Decoding Molecular Graph Embeddings with Reinforcement Learning

Steven Kearnes [1] **Li Li** [1] **Patrick Riley** [1]

Abstract

We present RL-VAE, a graph-to-graph variational autoencoder that uses reinforcement learning to decode molecular graphs from latent embeddings. Methods have been described previously for graph-to-graph autoencoding, but these approaches require sophisticated decoders that increase the complexity of training and evaluation (such as requiring parallel encoders and decoders or non-trivial graph matching). Here, we repurpose a simple graph generator to enable efficient decoding and generation of molecular graphs.

1. Introduction

There are two prevailing approaches for *in silico* molecular property optimization—molecular autoencoders and generative models—and there is a very real division in the field depending on whether the optimization happens in the latent space of an autoencoder or by conditioning a generative process. Both types of models have advantages and disadvantages. Autoencoders have a latent space that naturally lends itself to continuous optimization; this was elegantly demonstrated by Gómez-Bombarelli et al. (2018). On the other hand, these models struggle to decode valid molecular graphs due to their discrete nature and the constraints of chemistry. Early work focused on character-level autoencoders trained on SMILES strings (Weininger, 1988)—for example, caffeine is represented as `CN1C=NC2=C1C(=O)N(C(=O)N2C)C`—but the syntactic complexity and long-range dependencies of SMILES make decoding particularly difficult. Subsequent models incorporated grammatical and syntactic constraints to avoid generating invalid strings (Dai et al., 2018; Kusner et al., 2017).

Attempts to decode graphs directly (*i.e.*, without a SMILES intermediate) have led to useful but complicated models. For example, Jin et al. (2018) describe a model that requires reducing each molecule to a tree structure and combining information from two parallel decoders to expand a predicted tree to a valid graph, and Simonovsky & Komodakis (2018) predict entire graphs in one step but require complicated graph-matching and heuristics to avoid disconnected or invalid results.

Recent work in generative models has sidestepped the need for differentiable graph-isomorphic loss functions that makes graph decoders awkward, instead borrowing methods from reinforcement learning to iteratively construct chemically valid graphs according to a learned value function. Examples include You et al. (2018); Li et al. (2018b;a); Zhou et al. (2018), which all formulate graph generation as a Markov decision process (MDP) and learn policies for generating molecules that match specific criteria. While effective for generation and optimization, these methods sacrifice the notion of a continuous latent space with potentially useful structure (Mikolov et al., 2013).

Here we describe a model, RL-VAE (Reinforcement Learning Variational Autoencoder), which combines advantages from both types of models: specifically, the model has a latent space that enables continuous optimization while using an iterative MDP-based graph decoder that guarantees chemically valid output. To the best of our knowledge, this is the first example of a variational autoencoder that uses a reinforcement learning agent as the decoder.

2. Methods

The overall architecture of the RL-VAE is depicted in Figure 1. The following subsections detail the model and our training procedures.

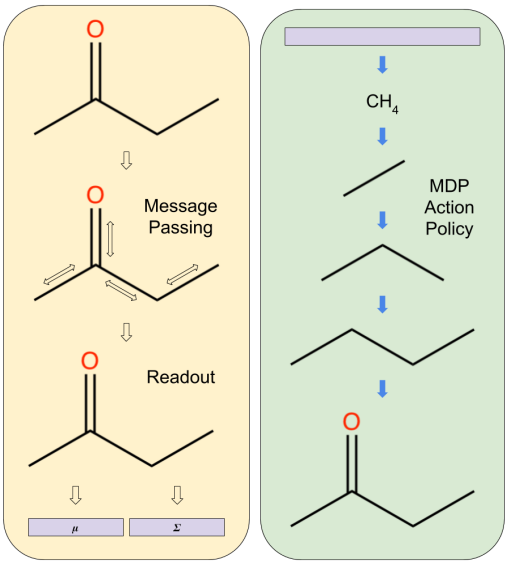
2.1. Graph Encoder

Input molecular graphs were represented with node- and edge-level features: one-hot encodings of atom and bond types, respectively. The set of allowed atoms was {H, C, N, O, F}; the set of allowed bonds was {single, double, triple, aromatic}. No other input features were used. The node and edge features were linearly mapped (without any bias terms) to a learned latent space with dimension 128 prior to being fed into the encoder.

Molecular graphs were encoded using a message passing neural network (MPNN) (Gilmer et al., 2017). In particular,

[1]Google Research, Mountain View, California, USA. Correspondence to: Steven Kearnes <kearnes@google.com>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data.



(a) Encoder. (b) Decoder.

Figure 1. Graphical description of the RL-VAE architecture. (a) The encoder converts a molecular graph into an embedding distribution parameterized by μ and Σ using a message passing neural network (MPNN). (b) The decoder samples a vector from the embedding distribution and decodes it into a molecular graph using a learned value function for a molecule-specific Markov decision process (MDP).

we followed the description from Li et al. (2018a), including a gating network that determines the relative contribution of each node to the graph-level representation.

The MPNN used two graph convolutional layers; the layers did not apply any learned transformations to the messages, but a gated recurrent unit (GRU) (Cho et al., 2014) with learned parameters was used to update node states with the aggregated messages from neighboring nodes. The final node features were mapped to 256-dimensional vectors with a learned linear transform, gated, and summed to produce the graph-level representation (this process is referred to as a "readout" transformation). Two readout transformations with separate weights were used to predict the mean μ and (log) standard deviation Σ used for variational sampling of the latent space (Doersch, 2016). The final embedding space thus had 256 dimensions; molecules were embedded as distributions—parameterized by μ and Σ—from which individual molecular embeddings were sampled.

2.2. Graph Decoder

Molecular embeddings were decoded using a variant of MolDQN, a graph generator that uses a Markov decision process (MDP) to construct graphs sequentially (Zhou et al., 2018). The model was trained with Double Q-learning (van Hasselt et al., 2015) to approximate $V(s)$, the state value function.

The MDP was modified slightly from Zhou et al. (2018). Specifically: (1) bonds could not be removed or promoted (*i.e.*, the model was not allowed to backtrack or update earlier steps); (2) when two single-atom fragments were created by removing a bond, both fragments were added to the action set (the original implementation added only the first fragment in a sorted list of SMILES); and (3) our implementation did not add triple bonds between existing atoms in the graph (to avoid forming rings containing triple bonds). Notably, this MDP guarantees that all generated molecules are chemically valid.

