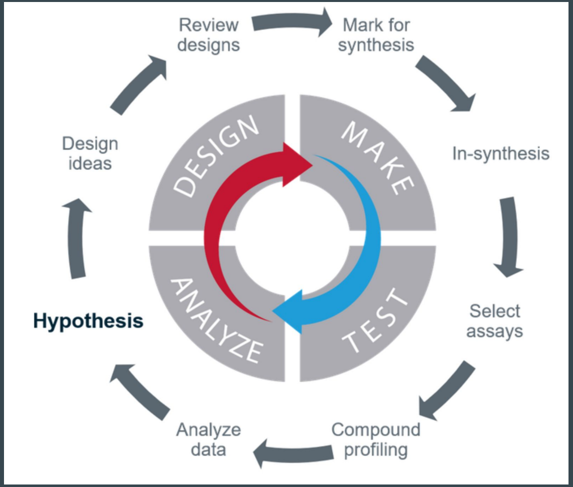
Barking up the right tree: an approach to search over molecule synthesis DAGs

Authors: John Bradshaw, Brooks Paige, Matt J. Kusner, Marwin H. S. Segler, José Miguel Hernández-Lobato

Presenter: Jun Chen

Introduction:

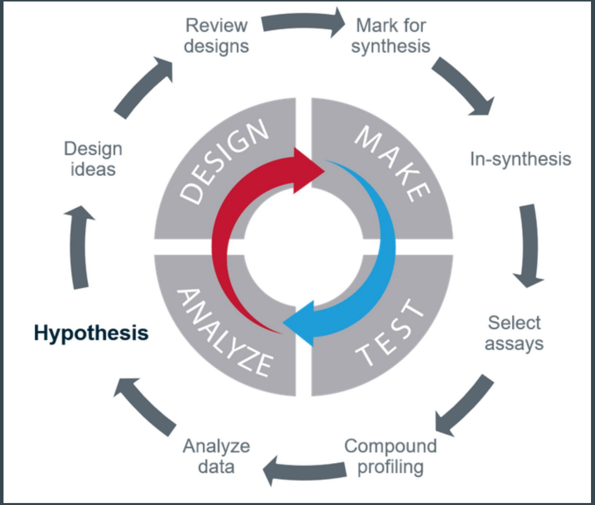
- Important of designing and discovering new molecules
- The molecular discovery process usually proceeds in design-make-test-analyze cycles, where new molecules are designed, made in the lab, tested in lab-based experiments to gather data, which is then analyzed to inform the next design step.



Goals to accelerate the DMTA cycles

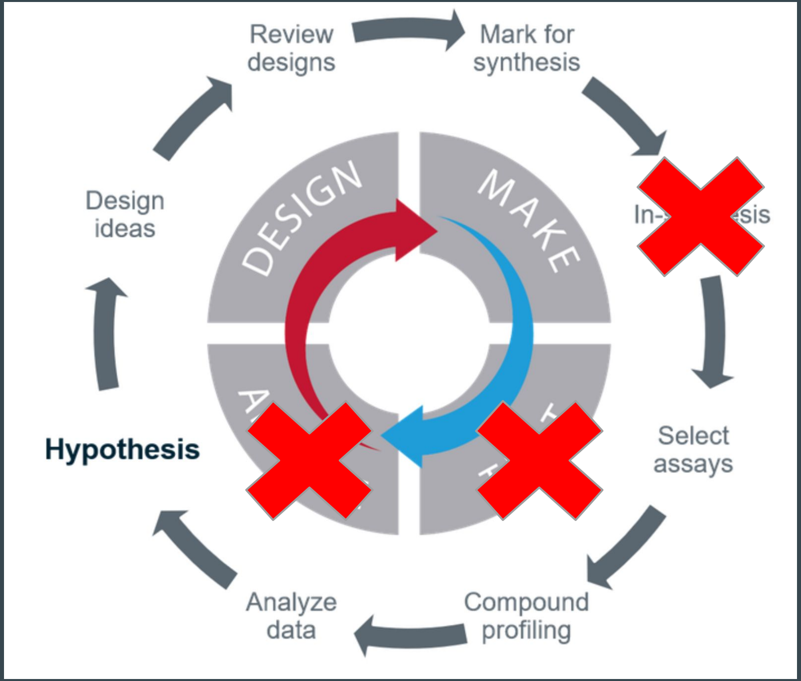
G1. Learning strong generative models of molecules that can be used to sample novel molecules, for downstream screening and scoring tasks

G2. Molecular optimization: Finding molecules that optimize properties of interest (e.g. binding affinity, solubility, non-toxicity, etc.).



