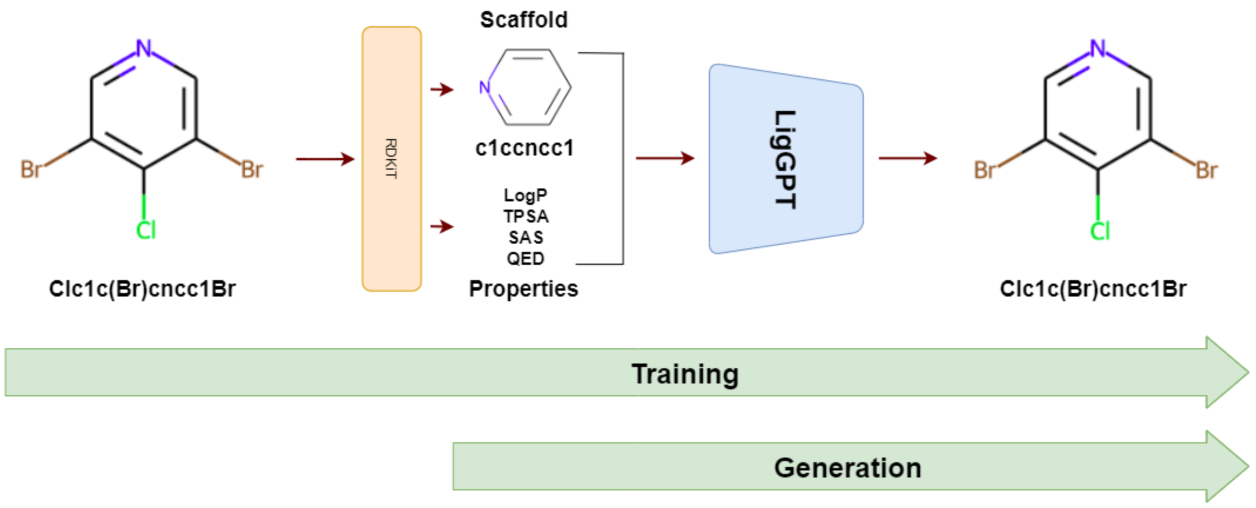


Figure 2: Pipeline for Training and Generation using the LigGPT model.

molecular properties and to extract Bemis-Murcko scaffolds.[48] The MOSES dataset was created mainly to represent lead like molecules and therefore has a distribution of molecules with ideal druglike properties. However, to test the models control on conditional generation we require a larger distribution of property values available in the Guacamol dataset as can be seen in Figure 1. Therefore, we use the GuacaMol dataset to test property conditional generation. The MOSES dataset also provides a test set of scaffolds which we use to evaluate scaffold and property conditional generation.

The models were trained to learn some properties of the molecules for controlled generation and optimisation. The properties used are the following:

- **logP**: the logarithm of the partition coefficient. Partition coefficient compares the solubilities of the solute in two immiscible solvents at equilibrium. If one of the solvents is water and the other is a non-polar solvent, then logP is a measure of hydrophobicity.

- **Synthetic Accessibility score (SAS[49])**: measures the difficulty of synthesizing a compound. It is a score between 1 (easy to make) and 10 (very difficult to make).

- **Topological Polar Surface Area (TPSA)**: the sum of surface area over all polar atoms. It measures the drug's ability to permeate cell membranes. Molecules with TPSA greater than 140 Å$^2$ tend to be poor in permeating cell membranes.

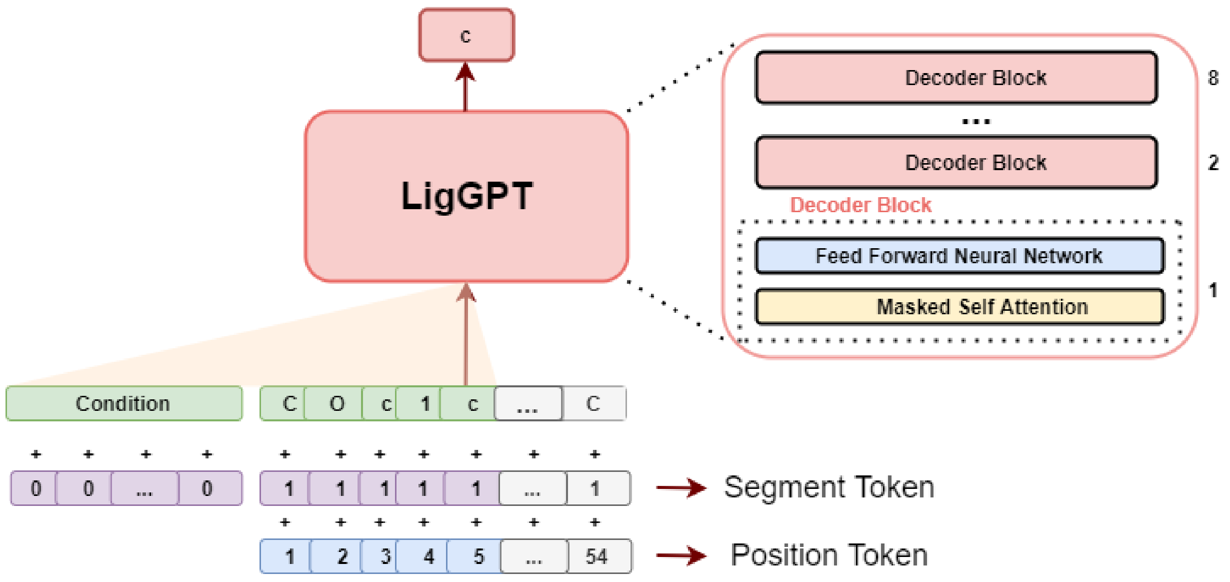


Figure 3: LigGPT model architecture.

- **Quantitative Estimate of Drug-likeness (QED[50])**: This quantifies drug-likeness by taking into account the main molecular properties. It ranges from zero (all properties unfavourable) to one (all properties favourable).

## Model Overview

The model schematic of LigGPT for training and generation is given in Figure 2. For training, we extract molecular properties and scaffolds from molecules using rdkit and pass them as conditions alongside the molecular SMILES. For generation, we provide the model a set of property and scaffold conditions along with a start token to sample a molecule.

Our model is illustrated in Figure 3. The model is essentially a mini version of the Generative Pre-Training Transformer (GPT) model[44]. Unlike GPT1 that has around 110M parameters, LigGPT has only around 6M parameters. LigGPT comprises stacked decoder blocks, each of which, is composed of a masked self-attention layer and fully connected neural network. Each self-attention layer returns a vector of size 256, that is taken as input by the fully connected network. The hidden layer of the neural network outputs a vector of size 1024 and uses a GELU activation and the final layer again returns a vector of size 256 to be used as input for the next decoder block. LigGPT consists of 8 such decoder blocks.

To keep track of the order of the input sequence, position values are assigned to each token. During conditional training, segment tokens are provided to distinguish between the condition and the SMILES tokens. All the tokens are mapped to the same space using their respective embedding layers. All the embeddings are then added, resulting in a vector of size 256, which is then passed as input to the model.