To condition the generative process to reconstruct specific molecules, an embedding of the target molecule was concatenated to the embedding of the current state as input to the value function. Additionally, the value function received information about the number of steps remaining in the episode, as well as whether the proposed action lead to a terminal state (*i.e.*, whether this was the last step of the episode) (Pardo et al., 2017). The state embedding was computed with a separate MPNN; the architecture of this model was identical to the encoder described in Section 2.1 except that it was not stochastic—only a single graph-level vector was computed. Thus, the value function is:

$$V_T(s, y, t) = g\left([f_{\text{state}}(s), f_{\text{target}}(y), t_1, t_2]\right) \quad (1)$$

$$t_1 = \frac{2(T-t)}{T} - 1 \quad (2)$$

$$t_2 = \mathbb{1}(t = T-1), \quad (3)$$

where f_{state} is the state embedding MPNN, f_{target} is the (stochastic) target embedding MPNN, y is the input/target molecule, T is the maximum number of steps in the episode (set to 20 in our experiments), and t is the number of steps taken in the episode so far (*i.e.*, the number of steps prior to but not including s). Note that (1) the target molecule embedding $f_{\text{target}}(y)$ was only sampled once per episode, not once per step; and (2) arbitrary points in the latent space can be decoded by replacing $f_{\text{target}}(y)$ with any embedding vector. The function g was implemented as a fully connected neural network with a single hidden layer containing 256 ReLUs.

The reward is a numerical description of progress toward (or away from) a goal. In our case, the goal is reconstruction of an input molecule, and the reward measures the similarity between the input (target) molecule and the current state of the generator. In practice, we averaged over four different measures that capture different aspects of molecular similarity and provide complementary information to guide the generation process. The similarity measures that comprise the reward function can be divided into two groups:

2.2.1. Fingerprint Similarity

We used three types of molecular fingerprints, all computed with RDKit (Landrum et al., 2006): extended-connectivity (ECFP), also known as "circular" or "Morgan" fingerprints, with $r = 3$ (Rogers & Hahn, 2010); topological (path); and atom-pair (Carhart et al., 1985). For each fingerprint, we computed the sparse (unhashed) Tversky similarity (Horvath et al., 2013) using the following (α, β) pairs: $\{(0.5, 0.5), (0.95, 0.05), (0.05, 0.95)\}$. The resulting similarity values were then averaged to give a single value for each fingerprint type.

2.2.2. Atom Count Similarity

We computed the Tanimoto similarity across atom types:

$$T(A, B) = \frac{\sum_i \min\{\texttt{count}(A, i), \texttt{count}(B, i)\}}{\sum_i \max\{\texttt{count}(A, i), \texttt{count}(B, i)\}}, \quad (4)$$

where A and B are molecules, and $\texttt{count}(A, i)$ is the number of atoms in molecule A with type i.

2.3. Data

We used the QM9 dataset (Ramakrishnan et al., 2014), which contains 133 885 molecules with up to nine heavy atoms in the set {C, N, O, F}. The data was filtered to remove 1845 molecules containing atoms with formal charges (since our MDP cannot reproduce them) and randomly divided into ten folds. Models were trained on eight folds (105 773 examples), with one fold each held out for validation ("tuning"; 13 154 examples) and testing (13 113 examples). Note that we did not repeat model training with different fold subsets; *i.e.*, we did not perform cross-validation.

The QM9 dataset is suited for experimentation since its molecules are relatively small. However, they tend to be relatively complex since the dataset attempts to exhaustively enumerate all possible structures of that size and containing that set of atoms. Additionally, fingerprint-based similarity metrics can become brittle when applied to small molecules, since they do not set as many bits in the fingerprint—see Figure A1 in the Supplemental Material for examples of QM9-sized molecule pairs with different similarity values.

2.4. Model Training

Models were implemented with TensorFlow (Abadi et al., 2015) and trained with the Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Training proceeded for ∼10 M steps with a batch size of 128; the learning rate was smoothly decayed from 10^{-5} with a decay rate of 0.99 every 100 000 steps.

2.4.1. Training Loss

The training loss had the form:

$$\mathcal{L}(s, y, \mu, \Sigma) = H\left(\mathcal{L}_{\text{TD}}(s, y)\right) + \lambda \mathcal{L}_{\text{KL}}\left(\mu, \Sigma\right). \quad (5)$$

$\mathcal{L}_{\text{TD}}(s)$ is the temporal difference loss used to train the decoder:

$$\mathcal{L}_{\text{TD}}(s, y) = V_T(s, y, t) - \hat{V}_T(s, y, t) \quad (6)$$

$$\hat{V}_T(s, y, t) = R(s, y) + \gamma \max_{s'} V_T(s', y, t + 1), \quad (7)$$

where R is the reward function (see Section 2.2), γ is the discount factor (we used 0.99), and H is the Huber loss function with $\delta = 1$ (Huber, 1964). Additionally, the weight on this loss for each example was adjusted such that terminal and non-terminal states had approximately the same "mass" in each batch.

$\mathcal{L}_{\text{KL}}(h)$ is the Kullback–Leibler divergence between a multivariate unit Gaussian prior and the variational distribution—parameterized by the predicted mean vector μ and diagonal covariance matrix Σ—from which the target molecule embedding is sampled. The hyperparameter λ controls the relative contribution of this term to the total loss ($\lambda = 10^{-5}$ in all of the experiments reported here).

2.4.2. Experience Replay

The states s used in training were uniformly sampled from an experience replay buffer (Mnih et al., 2015). The buffer was updated in each training step by sampling a batch of molecules from the training data and generating two episodes for each molecule:

1. The current value function approximation was used to run an episode with ϵ-greedy action selection. ϵ started at 1.0 and was smoothly decayed during training with a decay rate of 0.95 every 10 000 steps.
2. An idealized episode was created to reconstruct the target molecule (see Section A of the Supplemental Material for details). This ensured that the replay buffer included examples of productive steps.

The replay buffer stored states, rewards, actions, and the number of steps taken; each 20-step episode created 20 examples for the buffer. At the beginning of model training, the buffer was populated to contain at least 1000 examples by running several batches without any training. When the buffer reached the maximum size of 10 000 examples, the oldest examples were discarded to make room for new examples.

Each training step used a batch size of 8 to generate episode data for the replay buffer (corresponding to 320 buffer entries) and a batch size of 128 for sampling from the buffer to compute the training loss.

Table 1. Reconstruction accuracy for various models. The random walk model chooses actions randomly. The symbol γ is the discount factor in Eq. 7; $\gamma = 0$ only considers immediate rewards. Details on JT-VAE and GVAE training and evaluation are given in Section D of the Supplemental Material.

