MOLECULAR GEOMETRY PREDICTION USING A DEEP GENERATIVE GRAPH NEURAL NETWORK

Elman Mansimov
New York University

Omar Mahmood
New York University

Seokho Kang
Sungkyunkwan University

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

ABSTRACT

A molecule's geometry, also known as its **conformation**, is one of a molecule's most important properties, determining the reactions it participates in, the bonds it forms, and the interactions it has with other molecules. Conventional conformation generation methods minimize hand-designed molecular force field energy functions that are not well correlated with the true energy function of a molecule observed in nature. They generate geometrically diverse sets of conformations, some of which are very similar to the ground-truth conformations and others of which are very different. In this paper we propose a conditional deep generative graph neural network that learns an energy function from data by directly learning to generate molecular conformations given a molecular graph. On three large scale small molecule datasets, we show that our method generates a set of conformations that on average is far more likely to be close to the corresponding reference conformations than are those obtained from conventional force field methods. Our method maintains geometrical diversity by generating conformations that are not too similar to each other. We also show that our method can be used to provide initial coordinates for conventional force field methods. On one of the evaluated datasets we show that this combination allows us to combine the best of both methods, yielding generated conformations that are on average close to ground-truth conformations with some very similar to ground-truth conformations.

1 INTRODUCTION

We consider a molecule as an undirected, complete graph $G = (V, E)$, where V is a set of vertices corresponding to atoms, and E is a set of edges representing the interactions between pairs of atoms from V. Each atom is represented as a vector $v_i \in \mathbb{R}^{d_v}$ of node features, and the edge between the i-th and j-th atoms is represented as a vector $e_{ij} \in \mathbb{R}^{d_e}$ of edge features. There are M vertices and $M(M-1)/2$ edges. We define a plausible conformation as one that may correspond to a stable configuration of a molecule. Given the graph of a molecule, the task of molecular geometry prediction is the generation of a set of plausible conformations $X_a = (x_1^a, \ldots, x_M^a)$, where $x_i^a \in \mathbb{R}^3$ is a vector of the 3-D coordinates of the i-th atom in the a-th conformation.

Molecules can transition between conformations and end up in different local minima based on the stability of the respective conformations and environmental conditions. As a result, there is more than one plausible conformation associated with each molecule; it is hence natural to formulate conformation generation as finding (local) minima of an energy function $\mathcal{F}(X, G)$ defined on a pair of molecule graph and conformation:

$$\{X_1, \ldots, X_S\} = \arg\min_X \mathcal{F}(X, G). \tag{1}$$

Alternatively, we could sample from a Gibbs distribution:

$$\{X_1, \ldots, X_S\} \sim p_{\mathcal{F}}(X|G) \tag{2}$$

$$p_{\mathcal{F}}(X|G) = \frac{1}{\zeta(G)} \exp\left\{-\mathcal{F}(X, G)\right\}, \tag{3}$$

where ζ is a normalizing constant. We use S to indicate the number of conformations we generate from each molecule. Under this view, the problem of conformation generation is decomposed into two stages. In the first stage, a computationally-efficient energy function $\mathcal{F}(X, G)$ is constructed. The second stage involves either performing optimization as in Equation 1 or sampling as in Equation 2 to generate a set of conformations from this energy function.

Traditional Energy Function Construction and Minimization/Sampling A conventional approach is to define an energy function semi-automatically. The functional form of an energy function is designed carefully to incorporate various chemical properties, whereas detailed parameters of the energy function are either computationally or experimentally estimated. Two widely used energy functions are the Universal Force Field (UFF) (Rappé et al., 1992) and the Merck Molecular Force Field (MMFF) (Halgren, 1996). Once the energy function is defined, a conventional approach is to run the minimization many times starting from different initial conformations. Due to the non-convexity of the energy function, each run is likely to end up in a unique local minimum, allowing us to collect a set of many conformations. Distance geometry (DG) (Blaney & Dixon, 1994) or its variants, such as experimental-torsion basic knowledge distance geometry (ETKDG) (Riniker & Landrum, 2015) is typically used to randomly generate an initial conformation that satisfies various geometric constraints such as lower and upper bounds on the distances between atoms. Starting from the initial conformation, an iterative optimization algorithm, such as L-BFGS (Liu & Nocedal, 1989) gradually updates the conformation until it finds a minimum of the energy function.