Problem: Synthesize

Can the designed molecules be synthesize ?



Generating multi-step molecular synthesis routes

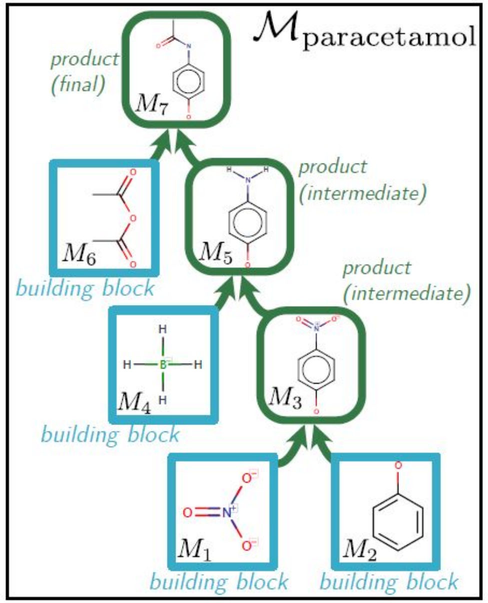
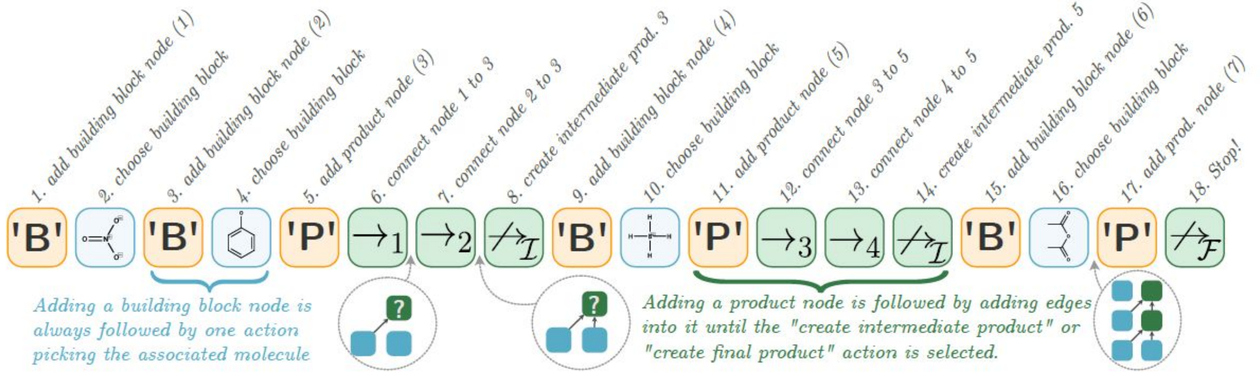


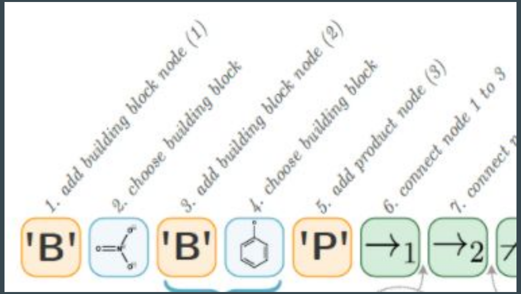
Figure 1: An example synthesis DAG for paracetamol [19] Note that we are ignoring some reagents, conditions and details of chirality for simplicity.

