LigGPT: Molecular Generation using a Transformer-Decoder Model

Viraj Bagal,[†] Rishal Aggarwal,[‡] and Deva Priyakumar[*,‡]

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

[‡]*Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad 500 032, India*

E-mail: deva@iiit.ac.in

Abstract

The usage of deep learning techniques for molecular generation has been gaining traction for 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 drug discovery. Inspired from Generative Pre-Training (GPT) models, in this study, we train a Transformer-Decoder on the next token prediction task using masked self-attention for the generation of molecules. We show that our model has the best performance on the GuacaMol dataset and comparable performance on the Moses dataset in generating valid, unique and novel molecules when benchmarked against other modern methods for molecular generation. Furthermore, we demonstrate that the model can be trained conditionally to control multiple properties of the generated molecules. As a potential real world application, we show that the model can also be used to generate molecules with desired scaffolds in addition to the desired properties, by passing these structures as conditions. Using saliency maps, we also highlight the interpretability of the generative process of the model. Code is available at `https://github.com/VirajBagal/LigGPT`

Introduction

It has been postulated that the total number of potential drug like candidates can range from 10^{23} to 10^{60} molecules.[1] However, only about 10^{8} molecules have been synthesized.[2] This disparity between synthesized and potential molecules beckons for the use of generative models that can model the distribution of molecules for efficient sampling.

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] Therefore, 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] Thus, such methods are preferred over classical methods of de novo drug design that are heavily dependant on predefined drug molecules.[10] The rapid adoption of deep generative model has also led to the development of benchmark datasets such as the Molecular Sets (MOSES)[11] and GuacaMol[9] datasets.

The representation of molecules in Simplified Molecular Input Line Entry System (SMILES)[12] notation as a string of characters enables the usage of modern NLP deep learning models for their computation. One such model is the Transformer architecture.[5] Transformers use self and masked attention to efficiently gain context from all previous input and output tokens for its predictions. Thus, it has shown the state of the art performance in language translation tasks. Transformers consist of both 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 model (GPT).[13] In this work we train a smaller version of the GPT model to predict the next token for molecular generation. We call this model LigGPT. 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.

While working with this method we provide the following contributions:

- To the best of our knowledge, this is the first work that has used the GPT architecture for molecular generation.
- We also show that it is the preferred method when using the GuacaMol dataset.We also show that our model performs on par with other methods when benchmarked on the Moses dataset.
- We conduct multiple experiments to show that our model has strong control over molecular properties and is able to control the value of multiple properties simultaneously during generation.
- We also show that the model can be trained to generate molecules containing user specified core structures/scaffolds while controlling multiple properties.

Previous Works

The earliest deep learning architectures for molecular generation involved the usage of Recurrent Neural Networks (RNNs) on molecular SMILES.[14,15] Such models have also previously been trained on large corpus of molecules and then focused through the usage of reinforcement learning[16,17] or transfer learning[14] to generate molecules of desirable properties and activity.

Auto-encoder variants such as the Variational Auto-Encoder(VAE)[18–22] and Adversarial Auto-Encoder(AAE)[23–26] 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.[21] This is because similar molecules can have very different SMILES representations. To counteract this problem, SMILES strings have also been randomized as inputs for such models.[27–29] Junction Tree VAE (JTVAE)[21] is also an alternative solution which represents molecules as graph tree structures. JTVAE 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.

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. These 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] was a method that 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 pretrained auto-encoder. This ensures that the model can work with latent representations and doesn't have to worry about SMILES syntax. Most of these methods have been benchmarked on the MOSES and GuacaMol datasets for easy comparison.

Often, methods use bayesian optimization,[35] reinforcement learning[16,34] or other optimization methods[36] to generate molecules of desirable properties. However, only few methods have been designed, that explicitly define the values of properties for generated molecules. Conditional RNNs[37] and Conditional Adversarially Regularized Autoencoder (CARAE)[26] are two such methods that sample molecules based on exact values. RNNs have also been previously used to generate molecules based on given scaffolds.[38] However, only a graph based method has been designed that ensures the presence of desired scaffolds while gener-

ating molecules with exact property values.[39] In this work, we show that transformers control molecular properties and scaffolds with extremely high accuracy, leading to our conclusion that they learn a good representation of the chemical space.

Methods

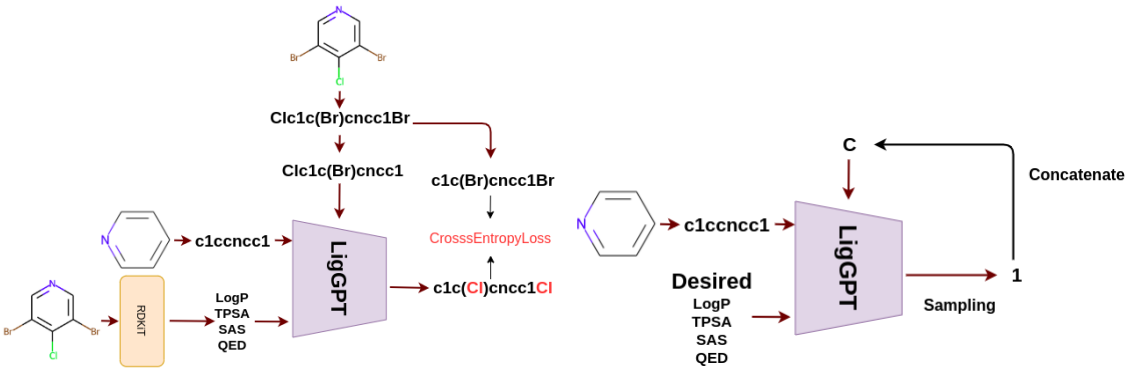
Datasets

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 clean 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[40] 24 and contains 1.6 Million molecules. To calculate molecular properties and to extract Bemis-Murcko scaffolds,[41] we used the Rdkit toolkit.[42] Molecular properties in the GuacaMol dataset are less restricted as compared to the MOSES dataset, as can be seen in Figure3. Therefore, we use the GuacaMol dataset to test property conditional generation. However, MOSES also provides a test set of scaffolds which we use to evaluate scaffold and property conditional generation.