2 Deep Generative Model for Molecular Geometry

We propose to "learn" an energy function $\mathcal{F}(G, X)$ from a database containing many pairs of a molecule and its experimentally obtained conformation. Let $\mathcal{D} = \{(G_1, X_1^*), \ldots, (G_N, X_N^*)\}$ be a set of examples from such a database, where X_n^* is "a" ground-truth conformation, often experimentally obtained or verified. Learning an energy function can then be expressed as the following optimization problem:

$$\hat{\mathcal{F}}(G, X) = \arg\max_{\mathcal{F}} \frac{1}{N} \sum_{n=1}^{N} \underbrace{\log p_{\mathcal{F}}(X_n^* | G_n)}_{\text{(a)}}, \tag{4}$$

where $p_{\mathcal{F}}$ is a Gibbs distribution defined using $\mathcal{F}$ as in Equation 3. In other words, we can learn the energy function $\mathcal{F}$ by maximizing the log-likelihood of the data D.

2.1 Conditional Variational Graph Autoencoders

We use a conditional version of a variational autoencoder (Kingma & Welling, 2014) to model the distribution p in Equation 4 (a) (we omit the subscript $\mathcal{F}$ for brevity). This choice enables an underlying model to capture the complicated, multi-modal nature of this distribution, while allowing us to efficiently sample from this distribution. This is done by introducing a set of latent variables $Z = \{z_1, \ldots, z_M\}$, where $z_m \in \mathbb{R}^{d_z}$ and rewriting the conditional log-probability $\log p(X|G)$ as

$$\log p(X|G) = \log \int p(X|Z, G) p(Z|G) \mathrm{d}Z, \tag{5}$$

The marginal log-probability in Equation 5 is generally intractable to compute, and we instead maximize the stochastic approximation to its lower bound, as is standard practice in problems involving variational inference:

$$\log p(X|G) \geq \mathbb{E}_{Z \sim Q(Z|G,X)}[\log \underbrace{p(X|Z, G)}_{\text{(b) likelihood}}] - \mathrm{KL}(\underbrace{Q(Z|G, X)}_{\text{(c) posterior}} \| \underbrace{P(Z|G)}_{\text{(a) prior}}) \tag{6}$$

$$\approx \frac{1}{K} \sum_{k=1}^{K} \log p(X|Z^k, G) - \mathrm{KL}(Q(Z|G, X) \| P(Z|G)), \tag{7}$$

where Z^k is the k-th sample from the (approximate) posterior distribution Q above. We assume that we can compute the KL divergence analytically, for instance by constructing Q and P to be normal distributions.

Modeling the Graph using a Message Passing Neural Network We use a message passing neural network (MPNN) (Gilmer et al., 2017), a variant of a graph neural network (Scarselli et al., 2009; Bruna et al., 2014), which operates on a graph G directly and is invariant to graph isomorphism. The MPNN consists of L layers. At each layer l, we update the hidden vector $h(v_i) \in \mathbb{R}^{d_h}$ of each node and hidden matrix $h(e_{ij}) \in \mathbb{R}^{d_h \times d_h}$ of each edge using the equation

$$h^l(v_i) = \text{GRU}(h^{l-1}(v_i), J(h^{l-1}(v_i), h^{l-1}(v_{j \neq i}), h(e_{i,j \neq i})), \tag{8}$$

where J is a linear one layer neural network that aggregates the information from neighboring nodes according to its hidden vectors of respective nodes and edges. GRU is a gated recurrent network that combines the new aggregate information and its corresponding hidden vector from previous layer (Cho et al., 2014). The weights of the message passing function J and GRU are shared across the L layers of the MPNN. Details on using MPNN to parametrize prior, likelihood and posterior distributions are described in Appendix A

Training and Inference With the choice of the Gaussian latent variables z_i, we can use the reparameterization trick (Kingma & Welling, 2014) to compute the gradient of the stochastic approximation to the lower bound in Equation 7 with respect to all the parameters of the three distributions. This property allows us to train our model on a large dataset using stochastic gradient descent (SGD). However, there are two major considerations that must be made before training this model on a large molecule database. These considerations are described in Appendix B.