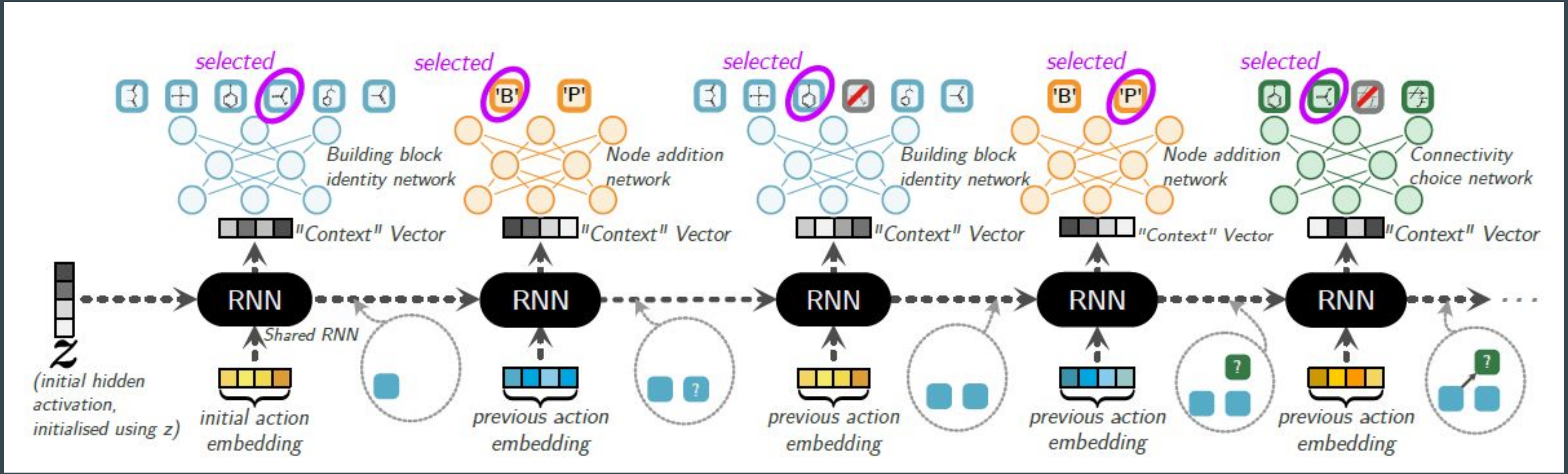


1. add building block node (1)
2. choose building block
3. add building block node (2)
4. choose building block
5. add product node (3)
6. connect node 1 to 3
7. connect node 2 to 3
8. create intermediate prod. 3
9. add building block node (4)
10. choose building block
11. add product node (5)
12. connect node 3 to 5
13. connect node 4 to 5
14. create intermediate prod. 5
15. add building block node (6)
16. choose building block
17. add prod. node (7)
18. Stop!
'B'
'B'
'P'
$\rightarrow_1$
$\rightarrow_2$
$\not\rightarrow_{\mathcal{I}}$
'B'
'P'
$\rightarrow_3$
$\rightarrow_4$
$\not\rightarrow_{\mathcal{I}}$
'B'
'P'
$\not\rightarrow_{\mathcal{F}}$
Adding a building block node is always followed by one action picking the associated molecule
Adding a product node is followed by adding edges into it until the "create intermediate product" or "create final product" action is selected.

Figure 2: An example of how we can *serialize* the construction of the DAG shown in Figure 1, with the corresponding DAG at that point in the sequence shown for three different time-points in the grey circles. The serialized construction sequence consists of a sequence of actions. These actions can be classified into belonging to three different types: (A1) node addition, (A2) building block molecular identity, and (A3) connectivity choice. By convention we start at the building block node that is furthest from the final product node, sampling randomly when two nodes are at equivalent distances.

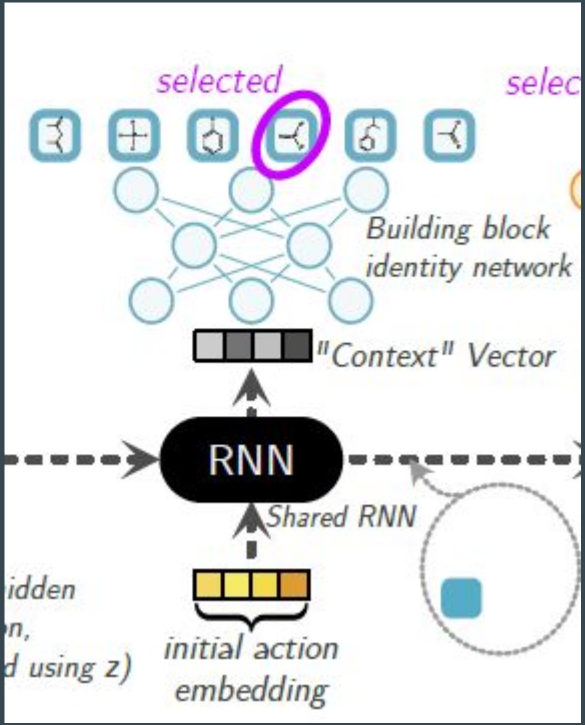
Define a probabilistic distribution over DAG