GPT architectures work on a masked self-attention mechanism. Self-attention is calculated through 'Scaled Dot Product Attention'. This involves 3 sets of vectors, the query, key and value vectors. Query vectors are used to query the weights of each individual value vector. They are first sent through a dot product with key vectors. These dot products are scaled by the dimensions of these vectors and then a softmax function is applied to get the corresponding weights. The value vectors are multiplied by their respective weights and added. The query, key and value vectors for each token are computed by weight matrices present in each decoder block. Attention can be represented by the following formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Where $Q$, $K$ and $V$ are query, key and value vectors respectively. $d_k$ here is the dimension of query and key vectors and $T$ is transpose of the matrix.

Self-attention provides attention to all the tokens of a sequence for prediction. However, this is not ideal when we are training a model to predict the next token in a sequence. Therefore, masked self attention is applied to mask attention to all sequence tokens that occur in future time steps. It is essential because, during generation, the network would have access only to the tokens predicted in the previous time-steps. Moreover, instead of performing a single masked self-attention operation, each block performs multiple masked self-attention operations (multi-head attention) in parallel and concatenates the output.

Multi-head attention provides better representations by attending to different representation subspaces at different positions.

We train this model on molecules represented as SMILES string. For this, we use a SMILES tokenizer to break up the string into a sequence of relevant tokens. Property conditions are also sent through a linear layer that maps the condition to a vector of 256 dimensions. The resultant vector is then concatenated at the start of the sequence of the embeddings of the SMILES tokens. For scaffold condition, we use an embedding layer to map the tokens to 256-dimensional vectors as well. This embedding layer is shared with the molecule embedding layer. Similar to property condition, the scaffold representation is then concatenated at the start of the sequence of the embeddings of the SMILES tokens. The model is trained such that the predicted tokens are a result of attention to both the previous tokens as well as the conditions.

## Training Procedure and Evaluation Metrics

Each model is trained for 10 epochs using the Adam optimizer with a learning rate of $6e-4$. During generation a start token of a single carbon atom is provided to the network along with the conditions.

We trained and tested LigGPT on both the MOSES[10] and GuacaMol[9] datasets. We also conducted experiments to check LigGPT's capacity to control molecular properties and core structures. The models were trained on a NVIDIA 2080Ti GPU. Most of the models converged and showed best performance after 10 epochs. However, we noticed that training them for slightly fewer epochs led to similar results in terms of validity, novelty and uniqueness of generated molecules, which are the metrics used here (details below).

- **Validity**: the fraction of generated molecules that are valid. We use rdkit for validity check of molecules. Validity measures how well the model has learnt the SMILES grammar and the valency of atoms.

- **Uniqueness**: the fraction of valid generated molecules that are unique. Low uniqueness highlights repetitive molecule generation and low level of distribution learning by the model.
- **Novelty**: the fraction of valid unique generated molecules that are not in the training set. Low novelty is a sign of overfitting. We don't want the model to memorize the training data.
- **Internal Diversity ($\mathbf{IntDiv_p}$)**: measures the diversity of the generated molecules. This uses the Tanimoto similarity ($T$) between the fingerprints of each pair of molecules in the generated set ($S$).

$$IntDiv_p(S) = 1 - \sqrt[p]{\frac{1}{|S|^2} \sum_{s1, s2 \in S} T(s1, s2)^p}$$

The method implementation is available at `https://github.com/devalab/liggpt`

## Results and Discussion

In this section, we first present the results on unconditional generation of molecules. LigGPT's performance is then compared with other state-of-the-art approaches, followed by some insights on the interpretability of our model. We then demonstrate our model's ability of conditional generation based on property alone and scaffold alone. This is followed with the results on conditional generation based on property and scaffold together. At last, a potential use case of our model is demonstrated for one-shot optimization of QED value of a starting molecule.

### Unconditional Molecular Generation

The chemical space is practically inifinite due to which, the models are required to exhibit high validity, uniqueness and novelty scores with respect to molecular generation. High values

Table 1: Comparison of the different metrics corresponding to unconditional generation of molecules using different approaches trained on MOSES dataset. Temperature value of 1.6 was used for LigGPT.

| Models | Validity | Unique@10K | Novelty | $\text{IntDiv}_1$ | $\text{IntDiv}_2$ |
|---|---|---|---|---|---|
| CharRNN | 0.975 | **0.999** | 0.842 | 0.856 | 0.85 |
| VAE | 0.977 | 0.998 | 0.695 | 0.856 | 0.85 |
| AAE | 0.937 | 0.997 | 0.793 | 0.856 | 0.85 |
| LatentGAN | 0.897 | 0.997 | **0.949** | 0.857 | 0.85 |
| JT-VAE | **1.0** | **0.999** | 0.9143 | 0.855 | 0.849 |
| LigGPT | 0.9 | **0.999** | 0.941 | **0.871** | **0.865** |

Table 2: Comparison of the different metrics corresponding to unconditional generation of molecules using different approaches trained on GuacaMol dataset. Temperature value of 0.9 was used for LigGPT.

| Models | Validity | Unique | Novelty |
|---|---|---|---|
| SMILES LSTM | 0.959 | **1.0** | 0.912 |
| AAE | 0.822 | **1.0** | 0.998 |
| Organ | 0.379 | 0.841 | 0.687 |
| VAE | 0.870 | 0.999 | 0.974 |
| LigGPT | **0.986** | 0.998 | **1.0** |

of these metrics would ensure that the models have learnt the molecule grammar well and are not overfitting to the training data simultaneously. Internal diversity scores give an idea about the extent of chemical space traversed by different models. So, LigGPT is compared with previous approaches on these criteria. We compare the performance of LigGPT on the MOSES dataset to that of CharRNN, VAE, AAE, LatentGAN and JT-VAE. JT-VAE uses graphs as input while the others use SMILES. To get the optimal model for each dataset, the generative performance of LigGPT is checked for several sampling temperature values between 0.7 and 1.6. The model performs best at a temperature of 1.6 for MOSES and 0.9 for GuacaMol. The optimal model performance on each dataset is reported in Table 1 and Table 2.

On the MOSES benchmark, LigGPT performs the best in terms of the two internal diversity metrics. This indicates that even though LigGPT learns from the same chemical space as other models, it is better than others in generating molecules with lower redundancy. In case of validity, as mentioned earlier, JT-VAE always generates a valid molecule because it checks validity at every step of generation. Barring JT-VAE, we observe that CharRNN, VAE and AAE have high validity but low novelty. Compared to these three & JT-VAE, LigGPT has lower validity but much higher novelty which is more crucial for the purposes of molecular design. We find that the performance of LigGPT is comparable to LatentGAN. LatentGAN involves training of an autoencoder followed by the training of GAN on the latent space of the trained autoencoder. This is a 2-step process while on the other hand, LigGPT is trained end-to-end. LigGPT's validity and uniqueness are slightly higher than LatentGAN, but novelty scores are comparable. On the GuacaMol benchmark, LigGPT is easily the preferred method when compared to other methods tested on it. It returns very high validity, uniqueness and novelty scores on generation with a sampling temperature of 0.9. We believe this boost in performance, as compared to MOSES, is due to a larger diversity in molecules in the GuacaMol dataset (see Figure 1). Moreover, even though GuacaMol dataset has larger molecules as compared to MOSES dataset, LigGPT generates molecules with very high validity which indicates that this method handles long range dependencies very well. LigGPT takes only about 10 minutes to be trained on an epoch of the GuacaMol dataset. This is due to the parallel nature in which it computes self-attention as opposed to the sequential nature of training for other models. Thus, it can quickly be retrained on a new dataset or on a different set of properties.