Model Overview

The model schematic for training and generation is given in Figure 1. 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 2. The model is essentially a mini version of the Generative Pre-Training (GPT) model. Unlike GPT1 that has around 110M parameters, LigGPT has only around 6M parameters. LigGPT comprises stacked decoder blocks. Each decoder block is composed of a masked self-attention layer and feed forward layer. LigGPT



(a) Training (b) Generation

Figure 1: Schematic for (a)Training and (b)Generation using the LigGPT model.

consists of 8 such decoder blocks. SMILES strings are first tokenized. Further, to keep track of the position of each token in the sequence, position tokens are assigned to each position. During conditional training, segment tokens are provided to distinguish between the condition and the molecule representation. All the tokens are mapped to the same space using respective Embedding layers. All the embeddings are added and passed as input to the model.

GPT architectures work on a masked self-attention mechanism. Self-attention is calculated through 'Scaled Dot Product Attention' in the same manner as.[5] 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. Concretely, attention can be represented by the following formula:

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

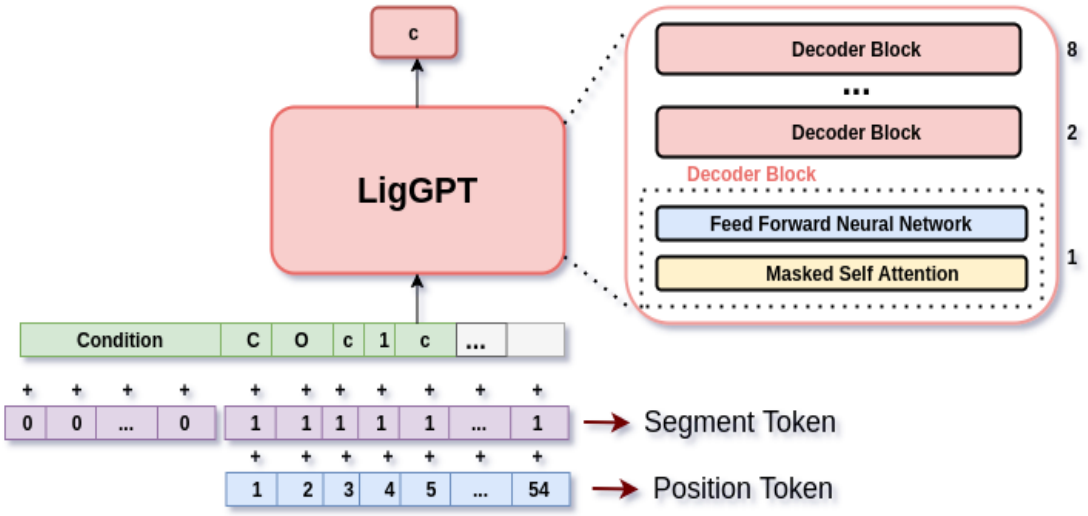


Figure 2: LigGPT consists of 8 Decoder blocks. Each decoder block is a stack of Masked Self-Attention and Feed forward layer. The network is trained on next token prediction task. For conditional training, the appropriate condition is transformed using linear layer and concatenated to the embedding of the SMILES representation of the molecule. Segment tokens allow the model to distinguish between the 'Condition' and the 'SMILES sequence'. Position tokens provide the information of the position of each SMILES token in the sequence. The embeddings of the SMILES token, segment token and position token are added and passed as input to the model.

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

The essential difference between masked self-attention and self-attention is that, masked self-attention masks tokens that occur after the current time step. This is essential as 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 masked self-attention 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