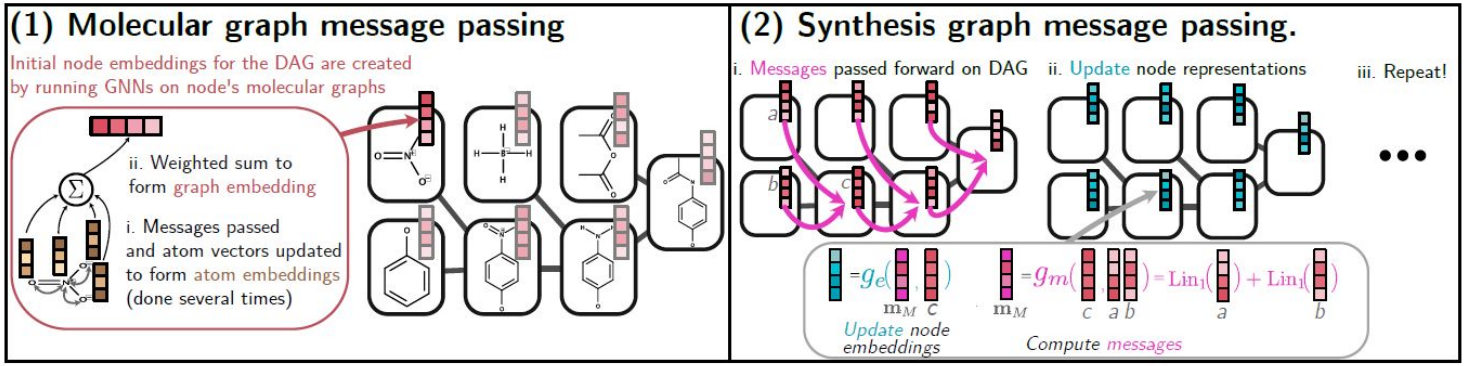

selected
selected
selected
selected
selected
'B'
'P'
Building block identity network
Node addition network
Building block identity network
Node addition network
Connectivity choice network
"Context" Vector
RNN
Shared RNN
z
(initial hidden activation, initialised using z)
initial action embedding
previous action embedding
?

Actions involving with molecular graph

- The model compute embeddings that also take the structure of the molecular graph into account.
- To compute molecular embeddings, the author choose deep graph neural networks (GNN), which allow to learn which characteristic of a graph are important to the task.



Autoencoder (DoG-AE)



(1) Molecular graph message passing
Initial node embeddings for the DAG are created by running GNNs on node's molecular graphs
ii. Weighted sum to form graph embedding
i. Messages passed and atom vectors updated to form atom embeddings (done several times)
(2) Synthesis graph message passing.
i. Messages passed forward on DAG
ii. Update node representations
iii. Repeat!
a
b
c
$= g_e(\mathbf{m}_M, c)$
Update node embeddings
$= g_m(c, a\, b) = \mathrm{Lin}_1(a) + \mathrm{Lin}_1(b)$
$\mathbf{m}_M$
Compute messages

Figure 4: The encoder embeds the DAG of Graphs (DoG) into a continuous latent space. It does this using a two-step hierarchical message passing procedure. In step 1 (Molecular graph message passing) it computes initial embeddings for the DAG nodes by forming graph-level embeddings using a GNN on the molecular graph associated with each node. In step 2 (Synthesis graph message passing) a message-passing algorithm is again used, however, this time on the synthesis DAG itself, passing messages forward. In our experiments we use GGNNs [45] for both message passing steps (see the Appendix for further details). The final representation of the DAG is taken from the node embedding of the final product node.