Learning a conditional variational autoencoder above corresponds to the first stage of conformation generation, that is, the stage of energy function construction. Once the energy function is constructed, we need to sample multiple conformations from the Gibbs distribution defined using the energy function, which is $\log P(X|G)$ in Equation 5. Our parameterization of the Gibbs distribution using a directed graphical model (Pearl, 1986) allows us to efficiently sample from this distribution. We first sample from the prior distribution, $\tilde{Z} \sim P(Z|G)$, and then sample from the likelihood distribution, $\tilde{X} \sim P(X|\tilde{Z}, G)$. In practice, we fix the output variance $\sigma_{i,j}$ of the likelihood distribution to be 1 and take the mean set $\{\mu_1, \ldots, \mu_M\}$ as a sample from the model.

2.2 Related Graph Neural Networks

Graph neural networks have recently been used to obtain latent representations of molecules and to generate molecules. One such approach involves reducing the entire molecular graph to a single vector latent variable (Simonovsky & Komodakis (2018)). This method requires the network to learn how many atoms to generate, and which of and how these atoms are connected. Since we do not need to generate molecules and always have knowledge of the complete molecular graph, we choose not to reduce our graph to a single vector latent variable. In another work, the authors propose a model in which both the edges and nodes are given latent representations (Kipf et al. (2018)). We choose not to use edge latent representations as molecular geometry is ultimately defined by the positions of the atoms, and is only related to the bonds between atoms through the bonds' influence on the positions of the atoms. Therefore we only learn a linear embedding for the edges. Junction tree variational autoencoders have also been used for molecular graph generation (Jin et al. (2018)). This method also involves a producing a latent representation of a graph, as well as for a junction tree containing predefined chemical groups as elements. If such groups have similar geometry in different situations, it may be worth investigating a variation of this method in combination with our method in future work. This would ensure that the network would not have to learn, for example, the positions of atoms in a benzene ring.

3 Experimental Setup

Data We experimentally verify the effectiveness of the proposed approach using three databases of molecules: QM9 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014), COD (Gražulis et al., 2012) and CSD (Groom et al., 2016). These datasets are selected as they possess distinct properties from each other, which allows us to carefully study various aspects of the proposed approach. Although, there is an overlap between COD and CSD databases, since both of these databases were based on published crystallography data. In general, COD/CSD databases contain larger molecules compared to QM9 database. More details describing characteristics of databases are described in Appendix C.

Dataset		ETKDG + Force Field UFF	ETKDG + Force Field MMFF	CVGAE	CVGAE + Force Field MMFF
QM9	success per test set	96.440%	96.440%	100%	99.760%
	success per molecule	98.725%	98.725%	100%	98.684%
	mean	0.425	0.415	0.390	0.367
	std. dev.	0.176	0.189	0.017	0.074
	best	0.126	0.092	0.325	0.115
COD	success per test set	99.133%	99.133%	100%	95.367%
	success per molecule	99.627%	99.627%	100%	99.071%
	mean	1.389	1.358	1.331	1.656
	std. dev.	0.407	0.415	0.099	0.425
	best	0.429	0.393	1.206	0.635
CSD	success per test set	97.400%	97.400%	100%	99.467%
	success per molecule	99.130%	99.130%	100%	97.967%
	mean	1.537	1.488	1.506	1.833
	std. dev.	0.421	0.418	0.115	0.434
	best	0.508	0.478	1.343	0.784

Table 1: Number of successfully processed molecules in the test set (success per test set ↑), number of successfully generated conformations out of 100 (success per molecule ↑), median of mean RMSD (mean ↓), median of standard deviation of RMSD (std. dev. ↓) and median of best RMSD (best ↓) per molecule on QM9, COD and CSD datasets.