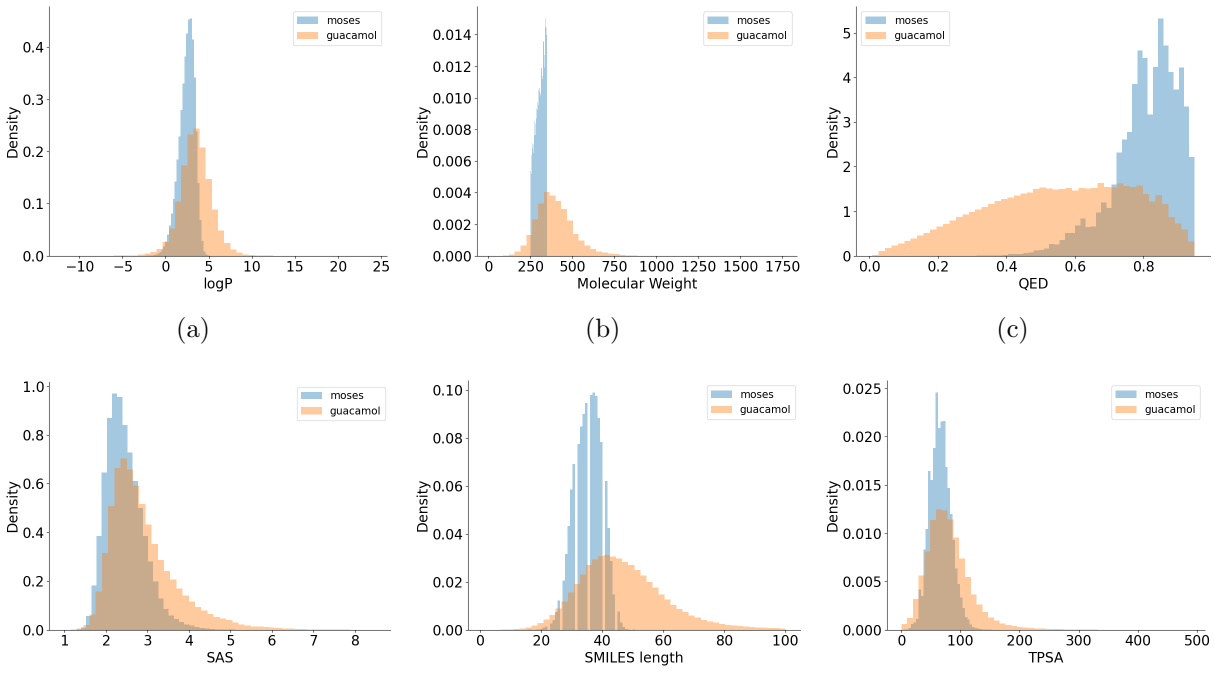


Figure 3: Distribution of property of molecules of Moses and Guacamol datasets.

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. During generation a start token of a single carbon atom is provided to the network along with the conditions.

Results and Discussion

We trained and tested LigGPT on both the MOSES and GuacaMol datasets. The models were trained on a NVIDIA 2080Ti GPU and took about 10 mins per epoch. The models converged and showed best performance after 10 epochs. However, we noticed that training it for slightly fewer epochs led to similar results in validity, novelty and uniqueness of generated

molecules.

Before explaining the experimental results, we describe the metrics used to evaluate the models:

- **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 mode collapse.
- **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 using Tanimoto similarity of each pair of molecules in the generated set (S).

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

Unconditional Generation

Table 1: Unconditional training on MOSES dataset. Temperature 1.6 was used.

Models	Validity	Unique@10K	Novelty	IntDiv_1	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**

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

Table 2: Unconditional training on GuacaMol dataset. Temperature 0.9 was used.

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**

SMILES. To get the optimal model for each dataset we check the generative performance for several sampling temperature values between 0.7 and 1.6. We notice that the model performs best at a temperature of 1.6 for MOSES and 0.9 for GuacaMol. We report the optimal model performance on each dataset 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 at 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. 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. We observe that LigGPT's validity and uniqueness is slightly higher than LatentGAN, but LatentGAN's novelty is greater by 0.008. On the GuacaMol benchmark, we see that 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 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. Moreover, even though Guacamol dataset has larger molecules as compared to Moses dataset, LigGPT generates molecules with very high validity. This indicates that LigGPT handles long range dependencies very well.

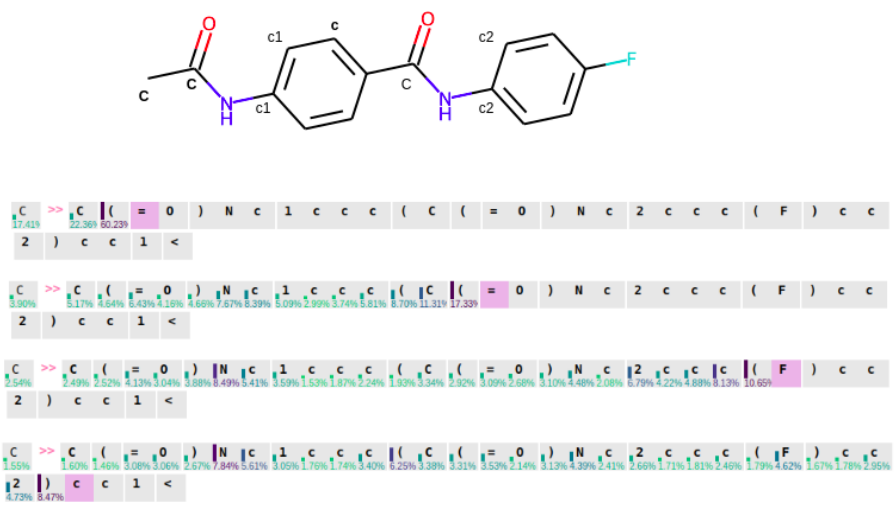


Figure 4: Input saliency maps for the shown generated molecule. The pink boxes are the tokens under consideration for saliency maps. The score of each token indicates the importance of that token for generating the token in pink box.

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 double bonds in the first two saliency maps, the model rightly attends mostly to the immediate previous 'C' and '(' tokens. These tokens refer to non-aromatic carbon and branching from the chain respectively and these tokens allow the insertion of double bond. When generating 'F' in the third saliency map, the model attends to the immediate previous aromatic carbon 'c', branching from chain '(' and 'N' tokens. Branching from aromatic carbon and possibility of the hydrogen bond N-H–F allows the insertion of 'F' . When generating 'c' in the fourth saliency map, high scores are assigned to '(', ')', farthest 'N' and the 'c' next to 'N'. All these tokens are in the neighbourhood of the generated 'c' token. '(' and ')' indicate the completeness of the side-chain and thus allows the model to focus on completing the aromcatic ring. This enables the insertion of 'c' token. Thus, these saliency maps provide some insight on the interpretability of the generative process.

Table 3: Comparison of CharRNN and LigGPT on the generation of 10,000 molecules by training only on 10% of the MOSES train split. CharRNN and LigGPT have 11.9 M and 6.3 M trainable parameters respectively.

Model	Validity	Unique	Novelty	Temperature
CharRNN	0.961	1.0	0.888	0.9
LigGPT	0.983	1.0	0.903	0.9
CharRNN	0.581	1.0	0.987	1.6
LigGPT	0.707	1.0	0.985	1.6

Further, we compare the performance of LigGPT 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 3. 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 LigGPT owing to its masked self-attention.