Basic generator DoG-Gen: molecular optimization via fine-tuning

- G2: performing molecular optimization via fine-tuning or reinforcement learning
- A model trained without latent space
- First pre-trained via maximum likelihood to match the training dataset distribution p(M).
- Then for optimization, fine tune the weights of the decoder by sampling a large number of candidate DAGs from the model, ranking them according to a target, and then fine-tuning the model's weights on the top K samples

Experiments results

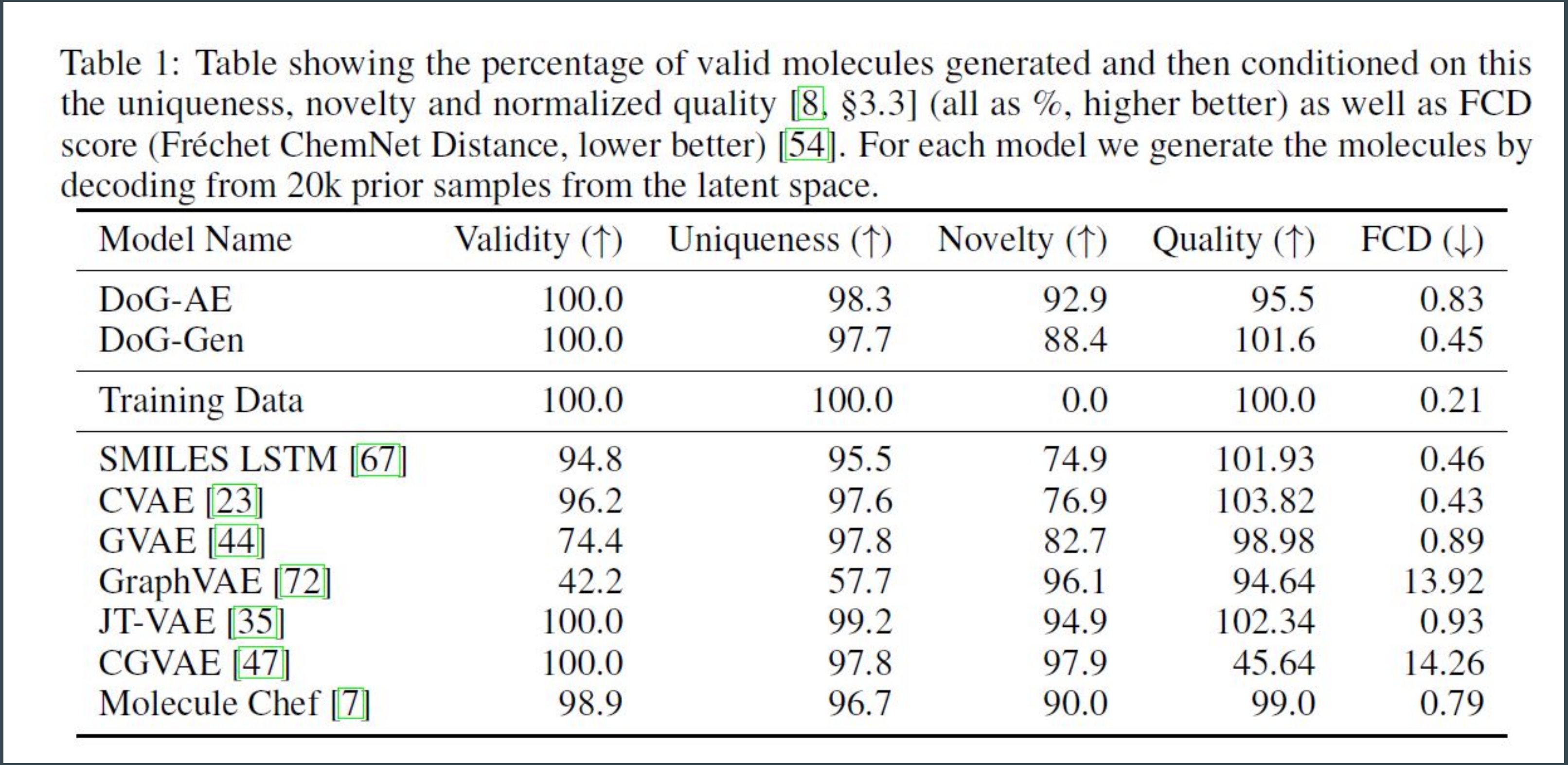


Table 1: Table showing the percentage of valid molecules generated and then conditioned on this the uniqueness, novelty and normalized quality [8, §3.3] (all as %, higher better) as well as FCD score (Fréchet ChemNet Distance, lower better) [54]. For each model we generate the molecules by decoding from 20k prior samples from the latent space.