While it is important to develop machine learning frameworks and pipelines for making tasks more efficient, it is desirable to also demonstrate that these models allow for interpretation. Figure 4 shows input saliency maps for some of the generated tokens of the shown generated molecule. Input saliency methods assign a score to each input token that indicates the importance of that token in generating the next token. '(', 'C' and 'c' refer to the branching from chain, non-aromatic carbon and aromatic carbon respectively. From Figure 4, we see that when generating the 'O' atom in the first saliency map, the model rightly attends to the previous double bond and 'N' atoms. Double bond satisfies the valency of the oxygen atom and the 'N' atom participates in the formation of tautomer (Lactam and

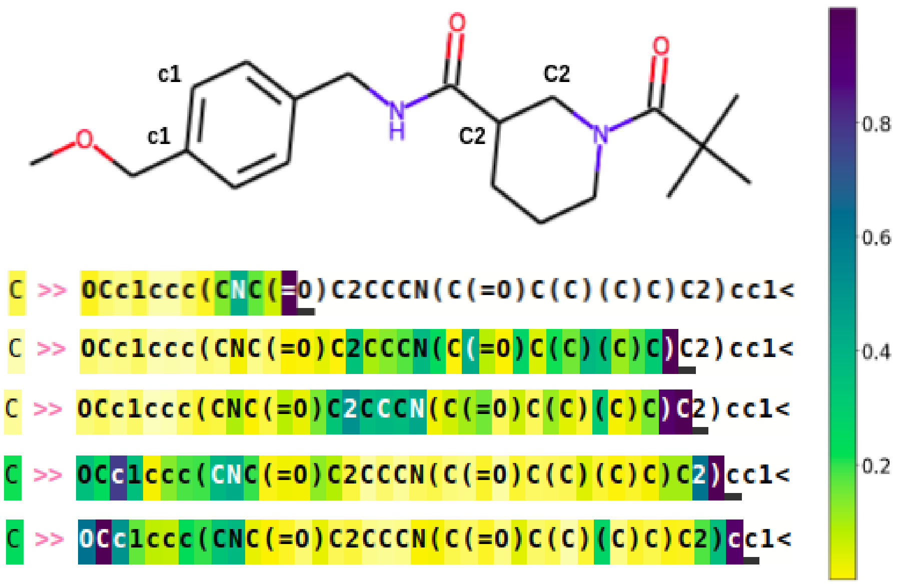
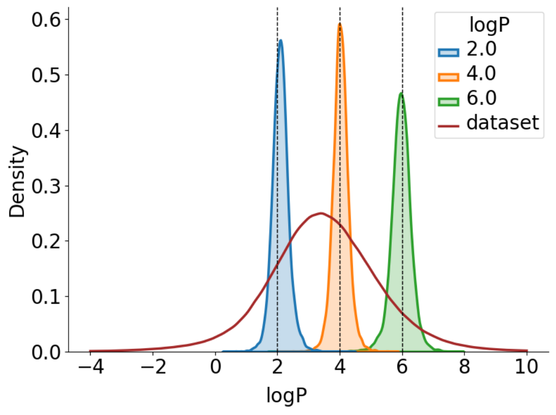

c1
c1
O
C2
C2
O
N H
N
C >> OCc1ccc(CNC(=O)C2CCCN(C(=O)C(C)(C)C)C2)cc1<
C >> OCc1ccc(CNC(=O)C2CCCN(C(=O)C(C)(C)C)C2)cc1<
C >> OCc1ccc(CNC(=O)C2CCCN(C(=O)C(C)(C)C)C2)cc1<
C >> OCc1ccc(CNC(=O)C2CCCN(C(=O)C(C)(C)C)C2)cc1<
C >> OCc1ccc(CNC(=O)C2CCCN(C(=O)C(C)(C)C)C2)cc1<
0.8
0.6
0.4
0.2

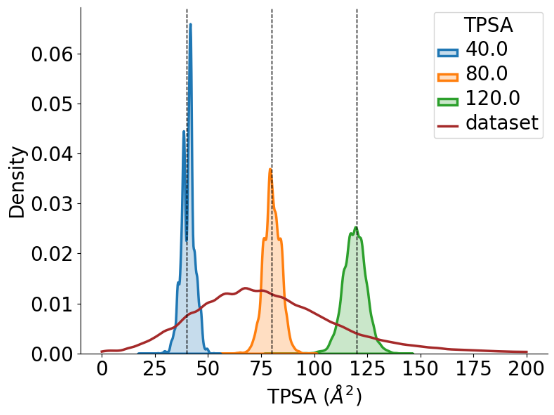
Figure 4: Input saliency maps for the shown generated molecule. The dark purple underline are the tokens under consideration for saliency maps. The intensity of color of each token indicates the importance of that token for generating the underlined token.

Lactim) which increases the stability of the structure. When generating the 'C' atom in the second saliency map, the model attends to '(', ')' to check if they are balanced, and also attends to the atoms in the non-aromatic ring. In the non-aromatic ring, it attends mostly to the immediate neighbors - '2' and 'N' atoms. When generating '2' token, it attends to the immediate previous 'C' token and the tokens in the non-aromatic ring. For the fourth and fifth saliency map, when generating 'c' tokens, the model rightly attends to the atoms in the aromatic ring since that ring is still incomplete. Thus, these saliency maps provide some insight on the chemical interpretability of the generative process.

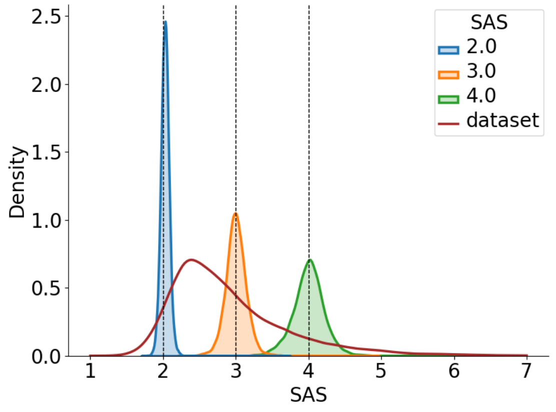
Further, to evaluate the robustness of the proposed framework, LigGPT's performance is compared with CharRNN in the low data regime. Here, we train both the models only on 10% of the MOSES training set and evaluate the metrics by generating 10,000 molecules. The results are reported in Table S1 of the Supporting Information. With temperature 0.9, LigGPT outperforms CharRNN on validity as well as novelty. With temperature 1.6, LigGPT has similar novelty as CharRNN but much better validity. Moreover, LigGPT has only 50% of the trainable parameters of CharRNN. This indicates greater efficiency of



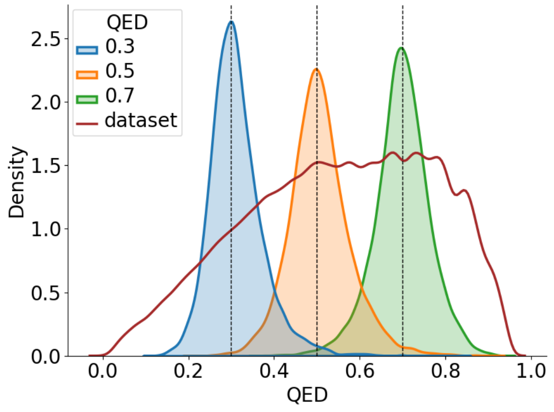
(a)



(b)



(c)



(d)

Figure 5: Distribution of property of generated molecules conditioned on: **(a)** logP **(b)** TPSA **(c)** SAS **(d)** QED . Distribution depicted using a solid red line corresponds to the whole dataset. Trained on GuacaMol dataset with Temperature=0.9.