Conditional Generation

Since GuacaMol has a larger range in property values, we test the model's ability to control molecular properties on it. While we use only logP, SAS,[43] TPSA and QED[44] 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 we generate 10,000 molecules to evaluate property control.

The explanation of the properties is given below:

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

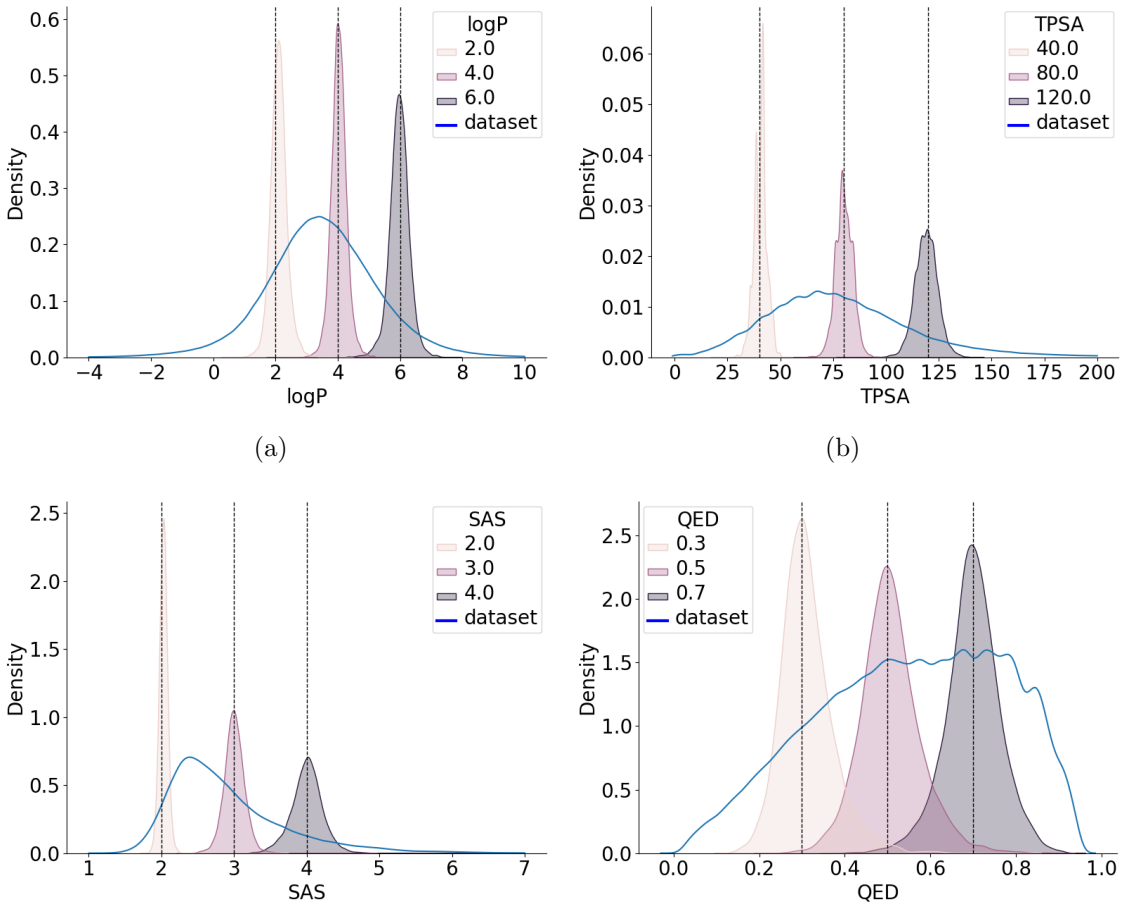


Figure 5: Distribution of property of generated molecules conditioned on: **(a)** logP **(b)** TPSA **(c)** SAS **(d)** QED . Trained on GuacaMol dataset. Temperature 0.9 used.