	TUNE	TEST
RANDOM WALK	0.00	0.00
RL-VAE ($\gamma = 0$)	0.03	0.03
RL-VAE ($\gamma = 0.99$)	0.66	0.67
JT-VAE (Jin et al., 2018)	0.74	0.74
GVAE (Kusner et al., 2017)	0.51	0.51

3. Results and Discussion

3.1. Reconstruction

After training an RL-VAE model on a subset of the QM9 data set, we evaluated reconstruction performance on the held-out tune and test sets, each containing ∼13 000 molecules (see Section 2.3). Each molecule was encoded as a distribution in the latent space, and a single embedding was sampled from this distribution and decoded.

We computed three metrics: (1) reconstruction accuracy, measured by canonical SMILES equivalence (ignoring stereochemistry, which is not predicted by the decoder); (2) Tanimoto similarity between input and output molecules; and (3) MDP edit distance, measured as the number of MDP steps required to reach the decoded molecule from the target molecule (calculated by brute-force search; see Section B of the Supplemental Material for details). We also verified that a greedy model with discount factor $\gamma = 0$ performed significantly worse. Reconstruction accuracy is shown in Table 1. Tanimoto similarity between input and output molecules fell off sharply if molecules were not perfectly reconstructed (Figure A2 and Figure A3 in the Supplemental Material), suggesting that MDP edit distance is a more useful metric for molecules of this size (see Section 2.3). Figure A4 in the Supplemental Material shows the distribution of MDP edit distances for the non-greedy model.

3.2. Sampling

We randomly chose two orthogonal directions that we used to explore the latent space, decoding molecules along the way. As we suspect is the case for most explorations of this type, the results are highly sensitive to the choice of directions, the starting molecule, the distance travelled in each direction, and the step size. Figure A5 in the Supplemental Material shows an example of such an exploration that reveals some local structure, although the decoded molecules do not exhibit very smooth transitions (*i.e.*, transitions often change more than one aspect of a structure).

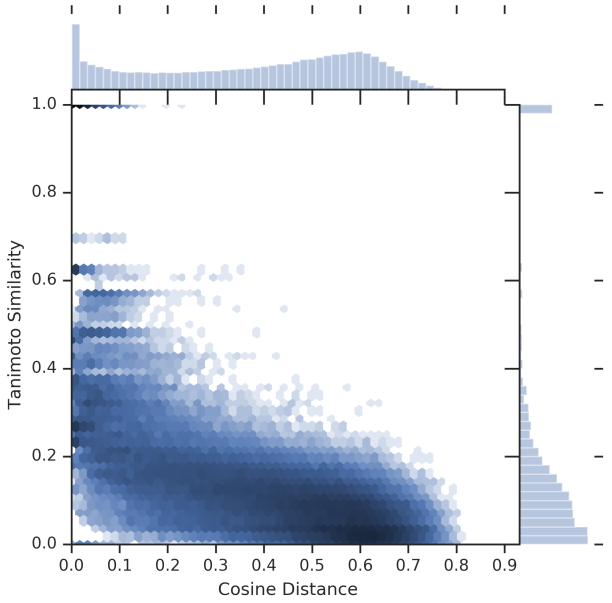


Figure 2. Tanimoto similarity between molecules decoded from original and perturbed embeddings as a function of cosine distance in the latent space (combined data from ten different original embeddings). The density color map is logarithmic.

As an attempt to more rigorously investigate the smoothness of the embedding space, we measured the similarity of decoded compounds as a function of cosine distance in the latent space (see Section C of the Supplemental Material for details). As shown in Figure 2, we observed that the similarity of the decoded molecules tends to decrease as the cosine distance increases, suggesting that the embedding space around these starting points has relatively smooth local structure (see the Supplemental Material for additional plots for each starting point).

4. Conclusion and Future Work

This paper describes a graph-to-graph autoencoder with an unusual training signal: temporal difference (TD) errors for Q-function predictions. Our preliminary results suggest that RL-VAE is a simple and effective approach that bridges the gap between embedding-based methods and generative models for molecular generation and optimization.

As this is a workshop paper describing preliminary work, we anticipate several avenues for future work, including: (1) additional hyperparameter optimization; (2) training on larger and more relevant data sets, such as ChEMBL (Gaulton et al., 2012); (3) evaluation on optimization tasks that are relevant for drug discovery—in particular, this excludes properties like logP or QED that are only useful as baselines (see Zhou et al. (2018)); and (4) Bayesian optimization as demonstrated by Gómez-Bombarelli et al. (2018).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Carhart, R. E., Smith, D. H., and Venkataraghavan, R. Atom pairs as molecular features in structure-activity studies: definition and applications. *J. Chem. Inf. Comput. Sci.*, 25(2):64–73, May 1985.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN Encoder-Decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, June 2014.

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. Syntax-Directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, February 2018.

Doersch, C. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, June 2016.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., and Overington, J. P. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res.*, 40(Database issue):D1100–7, January 2012.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, April 2017.

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., and Aspuru-Guzik, A. Automatic chemical design using a Data-Driven continuous representation of molecules. *ACS Cent Sci*, 4(2):268–276, February 2018.

Horvath, D., Marcou, G., and Varnek, A. Do not hesitate to use tversky-and other hints for successful active analogue searches with feature count descriptors. *J. Chem. Inf. Model.*, 53(7):1543–1562, July 2013.

Huber, P. J. Robust estimation of a location parameter. *Ann. Math. Stat.*, 35(1):73–101, March 1964.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, February 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, December 2014.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, March 2017.

Landrum, G. et al. RDKit: Open-source cheminformatics. http://www.rdkit.org, 2006.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, March 2018a.

Li, Y., Zhang, L., and Liu, Z. Multi-Objective de novo drug design with conditional graph generative model. *arXiv preprint arXiv:1801.07299*, January 2018b.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, January 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015.

Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. Time limits in reinforcement learning. *arXiv preprint arXiv:1712.00378*, December 2017.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Sci Data*, 1:140022, August 2014.

Rogers, D. and Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.*, 50(5):742–754, May 2010.

Simonovsky, M. and Komodakis, N. GraphVAE: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, February 2018.

van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, September 2015.

Weininger, D. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, February 1988.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. Graph convolutional policy network for Goal-Directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, June 2018.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. Optimization of molecules via deep reinforcement learning. *arXiv preprint arXiv:1810.08678*, October 2018.