Model Name	Validity (↑)	Uniqueness (↑)	Novelty (↑)	Quality (↑)	FCD (↓)
DoG-AE	100.0	98.3	92.9	95.5	0.83
DoG-Gen	100.0	97.7	88.4	101.6	0.45
Training Data	100.0	100.0	0.0	100.0	0.21
SMILES LSTM [67]	94.8	95.5	74.9	101.93	0.46
CVAE [23]	96.2	97.6	76.9	103.82	0.43
GVAE [44]	74.4	97.8	82.7	98.98	0.89
GraphVAE [72]	42.2	57.7	96.1	94.64	13.92
JT-VAE [35]	100.0	99.2	94.9	102.34	0.93
CGVAE [47]	100.0	97.8	97.9	45.64	14.26
Molecule Chef [7]	98.9	96.7	90.0	99.0	0.79

The latent space of synthesis DAGs

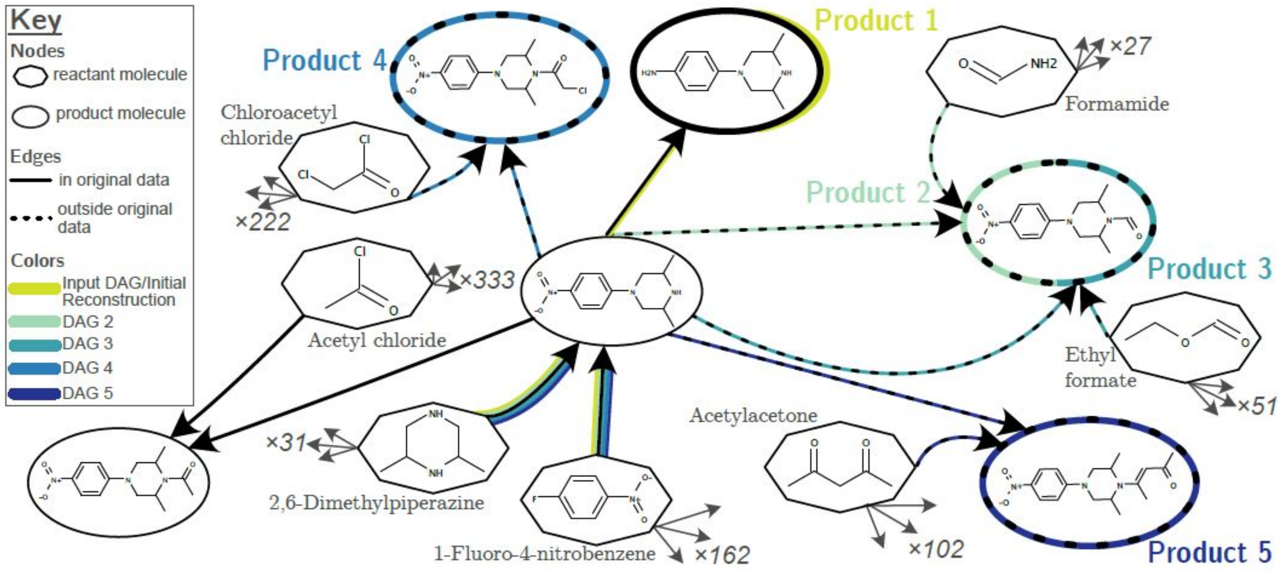


Figure 5: Using a variant of the DoG-AE model, as we randomly walk in the latent space we decode out to similar DAGs nearby, unseen in training. Reactions and nodes that exist in our original dataset are outlined in solid lines, whereas those that have been discovered by our model are shown with dashed lines.