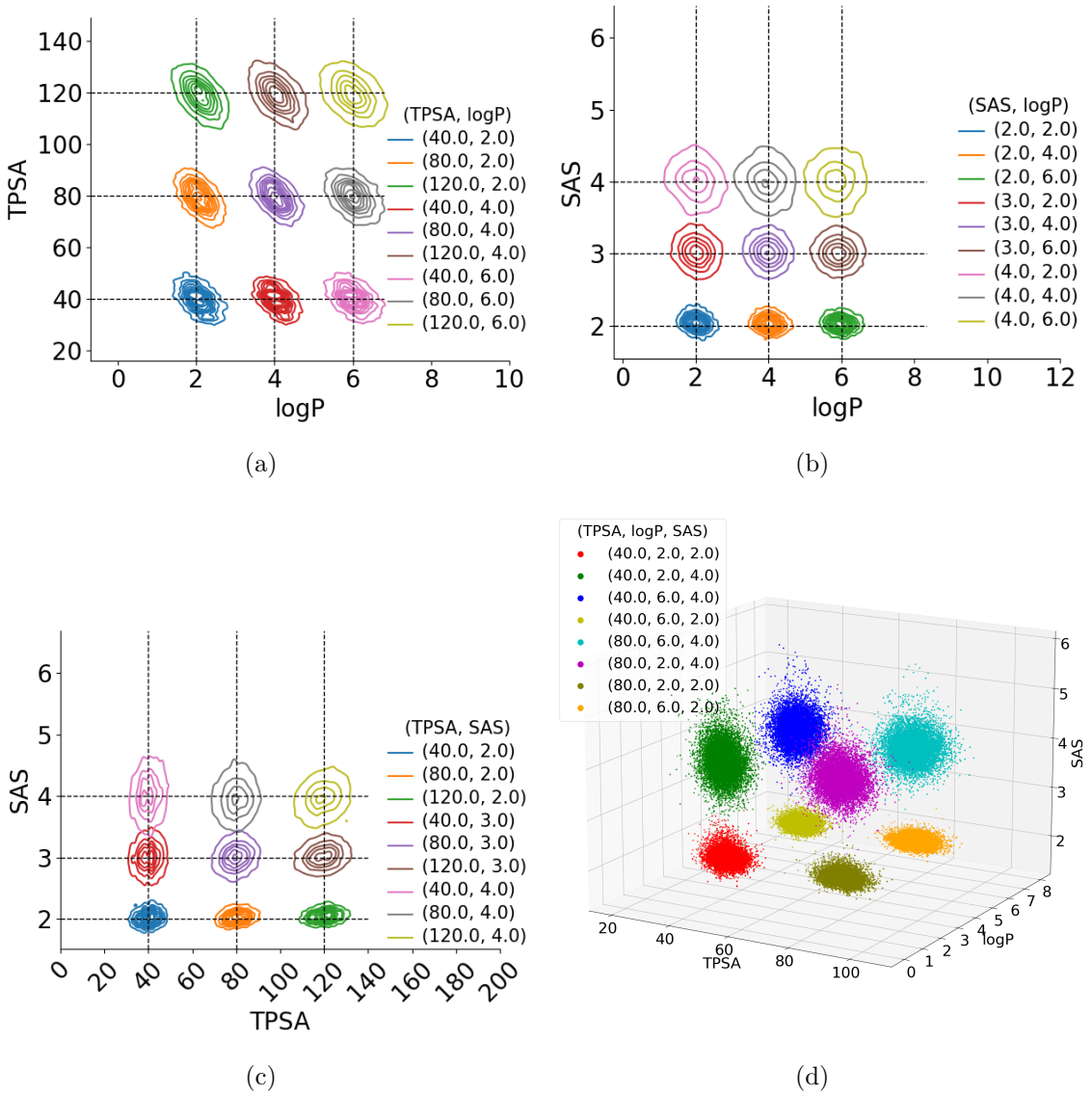
- **Topological Polar Surface Area (TPSA)**: the surface sum over all polar atoms. It measures the drug's ability to permeate cell membranes. Molecules with TPSA greater than 140 $Å^2$ tend to be poor at permeating cell membranes.

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

Generated distributions of molecular properties for controlling a single property are visualized in Figure 5. The average values for each property are reported in Table 4. As seen in Figure 5, the properties of generated molecules deviate only slightly from the intended

Table 4: Single property conditional training on GuacaMol dataset.Temperature 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



(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

values. This is further exemplified by the low MAD scores in the Table 4.

Next we check the model's capacity to control multiple properties simultaneously. For