Decoding Molecular Graph Embeddings with Reinforcement Learning
SUPPLEMENTAL MATERIAL

Steven Kearnes [1] **Li Li** [1] **Patrick Riley** [1]

A. Idealized episodes

To construct idealized episodes, molecules were broken into their constituent atoms and assembled one step at a time. When a new atom was added to the graph, bonds to existing atoms were added as additional steps (*i.e.*, atom and bond additions were interleaved in the episode). The first atom added to the graph corresponded to the first atom in the list of atoms returned by `RDKit`; subsequent atoms were added if they had a bond to at least one existing atom in the graph.

B. Brute-force search

MDP edit distance was calculated by brute-force search from molecule A to molecule B. Actions allowed from molecule A were exhaustively expanded until molecule B or a maximum step limit was reached. The MDP used for the brute-force search allowed a more flexible set of actions; in particular: (1) bond removal/promotion was allowed, (2) states were Kekulized so that bonds in aromatic systems could be modified, (3) rings of any size were allowed, (4) the "no modification" action was not allowed, and (5) bonds between ring atoms were allowed. This set of actions resulted a very flexible MDP that helped match the calculated edit distance to intuitive notions of distance.

C. Latent space embedding perturbations

For an array of scaling factors spanning $[-5, 5]$ in 0.1 intervals (excluding 0.0), we performed the following procedure: Starting with a single embedding, we uniformly sampled a 256-dimensional perturbation vector in the range $[0, 1)$, and decoded the perturbed embedding. This was repeated 100 times for each scaling factor, and the entire process was again repeated for ten different starting embeddings (representing ten different starting molecules; Figure A7), for a total of $100\,000$ perturbations. Cosine and Euclidean distances were calculated between the original embedding and the perturbed embedding; Tanimoto similarities between the input molecule and the molecule decoded from the perturbed embedding were calculated using sparse Morgan fingerprints with radius 3.

D. Literature model training and evaluation

D.1. JT-VAE

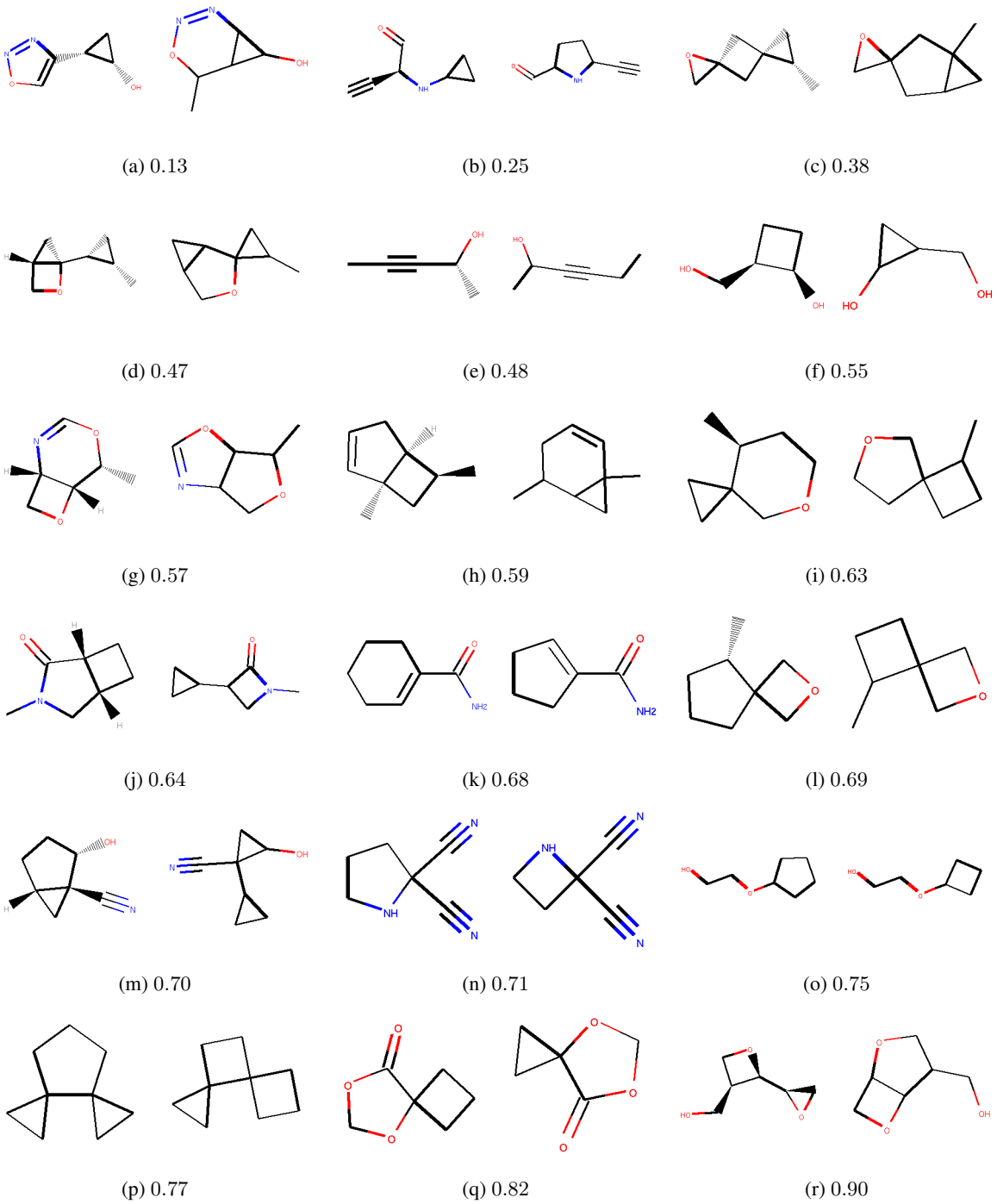
Code for the JT-VAE model described by Jin et al. (2018) was downloaded from `https://github.com/wengong-jin/icml18-jtnn`. The vocabulary set was extracted from the union of the training, tune and test sets. We used the default settings in the original implementation with our training set to train the model. The model was trained for $65\,000$ steps with batch size 32. Note that the data loader in this JT-VAE implementation drops samples if the number of samples in the last batch of each file chunk of data is less than the batch size. Therefore, the number of examples used to compute the reconstruction accuracy on the tune ($12\,736$) and test ($12\,704$) sets are slightly less than the total number of examples in the full tune ($13\,154$) and test ($13\,113$) sets.