Evaluation In principle, the quality of the sampled conformations should be evaluated based on their molecular energies, for instance by quantum methods such as density functional theory (DFT) (Hautier et al., 2012), which is often more accurate than force field methods (Kanal et al., 2017). However, the computational complexity of the DFT calculation is superlinear with respect to the number of electrons in a molecule, and so is often impractical (Ratcliff et al., 2017). Instead, we follow prior work on conformation generation (Hawkins, 2017) and evaluate the baselines and proposed method using the root-mean-square deviation (**RMSD**) of the heavy atoms between a ground-truth conformation and a predicted conformation. RMSD is defined as the square root of the mean square error and is fast and simple to calculate.

Baselines and Proposed Approach As a point of reference, we minimize a force field starting from a conformation created using ETKDG (Riniker & Landrum, 2015). We test both UFF and MMFF, and respectively call the resulting approaches **ETKDG+UFF** and **ETKDG+MMFF**. We use the implementations in RDKit [1] with the default hyperparameters. There are two modes of inference with the proposed approach. The first approach is to sample from a trained conditional variational graph autoencoder by first sampling from the prior distribution and taking the mean vectors from the likelihood distribution; we refer to this as **CVGAE**. We can then use these samples further as initializations of MMFF minimization; we refer to this as **CVGAE+MMFF**. The latter approach can be thought of as a trainable approach to initializing a conformation in place of DG or ETKDG. Details of hyperparameters of proposed model are described in Appendix D

Results When evaluating each method, we first sample 100 conformations per molecule for each method in the test set. We can make several observations from Table 1. First, compared to other methods, our proposed CVGAE always succeeds at generating the specified number of conformations for any of the molecules in the test set. Since all other evaluated approaches were unsuccessful at generating at least one conformation for a very small number of test molecules, we report results for the molecules for which all evaluated methods generated at least one conformation. We report the *median* of the mean of the RMSD, the *median* of the standard deviation of the RMSD and the *median* of the best (lowest) RMSD among all generated conformations for each test molecule. Across all three datasets, every evaluated method achieves roughly the same median of the mean RMSD. More importantly, the standard deviation of the RMSD achieved by CVGAE is *significantly* lower than that achieved by ETKDG + Force Field. After the initial generation stage, conforma-

[1] `https://github.com/rdkit/rdkit/`, version 2018.09.1

tions are usually further evaluated and optimized by running the computationally expensive DFT optimization. Reducing the standard deviation can lower the number of conformations on which DFT optimization has to be run in order to achieve a valid conformation. On the other hand, the best RMSD achieved by ETKDG + UFF/MMFF methods is lower than that achieved by CVGAE. Using MMFF initialized by CVGAE (CVGAE + MMFF) instead of ETKDG (ETKDG + MMFF) improves the mean results on the QM9 dataset for CVGAE, and yields a lower standard deviation and similar best RMSD compared to ETKDG + MMFF. Unfortunately, CVGAE + MMFF worsens the results achieved by CVGAE alone on the COD and CSD datasets.

We also report the diversity of conformations generated by all evaluated methods in Table 2 (in Appendix). Diversity is measured by calculating the mean and standard deviation of the pairwise RMSD between each pair of generated conformations per molecule. Overall, we can see that despite having a smaller median of standard deviation of RMSD between generated conformations and ground-truth conformations, CVGAE doesn't collapse to generating extremely similar conformations. Although, CVGAE generates relatively less diverse samples compared to ETKDG + MMFF baseline on all datasets. The conformations of molecules generated by CVGAE + MMFF are less diverse on the QM9 dataset and more diverse on COD/CSD datasets compared to ETKDG + MMFF baseline. More experimental results are described in Appendix E.

Discussion In this paper, we proposed a deep generative model for generating molecular geometry (conformation) given molecule graph. In contrast to traditional methods, the energy function or probability distribution of molecule is estimated directly from data using the latest techniques from representation learning on graphs and variational inference. We show that conformations generated by our model are on average far more likely to be close to the ground-truth conformation compared to those generated by conventional force field methods without generating geometrically similar conformations. On QM9 dataset, we show that the best of both methods can be combined by using the conformations generated by the deep generative graph neural network as an initialization to the force field method. Further work is necessary to investigate this combination as well as building larger generative models on COD/CSD datasets.