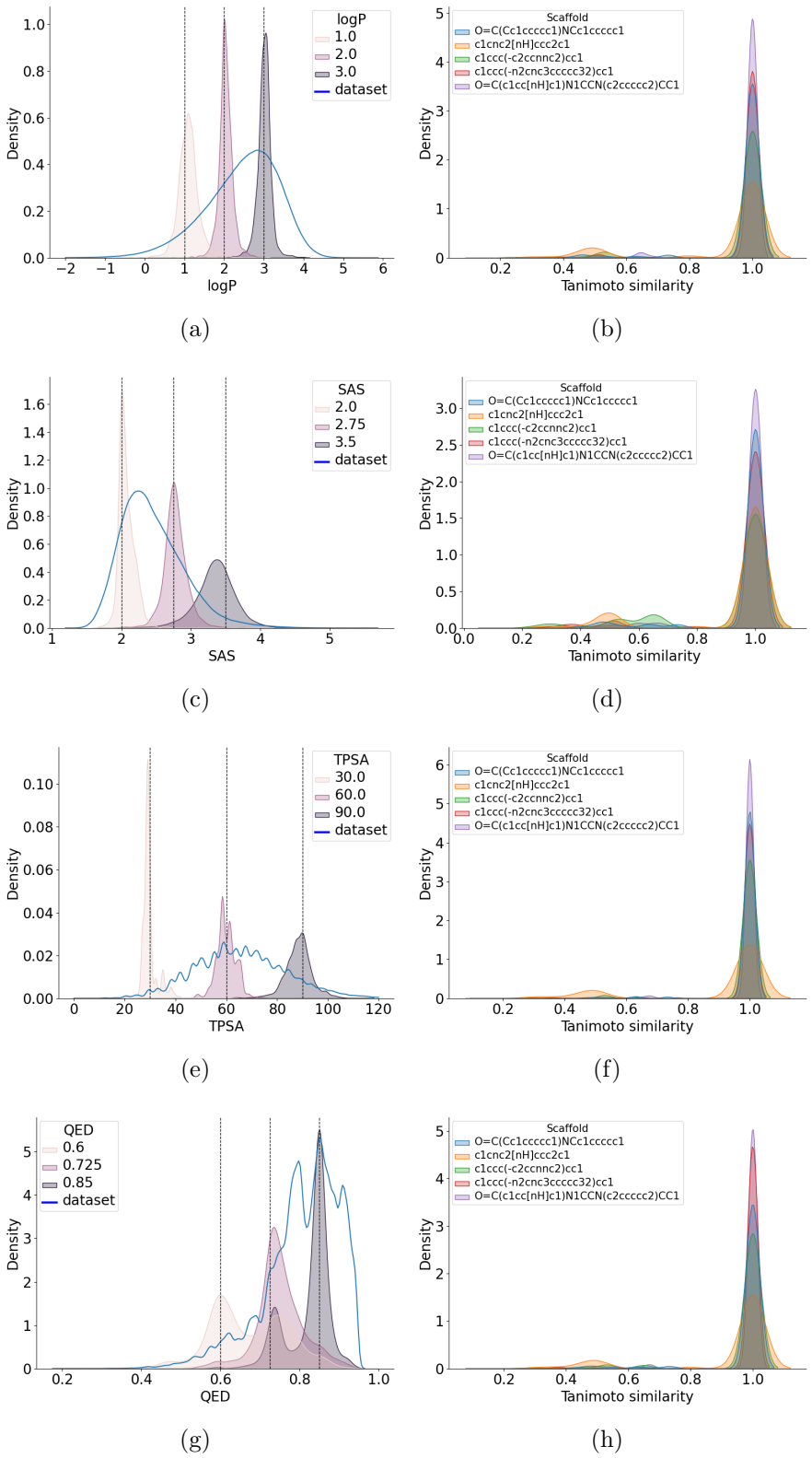


Figure 7: Distribution of property of generated molecules conditioned on: Scaffold + **(a)** logP **(c)** SAS **(e)** TPSA **(g)** QED . Distribution of tanimoto similarity of the scaffolds of the generated molecules and the scaffold used for condition for **(b)** logP **(d)** SAS **(f)** TPSA **(h)** QED. Trained on MOSES dataset. Temperature 1.6 used.

Table 5: 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

this, we use SAS, LogP and TPSA. 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 visualised in Figure 6. We see well separated clusters centered at the desired property values. As before, we also report the average MAD results for each property combination in Table 5. The low MAD results indicate the strong control LigGPT has over multiple properties for accurate generation.

Scaffold Based Methods

We evaluate the models ability to generate structures containing desired scaffolds while controlling molecular properties. 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. For our experiments we randomly chose 5 scaffolds of different sizes from the MOSES test set. 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 7. 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 7. 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 6: Scaffold + Single property (logP, TPSA) conditional training on MOSES dataset. Temperature 1.6 was used. Metric calculated only for molecules having tanimoto similarity of the scaffold of the generated molecule and the scaffold used for condition greater than 0.8. **(a)** O=C(Cc1ccccc1)NCc1ccccc1 **(b)** c1cnc2[nH]ccc2c1 **(c)** c1ccc(-c2ccnnc2)cc1 **(d)** c1ccc(-n2cnc3ccccc32)cc1 **(e)** O=C(c1cc[nH]c1)N1CCN(c2ccccc2)CC1

Cond	Validity	Unique	Novelty	MAD	Cond	Validity	Unique	Novelty	MAD
(a)+logP	0.893	0.812	1.0	0.145	(a)+TPSA	0.906	0.870	1.0	2.303
(b)+logP	0.712	0.975	1.0	0.151	(b)+TPSA	0.692	0.961	1.0	3.239
(c)+logP	0.826	0.922	1.0	0.146	(c)+TPSA	0.894	0.874	1.0	2.439
(d)+logP	0.891	0.858	1.0	0.160	(d)+TPSA	0.902	0.891	1.0	3.178
(e)+logP	0.898	0.461	1.0	0.125	(e)+TPSA	0.882	0.431	1.0	3.986
(a)+SAS	0.812	0.934	1.0	0.124	(a)+QED	0.872	0.951	1.0	0.05
(b)+SAS	0.726	0.775	1.0	0.174	(b)+QED	0.702	0.98	1.0	0.052
(c)+SAS	0.698	0.862	1.0	0.167	(c)+QED	0.849	0.947	1.0	0.045
(d)+SAS	0.823	0.910	1.0	0.173	(d)+QED	0.905	0.933	1.0	0.072
(e)+SAS	0.820	0.541	1.0	0.125	(e)+QED	0.824	0.571	1.0	0.081

Table 6. 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 this is the reason for large overlap between distributions generated for QED control. Some examples of generated molecules for two scaffolds are given in Figure 8. In all the generated molecules, we see that the conditioned scaffold is maintained. Subfigures also show the molecules conditioned on scaffold + logP and scaffold + SAS. We see the addition of 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. As expected, property-based clusters are not as well formed as before. However, there is a good separation between the clusters for two property control. We can also see that the intended values of molecular properties are close to the centers of these clusters. This can further be verified by results reported for multi-property control in Table 7. For three property control one of the clusters (red) is not well formed due to highly constraining property values. We see that