D.2. GVAE

Code for the GVAE model described by Kusner et al. (2017) was downloaded from `https://github.com/mkusner/grammarVAE`. Training with default parameters was halted by early stopping after 27 epochs. The tune set accuracy associated with the loss used for early stopping was reported to be very low ($<5\%$), which did not agree with our manual calculation of $\sim 50\%$; we suspect this is because the accuracy calculation used in the code is based on element-wise binarized tensor equality (predicted probabilities are cast to binary predicted labels with a threshold of 0.5 by the `binary_accuracy` function in Keras), whereas the generated SMILES strings only reflect tensor elements that appear before the first predicted padding element.

[1]Google Research, Mountain View, California, USA. Correspondence to: Steven Kearnes <kearnes@google.com>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data.



(a) 0.13 (b) 0.25 (c) 0.38

(d) 0.47 (e) 0.48 (f) 0.55

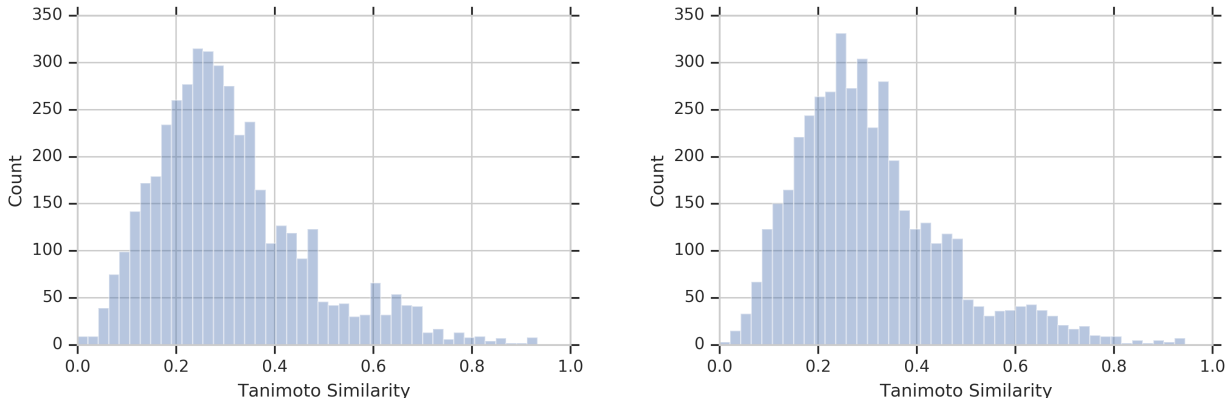
(g) 0.57 (h) 0.59 (i) 0.63

(j) 0.64 (k) 0.68 (l) 0.69

(m) 0.70 (n) 0.71 (o) 0.75

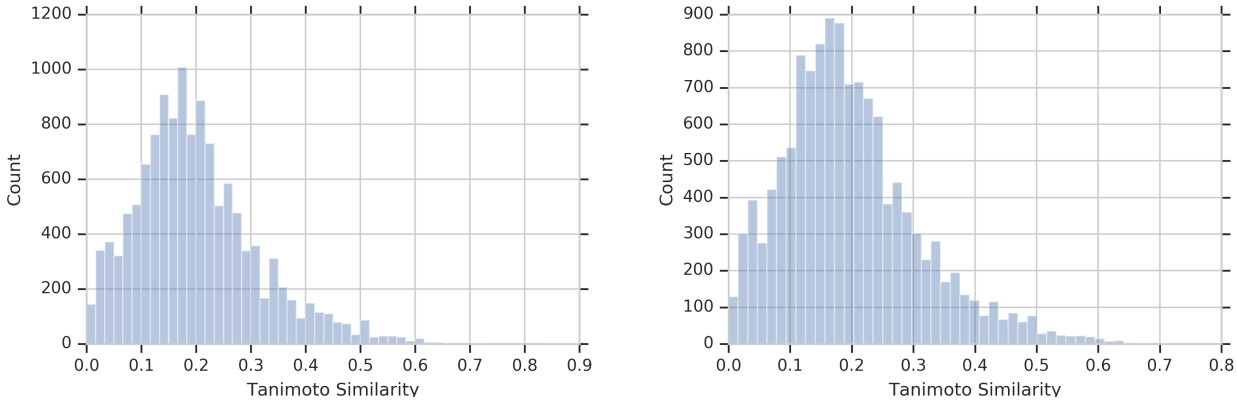
(p) 0.77 (q) 0.82 (r) 0.90

Figure A1. Tanimoto similarity for various molecule pairs. In each pair, the molecule on the left is taken from the QM9 tune set. Similarity is computed on sparse Morgan fingerprints with radius 3; note that stereochemistry is ignored. The interpretation of similarity values varies with the size of the molecule as the total number of bits set in the fingerprints changes.



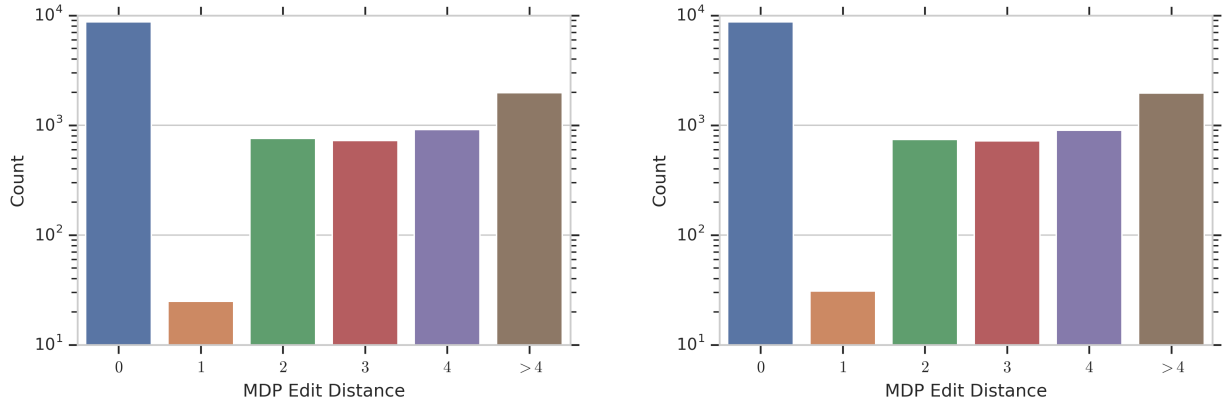
(a) Tune (b) Test

Figure A2. Tanimoto similarity of input and output molecules for $\gamma = 0.99$. Identical molecules (similarity $= 1$) are excluded for clarity. Similarity was computed on sparse Morgan fingerprints with radius 3.



(a) Tune (b) Test

Figure A3. Tanimoto similarity of input and output molecules for $\gamma = 0$. Identical molecules (similarity $= 1$) are excluded for clarity. Similarity was computed on sparse Morgan fingerprints with radius 3.