LigGPT owing to its masked self-attention.

## Generation based on Single and Multiple Properties

Many processes in biology and chemistry require molecules to have certain property values in order to perform some functions. For example, for molecules to penetrate the blood–brain barrier (and thus act on receptors in the central nervous system), a TPSA value less than 90 $\AA^2$ is usually needed[51]. This motivates the need for models to have accurate conditional generation. So, the next objective is to evaluate the ability of LigGPT to generate molecules that exhibit specific properties (conditional generation). Since GuacaMol has a wider range in property values, we test the model's ability to control molecular properties trained on it. While only logP, SAS, TPSA and QED are used for property control, we would like to note that the model can be trained to learn any property that is inferred from the molecule's 2D structure. For each condition, 10,000 molecules are generated to evaluate property control.

Table 3: Comparison of different metrics while generating molecules conditioned on single property based on training on GuacaMol dataset. Temperature value of 0.9 was used.

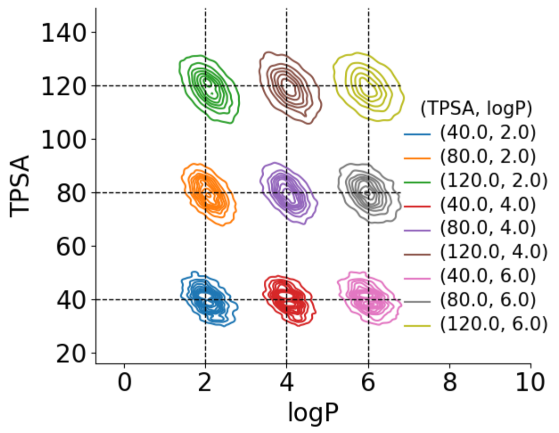
| Condition | Validity | Unique | Novelty | MAD |
|---|---|---|---|---|
| logP | 0.992 | 0.975 | 1.0 | 0.217 |
| TPSA | 0.992 | 0.966 | 1.0 | 3.339 |
| SAS | 0.993 | 0.965 | 1.0 | 0.108 |
| QED | 0.995 | 0.973 | 1.0 | 0.049 |

Distributions of molecular properties of LigGPT generated molecules while controlling a single property are depicted in Figure 5. The Mean Average Deviation (MAD), Validity, uniqueness and novelty values for each property are reported in Table 3. As seen in Figure 5, the distribution of properties are centered around the desired value. This is further exemplified by the low MAD scores (relative to the range of the property values) in the Table 3.

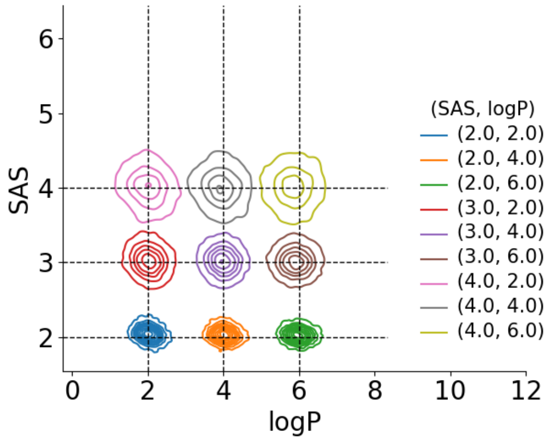
Table 4: Multi-property conditional training on GuacaMol dataset. Temperature 0.9 was used.

| Condition | Validity | Unique | Novelty | MAD_TPSA | MAD_logP | MAD_SAS |
|---|---|---|---|---|---|---|
| SAS+logP | 0.991 | 0.935 | 1.0 | - | 0.241 | 0.12 |
| SAS+TPSA | 0.992 | 0.929 | 1.0 | 3.543 | - | 0.133 |
| TPSA+logP | 0.989 | 0.942 | 1.0 | 3.528 | 0.227 | - |
| TPSA+logP+SAS | 0.99 | 0.874 | 1.0 | 3.629 | 0.254 | 0.158 |

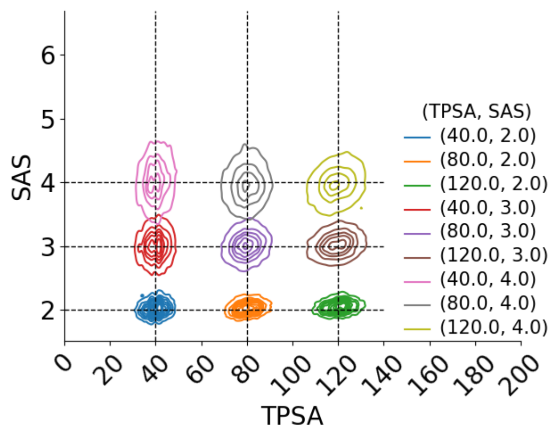
While generating molecules for specific purposes, in other words de novo design of molcules, it is necessary to optimize more than one property. For example, one may want molecules that have a specific values for logP and TPSA values. Hence, we check the model's capacity to control multiple properties simultaneously. For this, SAS, logP and TPSA are used. We evaluate the model's ability to generate desired distributions using two and three property controls at a time. Generated distribution of molecule properties is depicted in

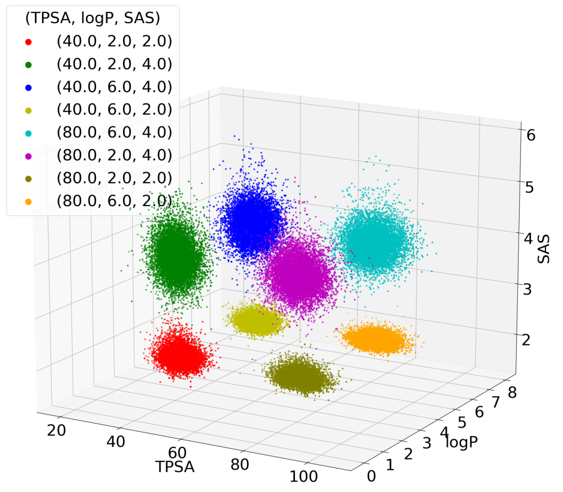


(a)



(b)





(c) (d)

Figure 6: Distribution of property of generated molecules conditioned on: **(a)** TPSA + logP **(b)** SAS + logP **(c)** SAS + TPSA **(d)** TPSA + logP + SAS. The values that are conditioned to are given in the Figures.

Figure 6. Well separated clusters centered at the desired property values are observed. As before, the low MAD values for each property combination, reported in Table 4, (as compared to the range of property values) indicate the strong control LigGPT has over multiple properties for accurate generation.