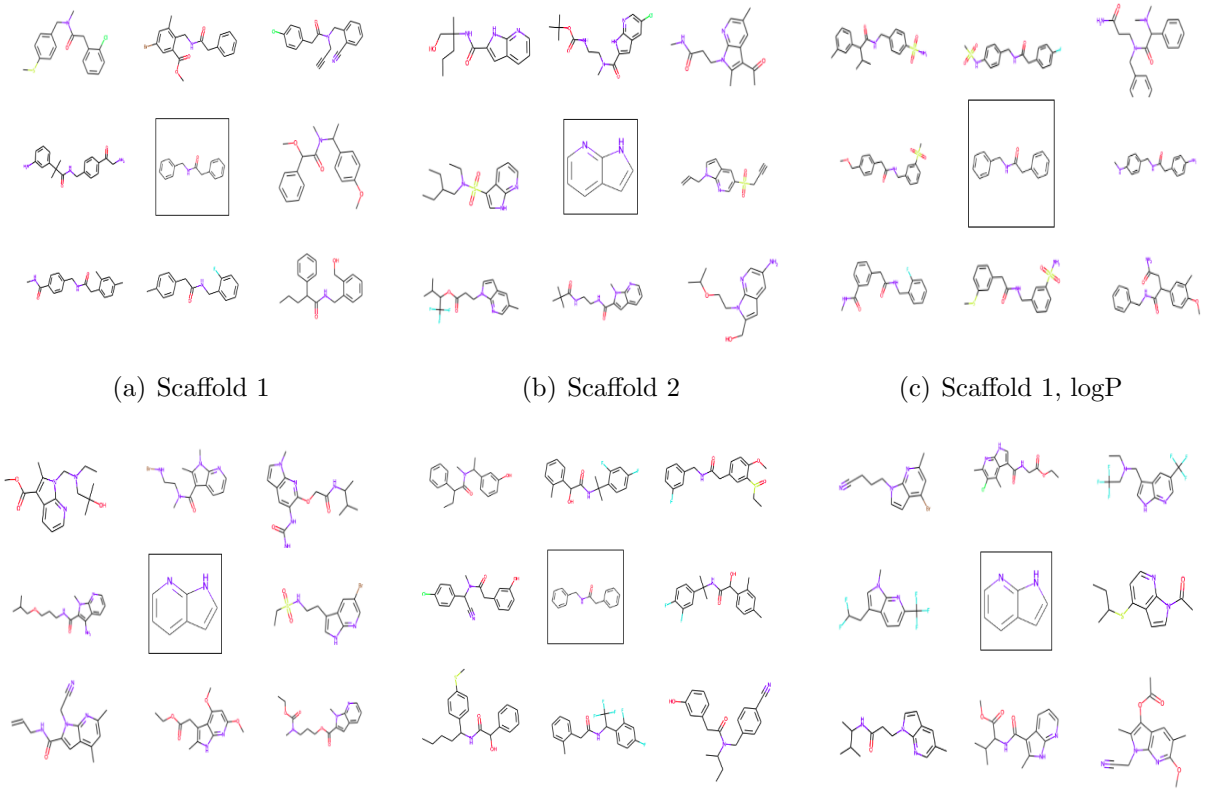
Table 7: Scaffold + Multi-property conditional training on MOSES dataset. Temperature 1.6 was used. Metric calculated only for molecules having tanimoto similarity of the scaffold of the generated molecule and the scaffold used for condition greater than 0.8. **(a)** O=C(Cc1ccccc1)NCc1ccccc1 **(b)** c1cnc2[nH]ccc2c1 **(c)** c1ccc(-c2ccnnc2)cc1 **(d)** c1ccc(-n2cnc3ccccc32)cc1 **(e)** O=C(c1cc[nH]c1)N1CCN(c2ccccc2)CC1

Cond	Validity	Unique	Novelty	MAD_TPSA	MAD_logP
(a)+TPSA+logP	0.812	0.737	1.0	3.667	0.249
(b)+TPSA+logP	0.693	0.931	1.0	4.117	0.199
(c)+TPSA+logP	0.830	0.852	1.0	3.903	0.152
(d)+TPSA+logP	0.773	0.818	1.0	4.617	0.204
(e)+TPSA+logP	0.776	0.511	0.999	4.046	0.242

Cond	Validity	Unique	Novelty	MAD_SAS	MAD_logP
(a)+SAS+logP	0.727	0.818	1.0	0.146	0.255
(b)+SAS+logP	0.591	0.649	1.0	0.193	0.191
(c)+SAS+logP	0.75	0.711	1.0	0.196	0.183
(d)+SAS+logP	0.748	0.731	1.0	0.171	0.246
(e)+SAS+logP	0.847	0.439	1.0	0.153	0.203

Cond	Validity	Unique	Novelty	MAD_TPSA	MAD_SAS
(a)+TPSA+SAS	0.751	0.901	1.0	3.947	0.192
(b)+TPSA+SAS	0.649	0.744	1.0	5.120	0.226
(c)+TPSA+SAS	0.683	0.803	1.0	4.074	0.210
(d)+TPSA+SAS	0.733	0.861	1.0	4.345	0.199
(e)+TPSA+SAS	0.838	0.482	1.0	3.827	0.162

Cond	Validity	Unique	Novelty	MAD_TPSA	MAD_logP	MAD_SAS
(a)+TPSA+logP+SAS	0.618	0.681	1.0	4.935	0.551	0.311
(b)+TPSA+logP+SAS	0.653	0.649	1.0	5.325	0.238	0.262
(c)+TPSA+logP+SAS	0.582	0.620	1.0	5.318	0.292	0.242
(d)+TPSA+logP+SAS	0.530	0.646	1.0	5.559	0.531	0.309
(e)+TPSA+logP+SAS	0.754	0.388	1.0	5.729	0.403	0.241



(a) Scaffold 1 (b) Scaffold 2 (c) Scaffold 1, logP

(d) Scaffold 2, logP (e) Scaffold 1, SAS (f) Scaffold 2, SAS

Figure 8: Scaffold 1: O=C(Cc1ccccc1)NCc1ccccc1. Scaffold 2: c1cnc2[nH]ccc2c1. In all the subfigures, the molecule in black box is the scaffold used for conditional generation. **(a, b)** 8 random generated molecules having the same scaffold as scaffold 1 and 2 respectively. **(c, d)** Conditioned on scaffold as well as logP = 2. **(e, f)** Conditioned on scaffold as well as SAS = 2.75.

the rest of the clusters are largely well formed and separated.