(a) Tune (b) Test

Figure A4. MDP edit distance between input and output molecules for $\gamma = 0.99$.

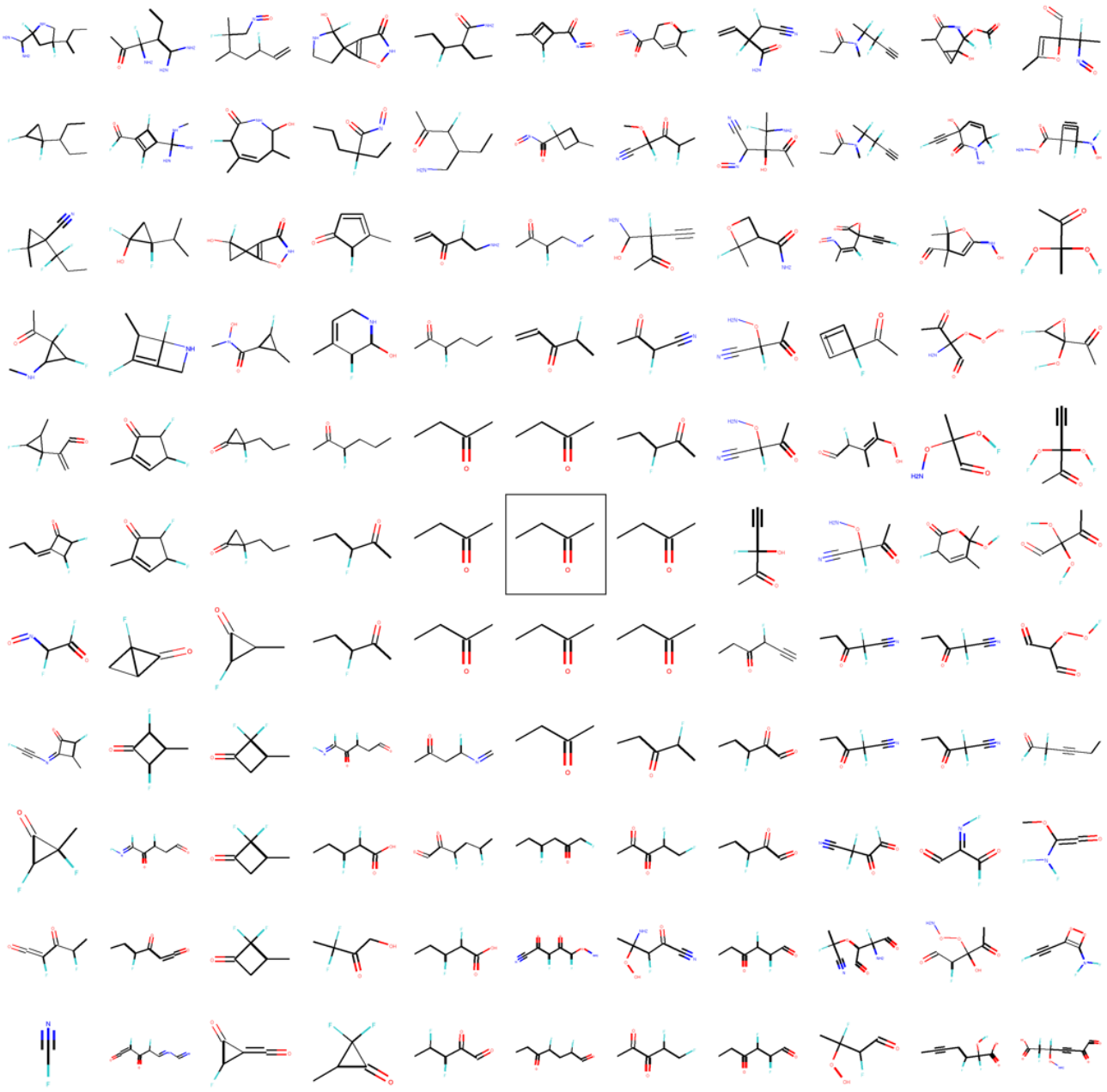


Figure A5. Exploration of the embedding space in two random orthogonal directions. The starting molecule (boxed) was `CCC(C)=O`. The edges of the image correspond to embeddings at ± 20x a unit vector in that direction; intermediate points were sampled at intervals of 4x.

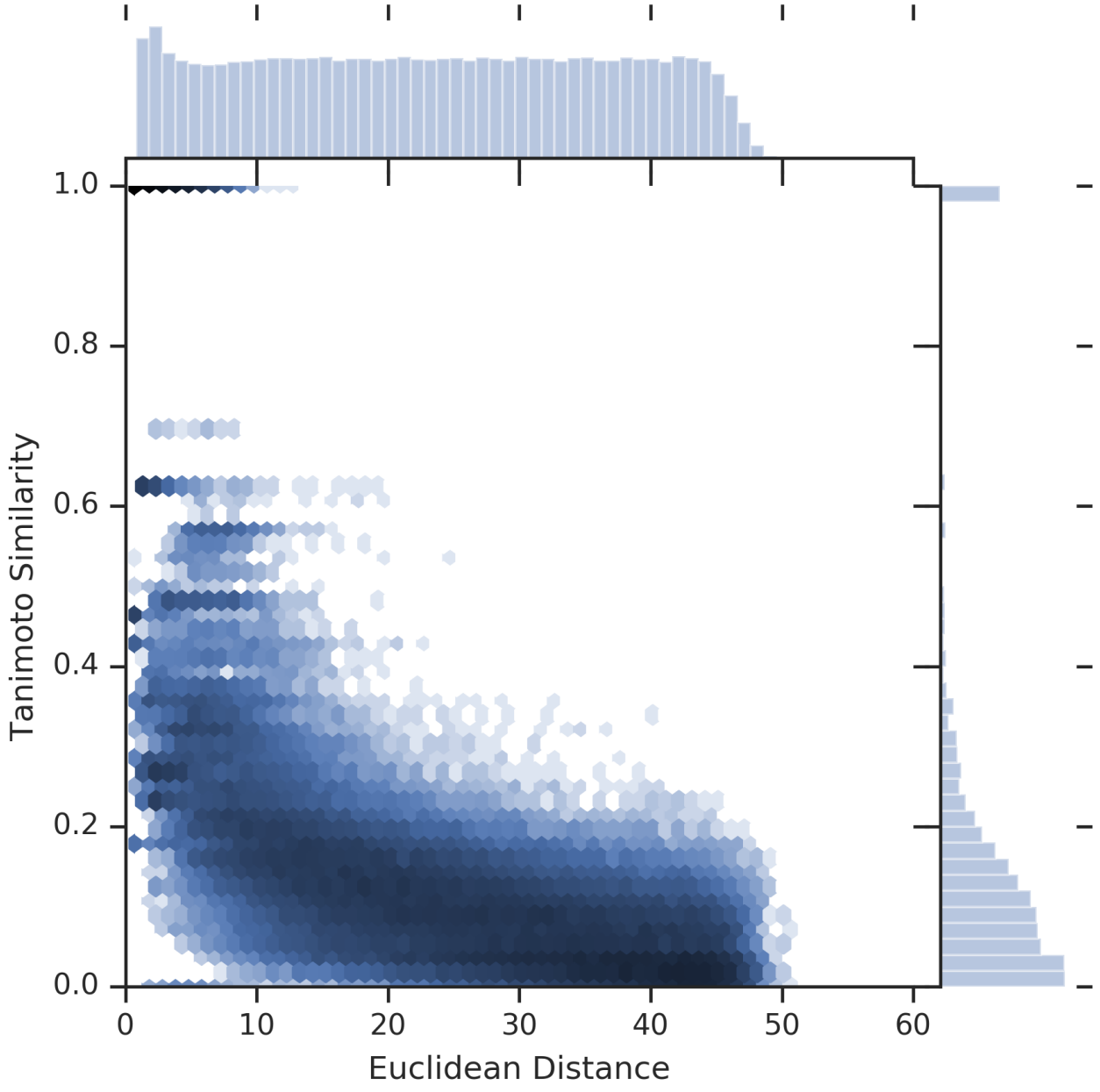
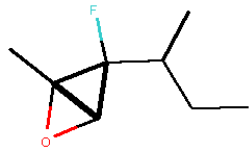
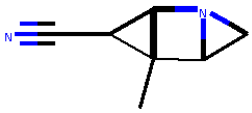