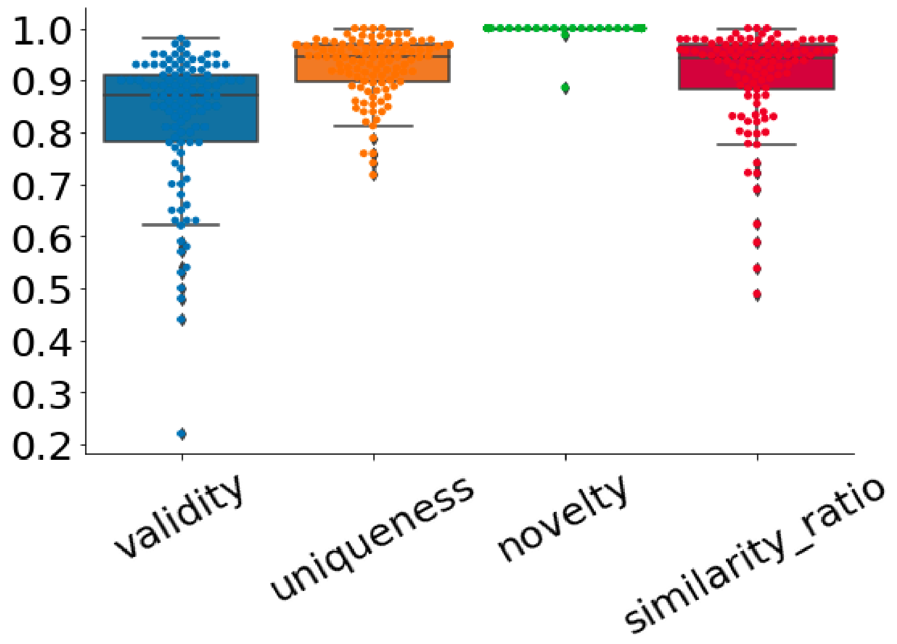


Figure 7: Boxplot of the evaluation metrics for the scaffold conditioned results

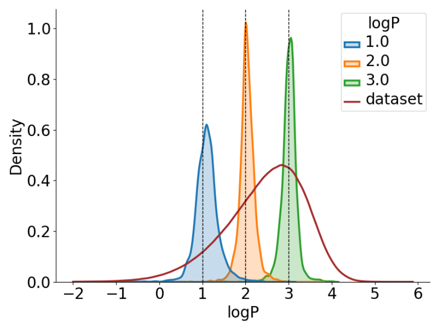
## Generation based on Scaffold

The above section demonstrated the ability of LigGPT to generate molecules with desired properties. In certain exercises, for eg. lead optimization, chemists intend to generate molecules containing a specific scaffold/skeleton and at the same time achieve desired property values. We evaluate the ability of LigGPT to generate structures with certain property values while maintaining the structure of the scaffold, results of which are presented in this and the next sections. We conduct these experiments on the MOSES benchmark dataset as it contains a set of test scaffolds that are non overlapping with the set of scaffolds that are present in the training set. We select a random set of 100 test scaffolds, then generate 100 molecules for each scaffold followed by calculation of validity, uniqueness, novelty and 'similarity ratio'. 'Similarity ratio' is defined as the fraction of valid generated molecules having Tanimoto similarity of the scaffold of the generated molecule and the conditioned scaffold greater than 0.8. The distribution of each of the metrics in terms of box plot, with the swarm plot overlaid on it, is shown in Figure 7. From the boxplot, it is seen that even after using high temperature of 1.6, around 75 of the scaffolds have validity greater than 0.8. By virtue of high temperature, almost all the scaffolds have uniqueness greater than 0.8. All the scaffolds have novelty greater than 0.8. Around 75 scaffolds have 'similarity ratio' greater than 0.9, which suggests most of the generated valid molecules have very similar scaffold to the scaffold used for condition. Some examples of generated molecules for two scaffolds are given in Figure S2 of the Supporting Information. In all the generated molecules, the conditioned scaffold strucutre is maintained.

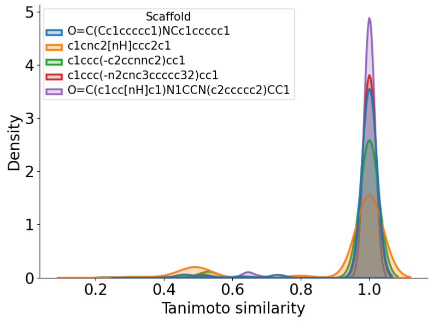
## Generation based on Scaffold and Property

We evaluate the models ability to generate structures containing desired scaffolds while controlling molecular properties. For our experiments, five scaffolds of different sizes were randomly chosen from the MOSES test set (Figure S1 of the Supporting Information). In these experiments, we define valid molecules as those molecular graphs that satisfy chemical valencies and contain scaffolds that have a Tanimoto similarity of at least 0.8 to the desired scaffold. The validity score of all scaffold based experiments are calculated based on this definition.

Generated distributions for single property control can be seen in Figure 8. Tanimoto similarity is calculated between the scaffold of the generated molecule and the conditional scaffold. Distribution of these Tanimoto similarity scores are also plotted in Figure 8. The distribution plots peak at 1 for all the scaffolds and properties. Since scaffold based generation is more constraining for property control, generated distributions are not as narrow and well separated as before. The quantitative results for single property control are reported in Table S2 in the Supporting Information. The low MAD scores still show that LigGPT deviates only slightly from intended values despite the constraints. QED is a function that is dependant on multiple molecular properties simultaneously. Therefore, QED is greatly influenced by the structure of the scaffold itself making it very hard to control under such constraints. We believe such competing objectives are the reason for large overlap between distributions generated for QED control. Figure S3 of the Supporting Information shows

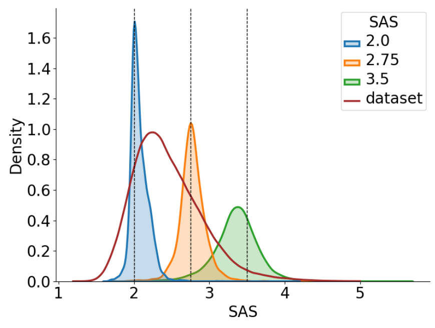


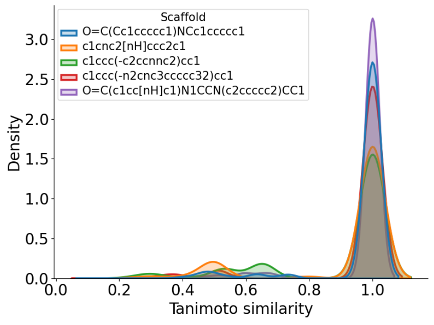
(a)



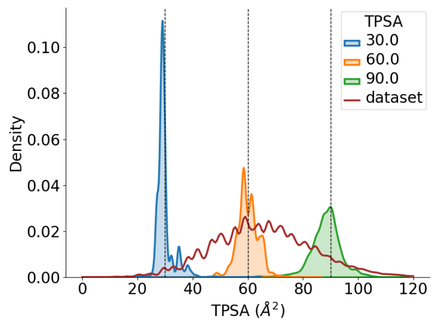
(b)

(c)

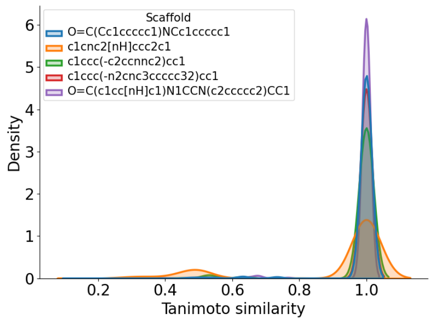




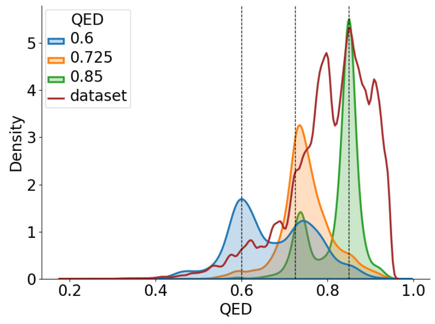
(d)