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 generate valid SMILES strings that satisfy chemical valencies. We see through our benchmarking experiments that LigGPT shows very good validity scores even with sampling temperature as high

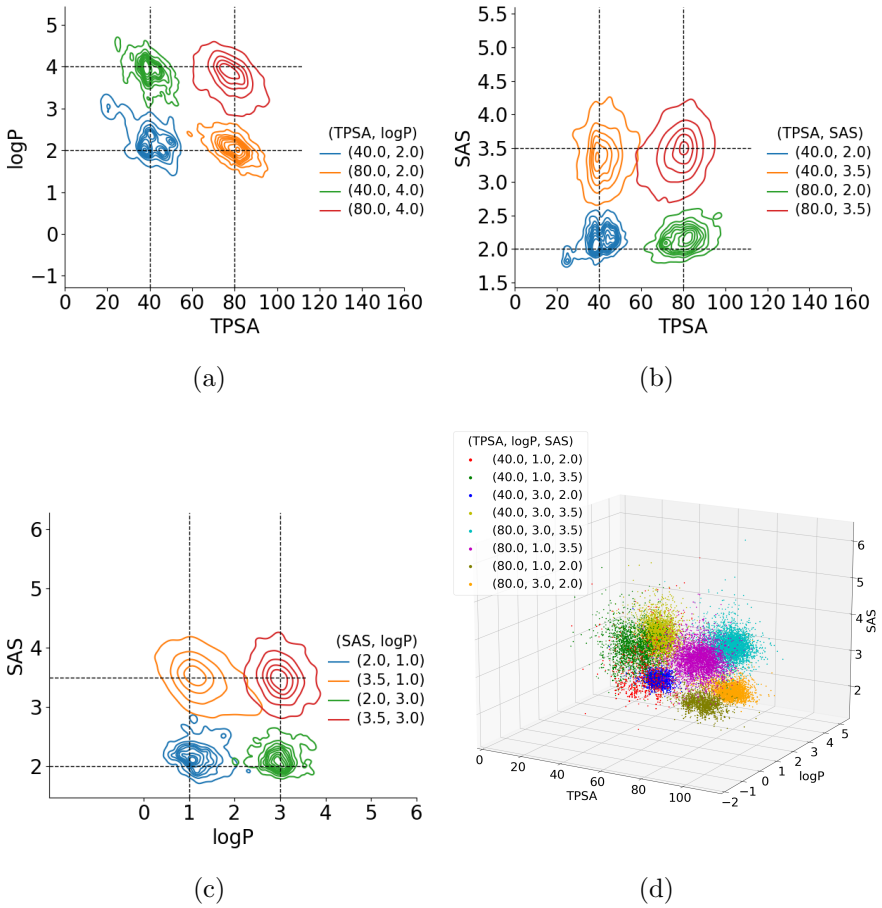


Figure 9: Distribution of property of generated molecules conditioned on: Scaffold + **(a)** TPSA + logP **(b)** SAS + TPSA **(c)** SAS + logP **(d)** TPSA + logP + SAS. Trained on MOSES dataset and temperature 1.6 used.

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.

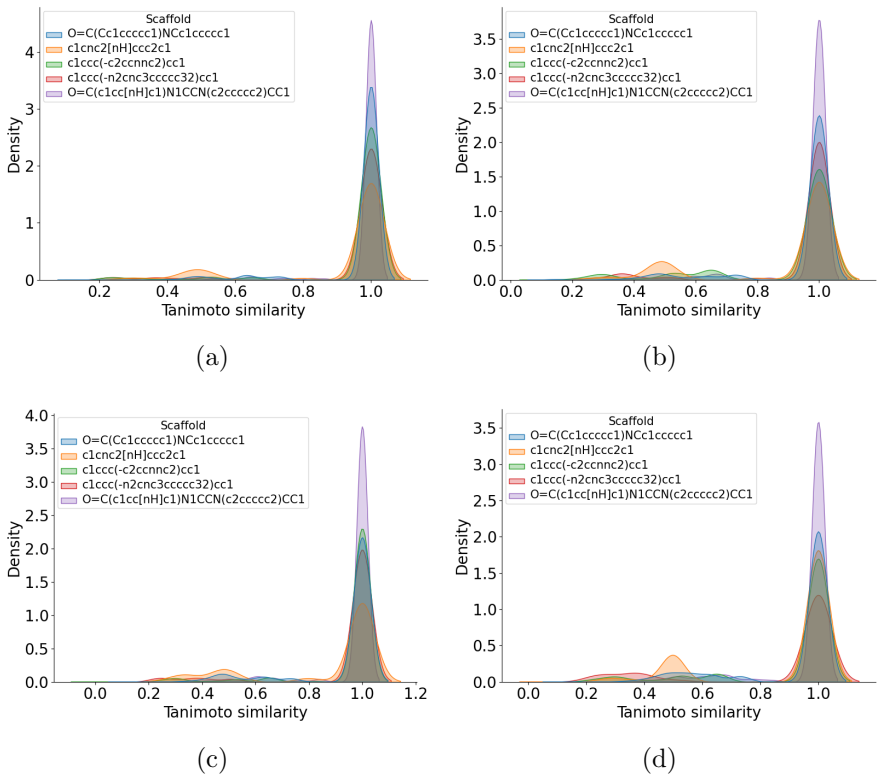


Figure 10: Distribution of tanimoto similarity of the scaffolds of the generated molecules and the scaffold used for condition for **(a)** TPSA + logP **(b)** SAS + TPSA **(c)** SAS + logP **(d)** TPSA + logP + SAS. Trained on MOSES dataset and temperature 1.6 used.

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. The implementation of the model is available at `https://github.com/VirajBagal/LigGPT`

Acknowledgement

The authors thank Manasa k, . . . for their comments during the preparation of the manuscript.

Supporting Information Available

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) Schneider, G.; Fechner, U. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery* **2005**, *4*, 649–663.

(11) 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*.

(12) 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.

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

(14) 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.

(15) 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.

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

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

(18) 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.

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

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

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

(22) 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.

(23) 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.

(24) 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.

(25) 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.

(26) 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.

(27) 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.

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

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

(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) 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.

(37) 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.

(38) 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.

(39) 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**,

(40) 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.

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

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

(43) 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.

(44) 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.

Graphical TOC Entry