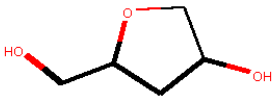
Figure A6. Tanimoto similarity between molecules decoded from original and perturbed embeddings as a function of cosine distance in the latent space (combined data from ten different original embeddings) as a function of Euclidean distance in the latent space, starting from ten different embedding locations. The density color map is logarithmic. Similarity was computed on sparse Morgan fingerprints with radius 3.



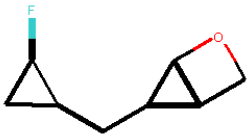
(a) CCC(C)C1(F)C2OC21C



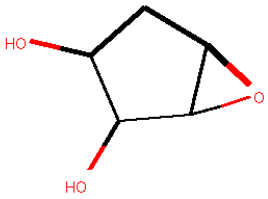
(b) CC12C(C#N)C1N1CC12



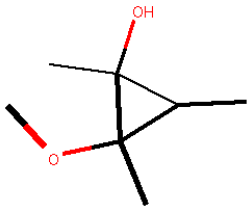
(c) OCC1CC(O)CO1



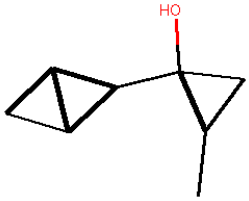
(d) FC1CC1CC1C2COC21



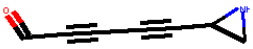
(e) OC1CC2OC2C1O



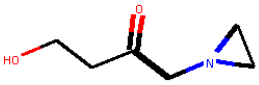
(f) COC1(C)C(C)C1(C)O



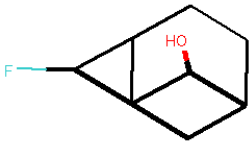
(g) CC1CC1(O)C1C2CC21



(h) O=CC#CC#CC1CN1

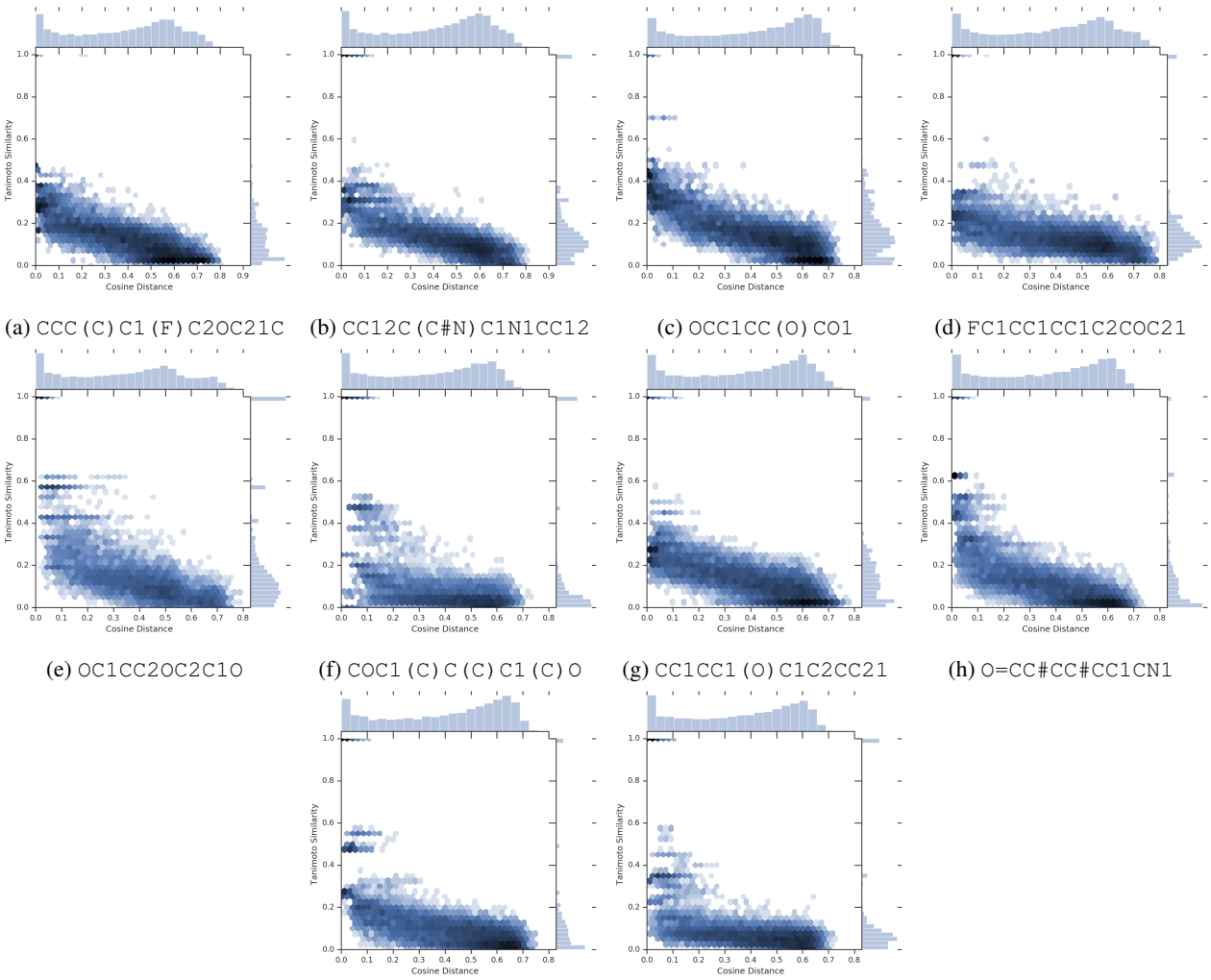


(i) O=C(CCO)CN1CC1



(j) OC1C2CCC3C(F)C13C2