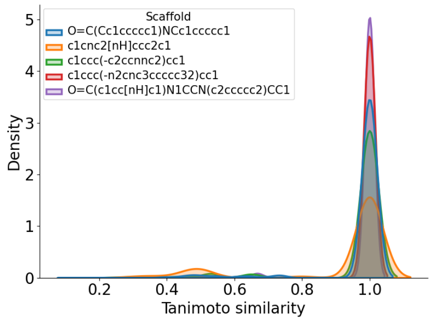
(e)



(f)

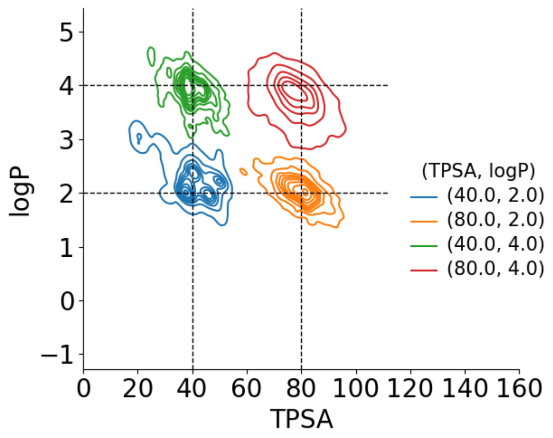


(g)

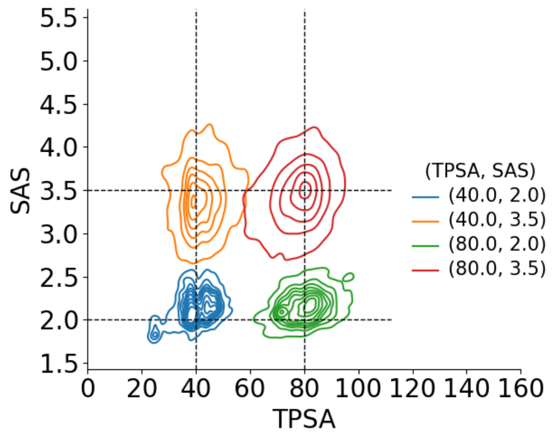


(h)

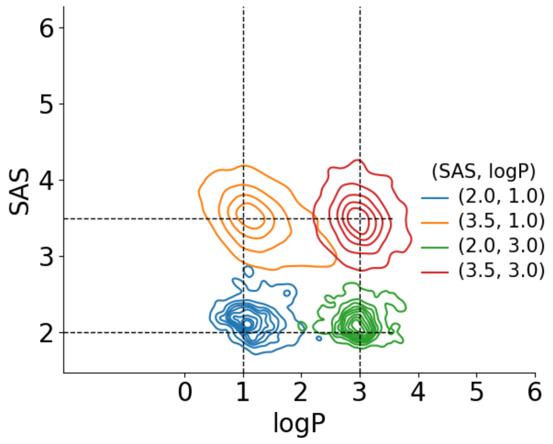
Figure 8: Distribution of property of generated molecules conditioned on: Scaffold + **(a)** logP **(c)** SAS **(e)** TPSA **(g)** QED . Distribution of tanimoto similarity of the scaffolds of the generated molecules and the scaffold used for condition for **(b)** logP **(d)** SAS **(f)** TPSA **(h)** QED. Trained on MOSES dataset. Temperature 1.6 used.



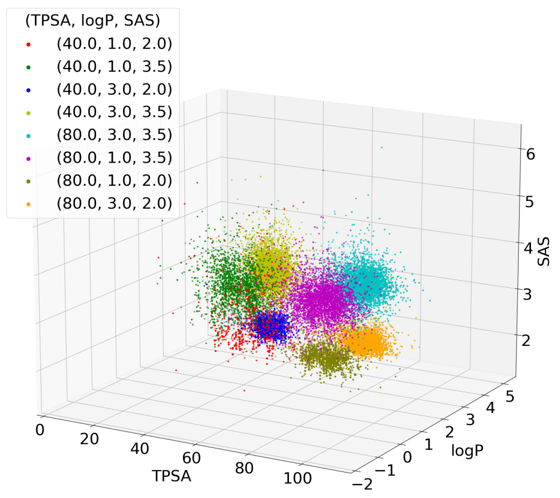
(a)



(b)



(c)



(d)

Figure 9: Distribution of property of generated molecules conditioned on: Scaffold + **(a)** TPSA + logP **(b)** SAS + TPSA **(c)** SAS + logP **(d)** TPSA + logP + SAS. Trained on MOSES dataset and temperature 1.6 used.

the molecules conditioned on scaffold + logP and scaffold + SAS. LigGPT adds different functional groups to the scaffold in order to get the desired property value. Multi-property control clusters are plotted in Figure 9. Even when using multiple properties, we see the Tanimoto similarity distributions peaking at 1 in Figure 10. Understandably, property-based clusters are not as well formed as before. However, there is a good separation between the clusters for two property control. The intended values of molecular properties are close to the