Optimization

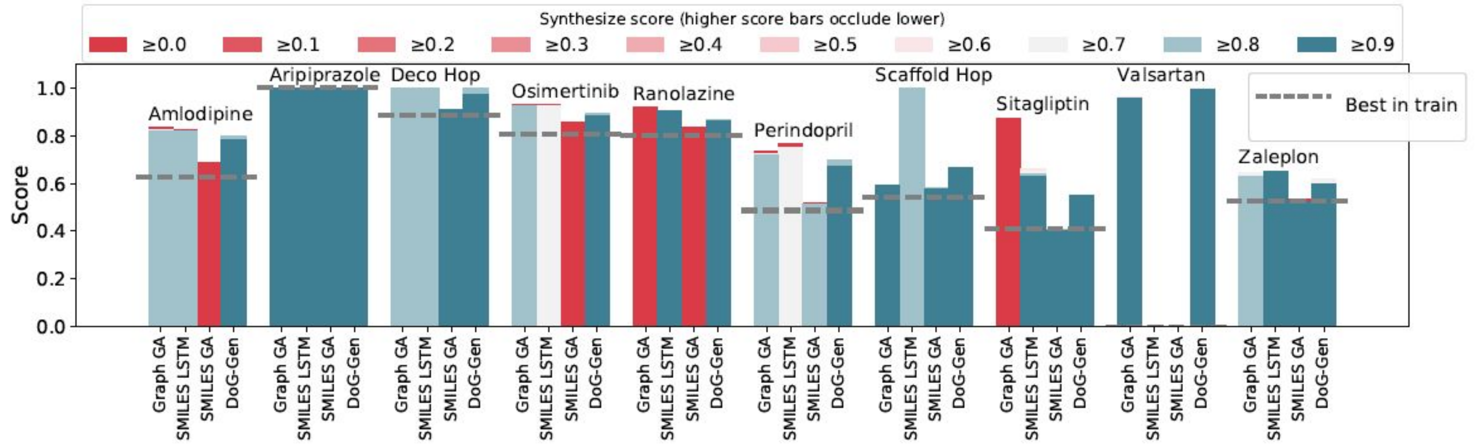
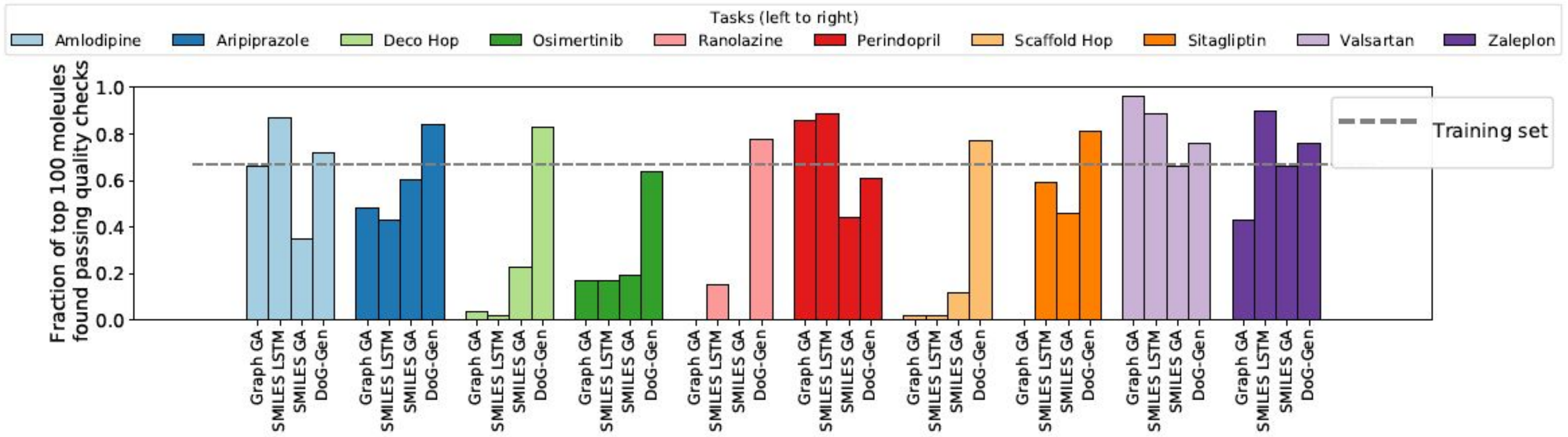


Figure 6: The score of the best molecule found by the different approaches over a series of ten GuacaMol benchmark tasks [8, §3.2], with the task name labeled above each set of bars. GuacaMol molecule scores (y-axis) range between 0 and 1, with 1 being the best. We also use colors to indicate the synthesizability score of the best molecule found. Note that bars representing a molecule within a higher synthesizability score bucket (e.g blue) will occlude lower synthesizability score bars (e.g. red). The dotted gray lines represent the scores of the best molecule in our training set.

More experiments result



	Frac. Synthesizable (↑)	Synth. Score (↑)	Median # Steps (↓)	Quality (↑)
DoG-Gen	0.9	0.76	4	0.75
Graph GA	0.42	0.33	6	0.36
SMILES LSTM	0.48	0.39	5	0.49
SMILES GA	0.29	0.25	3	0.39

Thank you