Figure A7. Molecules decoded from original embeddings used for latent space exploration. A single embedding that decoded to each molecule was used as the starting point for the explorations described in Section 3.2.

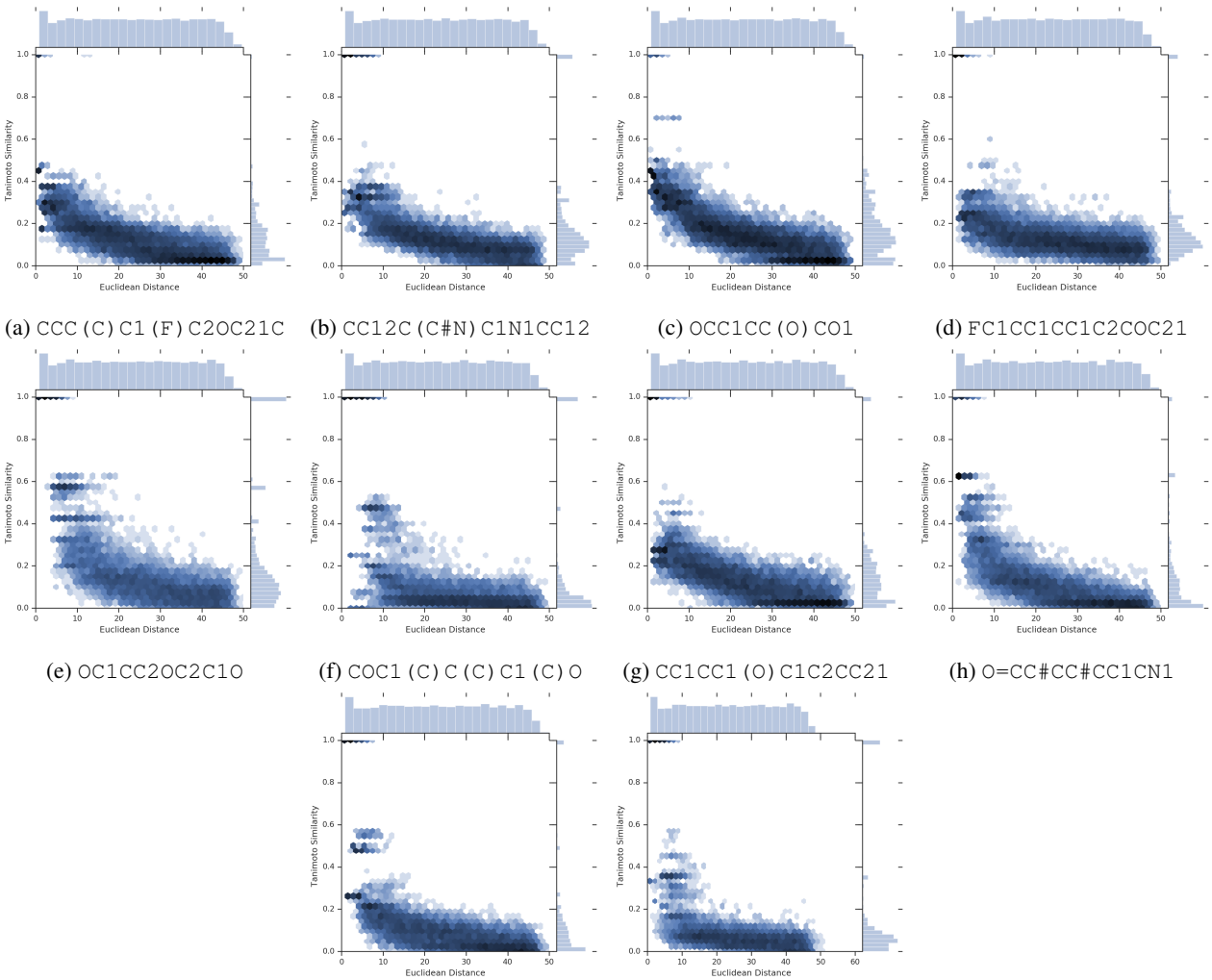


(a) CCC(C)C1(F)C2OC21C (b) CC12C(C#N)C1N1CC12 (c) OCC1CC(O)CO1 (d) FC1CC1CC1C2COC21

(e) OC1CC2OC2C1O (f) COC1(C)C(C)C1(C)O (g) CC1CC1(O)C1C2CC21 (h) O=CC#CC#CC1CN1

(i) O=C(CCO)CN1CC1 (j) OC1C2CCC3C(F)C13C2

Figure A8. Tanimoto similarity between molecules decoded from original and perturbed embeddings as a function of cosine distance in the latent space as a function of cosine distance in the latent space. The density color map is logarithmic.



(a) CCC(C)C1(F)C2OC21C (b) CC12C(C#N)C1N1CC12 (c) OCC1CC(O)CO1 (d) FC1CC1CC1C2COC21

(e) OC1CC2OC2C1O (f) COC1(C)C(C)C1(C)O (g) CC1CC1(O)C1C2CC21 (h) O=CC#CC#CC1CN1

(i) O=C(CCO)CN1CC1 (j) OC1C2CCC3C(F)C13C2

Figure A9. Tanimoto similarity between molecules decoded from original and perturbed embeddings as a function of cosine distance in the latent space as a function of Euclidean distance in the latent space. The density color map is logarithmic. Similarity was computed on sparse Morgan fingerprints with radius 3.