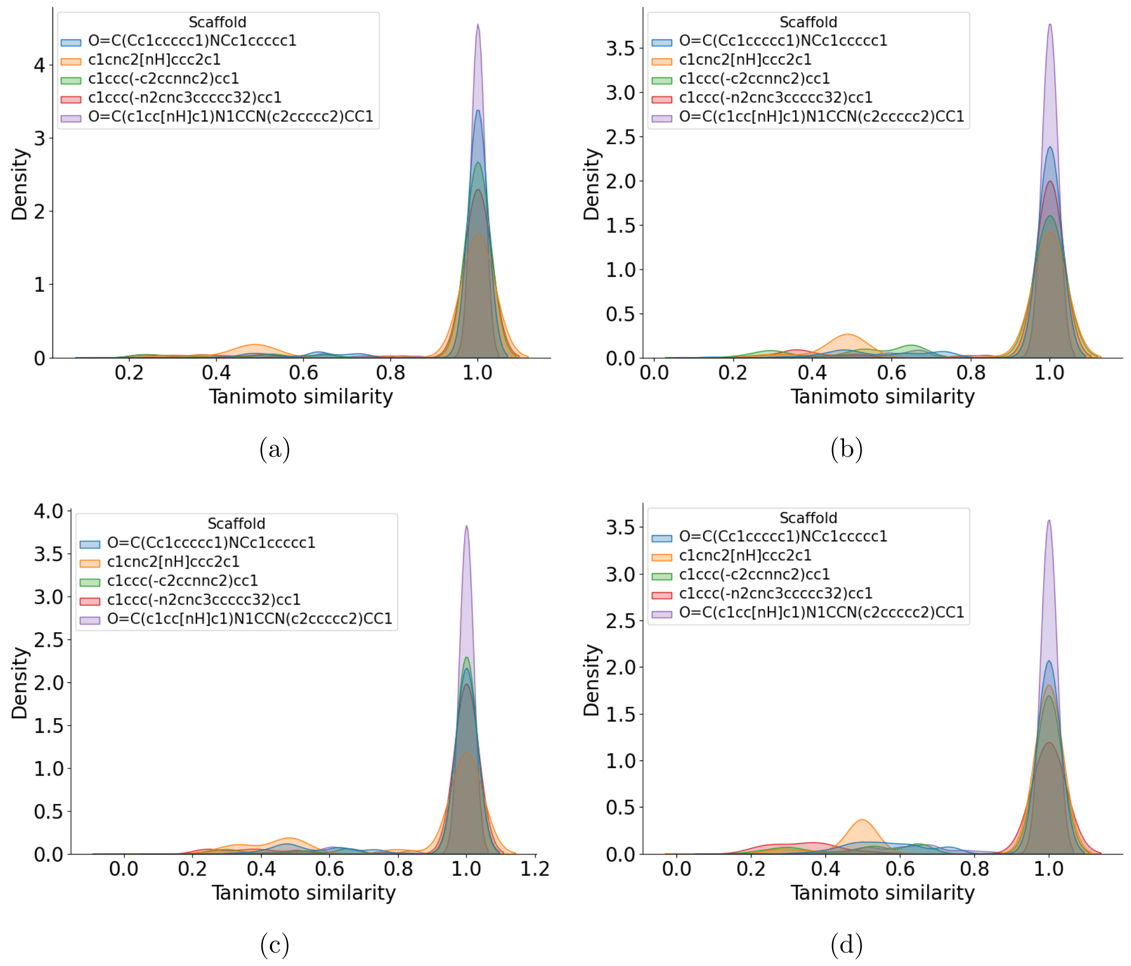


Figure 10: Distribution of tanimoto similarity of the scaffolds of the generated molecules and the scaffold used for condition for **(a)** TPSA + logP **(b)** SAS + TPSA **(c)** SAS + logP **(d)** TPSA + logP + SAS. Trained on MOSES dataset and temperature 1.6 used.

centers of these clusters. This can further be verified by results reported for multi-property control in Table S3 in the Supporting Information. For three property control one of the clusters (red) is not well formed due to highly constraining property values. We see that the rest of the clusters are largely well formed and separated.

## One Shot Lead Optimization

Due to LigGPT's strong ability to control molecular properties, we believe it has potential usage in real world problems such as lead optimization. We call this 'one shot optimization' as it's a one step process of providing the desired scaffold and properties for optimized

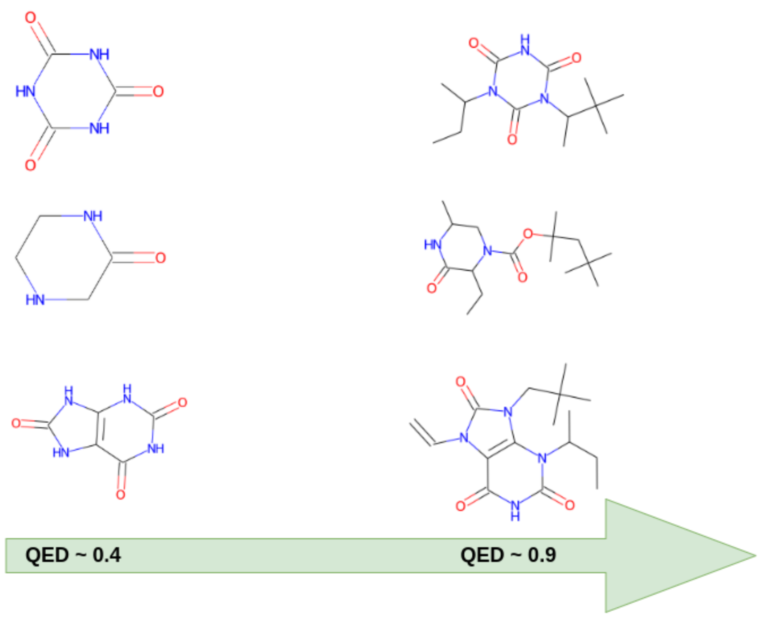


Figure 11: One shot optimization of QED value conditioned on the scaffold

molecule generation. To demonstrate this, three scaffolds from test set having QED around 0.4 are sampled. Using these scaffolds and QED 0.9 as the condition, we generate molecules using LigGPT. Sample generated molecules are shown in Figure 11. The scaffold structure is maintained in the generated molecules and their QED values are around 0.9. While the usefulness is demonstrated here only on QED, we would like to note that it could be adopted for other molecular properties as well. Furthermore, it could also be used in other molecular design task such generation of active molecules with good docking scores for a particular protein, provided, a dataset for the protein is available for training beforehand.

## Conclusion

In this work, we designed a Transformer-Decoder model called LigGPT for molecular generation. This model utilises masked self-attention mechanisms that make it simpler to learn long range dependencies between string tokens. This is especially useful to learn the semantics of valid SMILES strings that satisfies valencies and ring closures. We see through our benchmarking experiments that LigGPT shows very good validity scores even with sampling temperature as high as 1.6 for the MOSES dataset and 0.9 for the GuacaMol dataset. Furthermore, as shown, this also allows the model to do well in low data regimes. The high sampling temperatures enables the model to generate large amounts of novel and unique molecules. Therefore, LigGPT is able to show good performance on both datasets with it outperforming all other methods benchmarked on the GuacaMol dataset.

We also show that the model learns higher level chemical representations through molecular property control. LigGPT is able to generate molecules with property values that deviate only slightly from the exact values that are passed by the user. It's also able to generate molecules containing user specified scaffolds while controlling these properties. It does this with good accuracy despite the constraining conditions of scaffold based drug design. Through this, we convey LigGPT's real world utility for molecular generation. Consequently, we believe that the LigGPT model should be considered a strong architecture to be used by itself or incorporated into other molecular generation techniques.

## Acknowledgement

The authors thank Manasa Kondamadugu, Yashaswi Pathak, Sarvesh Mehta and Manan Goel for their comments during the preparation of the manuscript. We thank IHub-Data, IIIT Hyderabad for financial support.

## Supporting Information Available

Supporting Information contains results of training on 10,000 molecules and results of scaffold and property conditioning. Furthermore figures of scaffolds, generated molecules from scaffold conditioning as well as scaffold and property conditioning experiments are provided.

## References

(1) Polishchuk, P. G.; Madzhidov, T. I.; Varnek, A. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of computer-aided molecular design* **2013**, *27*, 675–679.

(2) Kim, S.; Thiessen, P. A.; Bolton, E. E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B. A., et al. PubChem substance and compound databases. *Nucleic acids research* **2016**, *44*, D1202–D1213.

(3) Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems* **2014**, *27*, 2672–2680.

(4) Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2019; pp 4401–4410.

(5) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. Advances in neural information processing systems. 2017; pp 5998–6008.

(6) Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* **2018**,

(7) Chen, H.; Engkvist, O.; Wang, Y.; Olivecrona, M.; Blaschke, T. The rise of deep learning in drug discovery. *Drug discovery today* **2018**, *23*, 1241–1250.

(8) Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **2018**, *361*, 360–365.

(9) Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling* **2019**, *59*, 1096–1108.

(10) Polykovskiy, D.; Zhebrak, A.; Sanchez-Lengeling, B.; Golovanov, S.; Tatanov, O.; Belyaev, S.; Kurbanov, R.; Artamonov, A.; Aladinskiy, V.; Veselov, M., et al. Molecular sets (moses): A benchmarking platform for molecular generation models. *Frontiers in pharmacology* **2020**, *11*.

(11) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28*, 31–36.

(12) Pathak, Y.; Laghuvarapu, S.; Mehta, S.; Priyakumar, U. D. Chemically interpretable graph interaction network for prediction of pharmacokinetic properties of drug-like molecules. Proceedings of the AAAI Conference on Artificial Intelligence. 2020; pp 873–880.

(13) Segler, M. H.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science* **2018**, *4*, 120–131.

(14) Gupta, A.; Müller, A. T.; Huisman, B. J.; Fuchs, J. A.; Schneider, P.; Schneider, G. Generative recurrent networks for de novo drug design. *Molecular informatics* **2018**, *37*, 1700111.

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

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

(17) Liu, Q.; Allamanis, M.; Brockschmidt, M.; Gaunt, A. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems* **2018**, *31*, 7795–7804.

(18) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925* **2017**,

(19) Simonovsky, M.; Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. International Conference on Artificial Neural Networks. 2018; pp 412–422.

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

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

(22) Kadurin, A.; Nikolenko, S.; Khrabrov, K.; Aliper, A.; Zhavoronkov, A. druGAN: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics* **2017**, *14*, 3098–3104.

(23) Putin, E.; Asadulaev, A.; Vanhaelen, Q.; Ivanenkov, Y.; Aladinskaya, A. V.; Aliper, A.; Zhavoronkov, A. Adversarial threshold neural computer for molecular de novo design. *Molecular pharmaceutics* **2018**, *15*, 4386–4397.

(24) Polykovskiy, D.; Zhebrak, A.; Vetrov, D.; Ivanenkov, Y.; Aladinskiy, V.; Mamoshina, P.; Bozdaganyan, M.; Aliper, A.; Zhavoronkov, A.; Kadurin, A. Entangled conditional adversarial autoencoder for de novo drug discovery. *Molecular pharmaceutics* **2018**, *15*, 4398–4405.

(25) Hong, S. H.; Ryu, S.; Lim, J.; Kim, W. Y. Molecular Generative Model Based on an Adversarially Regularized Autoencoder. *Journal of Chemical Information and Modeling* **2019**, *60*, 29–36.

(26) Arús-Pous, J.; Johansson, S. V.; Prykhodko, O.; Bjerrum, E. J.; Tyrchan, C.; Reymond, J.-L.; Chen, H.; Engkvist, O. Randomized SMILES strings improve the quality of molecular generative models. *Journal of cheminformatics* **2019**, *11*, 1–13.

(27) Bjerrum, E. J. SMILES enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076* **2017**,

(28) Bjerrum, E. J.; Sattarov, B. Improving chemical autoencoder latent space and molecular de novo generation diversity with heteroencoders. *Biomolecules* **2018**, *8*, 131.

(29) Pathak, Y.; Juneja, K. S.; Varma, G.; Ehara, M.; Priyakumar, U. D. Deep learning enabled inorganic material generator. *Physical Chemistry Chemical Physics* **2020**, *22*, 26935–26943.

(30) Prykhodko, O.; Johansson, S. V.; Kotsias, P.-C.; Arús-Pous, J.; Bjerrum, E. J.; Engkvist, O.; Chen, H. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics* **2019**, *11*, 74.

(31) Guimaraes, G. L.; Sanchez-Lengeling, B.; Outeiral, C.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *arXiv preprint arXiv:1705.10843* **2017**,

(32) Sanchez-Lengeling, B.; Outeiral, C.; Guimaraes, G. L.; Aspuru-Guzik, A. Optimizing distributions over molecular space. An objective-reinforced generative adversarial network for inverse-design chemistry (ORGANIC). **2017**,

(33) De Cao, N.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973* **2018**,

(34) Putin, E.; Asadulaev, A.; Ivanenkov, Y.; Aladinskiy, V.; Sanchez-Lengeling, B.; Aspuru-Guzik, A.; Zhavoronkov, A. Reinforced adversarial neural computer for de novo molecular design. *Journal of chemical information and modeling* **2018**, *58*, 1194–1204.

(35) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* **2018**, *4*, 268–276.

(36) Mehta, S.; Laghuvarapu, S.; Pathak, Y.; Sethi, A.; Alvala, M.; Priyakumar, U. D. Enhanced Sampling of Chemical Space for High Throughput Screening Applications using Machine Learning.

(37) Winter, R.; Montanari, F.; Steffen, A.; Briem, H.; Noé, F.; Clevert, D.-A. Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science* **2019**, *10*, 8016–8024.

(38) Kim, K.; Kang, S.; Yoo, J.; Kwon, Y.; Nam, Y.; Lee, D.; Kim, I.; Choi, Y.-S.; Jung, Y.; Kim, S., et al. Deep-learning-based inverse design model for intelligent discovery of organic molecules. *npj Computational Materials* **2018**, *4*, 1–7.

(39) Maziarka, Ł.; Pocha, A.; Kaczmarczyk, J.; Rataj, K.; Danel, T.; Warchoł, M. Mol-CycleGAN: a generative model for molecular optimization. *Journal of Cheminformatics* **2020**, *12*, 1–18.

(40) Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. Proceedings of the IEEE international conference on computer vision. 2017; pp 2223–2232.

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

(42) Arús-Pous, J.; Patronov, A.; Bjerrum, E. J.; Tyrchan, C.; Reymond, J.-L.; Chen, H.; Engkvist, O. SMILES-based deep generative scaffold decorator for de-novo drug design. *Journal of Cheminformatics* **2020**, *12*, 1–18.

(43) Lim, J.; Hwang, S.-Y.; Kim, S.; Moon, S.; Kim, W. Y. Scaffold-based molecular design using graph generative model. *arXiv preprint arXiv:1905.13639* **2019**,

(44) Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving language understanding by generative pre-training.

(45) Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.

(46) Gaulton, A.; Hersey, A.; Nowotka, M.; Bento, A. P.; Chambers, J.; Mendez, D.; Mutowo, P.; Atkinson, F.; Bellis, L. J.; Cibrián-Uhalte, E., et al. The ChEMBL database in 2017. *Nucleic acids research* **2017**, *45*, D945–D954.

(47) Landrum, G. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling. 2013.

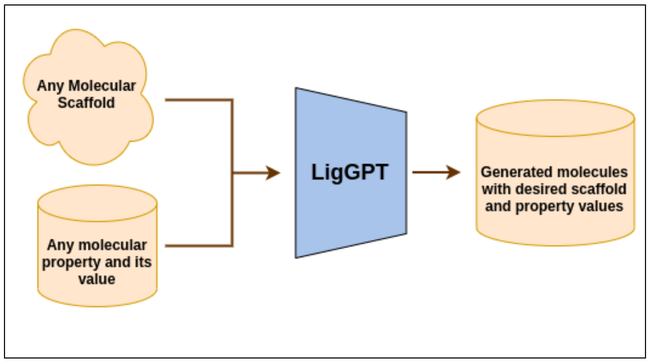
(48) Bemis, G. W.; Murcko, M. A. The properties of known drugs. 1. Molecular frameworks. *Journal of medicinal chemistry* **1996**, *39*, 2887–2893.

(49) Ertl, P.; Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics* **2009**, *1*, 8.

(50) Bickerton, G. R.; Paolini, G. V.; Besnard, J.; Muresan, S.; Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry* **2012**, *4*, 90–98.

(51) Clark, D. E. Rapid calculation of polar molecular surface area and its application to the prediction of transport phenomena. 1. Prediction of intestinal absorption. *Journal of Phatmaceutical Sciences* **1999**,

## Graphical TOC Entry


Any Molecular Scaffold
Any molecular property and its value
LigGPT
Generated molecules with desired scaffold and property values