REFERENCES

Jeffrey M. Blaney and J. Scott Dixon. Distance geometry in molecular modeling. *Rev. Comput. Chem.*, 1994.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1263–1272, 2017.

Saulius Gražulis, Adriana Daškevič, Andrius Merkys, Daniel Chateigner, Luca Lutterotti, Miguel Quiros, Nadezhda R. Serebryanaya, Peter Moeck, Robert T. Downs, and Armel Le Bail. Crystallography open database (COD): An open-access collection of crystal structures and platform for world-wide collaboration. *Nucleic Acids Res.*, 40(D1):D420–D427, 2012.

Colin R. Groom, Ian J. Bruno, Matthew P. Lightfoot, and Suzanna C. Ward. The cambridge structural database. *Acta Crystallogr. Sect. B-Struct. Sci.Cryst. Eng. Mat.*, 72(2):171–179, 2016.

Thomas A. Halgren. Merck molecular force field. I. basis, form, scope, parameterization, and performance of MMFF94. *J. Comput. Chem.*, 17(5-6):490–519, 1996.

Geoffroy Hautier, Anubhav Jain, and Shyue Ping Ong. From the computer to the laboratory: Materials discovery and design using first-principles calculations. *J. Mater. Sci.*, 47(21):7317–7340, 2012.

Paul C. D. Hawkins. Conformation generation: The state of the art. *J. Chem. Inf. Model.*, 2017.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2323–2332, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/jin18a.html.

Ilana Y. Kanal, John A. Keith, and Geoffrey R. Hutchison. A sobering assessment of small-molecule force field methods for low energy conformer predictions. *Int. J. Quantum Chem.*, 2017.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *J. Comput.-Aided Mol. Des.*, 30(8):595–608, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In Jennifer Dy and Andreas Kraus (eds.), *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 2018. URL http://proceedings.mlr.press/v80/kipf18a.html.

Greg Landrum. Rdkit: Open-source cheminformatics. URL http://www.rdkit.org. (accessed December 18, 2018).

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1-3):503–528, 1989.

Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artif. Intell.*, 29(3):241–288, 1986.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. Data*, 1:140022:1–7, 2014.

Bharath Ramsundar, Peter Eastman, Karl Leswing, Patrick Walters, and Vijay Pande. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019.

Anthony K. Rappé, Carla J. Casewit, K. S. Colwell, William A. Goddard III, and W. M. Skiff. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.*, 114(25):10024–10035, 1992.

Laura E. Ratcliff, Stephan Mohr, Georg Huhs, Thierry Deutsch, Michel Masella, and Luigi Genovese. Challenges in large scale quantum mechanical calculations. *Wiley Interdiscip. Rev.-Comput. Mol. Sci.*, 7(1):e1290, 2017.

Sereina Riniker and Gregory A. Landrum. Better informed distance geometry: Using what we know to improve conformation generation. *J. Chem. Inf. Model.*, 2015.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *J. Chem. Inf. Model.*, 52(11):2864–2875, 2012.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Netw.*, 20(1):61–80, 2009.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *CoRR*, abs/1802.03480, 2018. URL http://arxiv.org/abs/1802.03480.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15: 1929–1958, 2014.

A ARCHITECTURE OF THE CONDITIONAL VARIATIONAL GRAPH AUTOENCODER

A diagram of the model is shown in Figure 1.

Prior Parameterization We use the MPNN described in 2.1 to model the prior distribution $P(Z|G)$ in Equation 6 (a). We initialize $h^0(v_i)$ and $h(e_{ij})$ in Equation 8 as linear transformations of the feature vectors v_i and e_{ij} of the nodes and edges respectively:

$$h^0(v_i) = U_{\text{node}}^{\text{prior}} v_i; \quad h(e_{ij}) = U_{\text{edge}}^{\text{prior}} e_{ij}, \tag{9}$$

where $U_{\text{node}}^{\text{prior}}$ and $U_{\text{edge}}^{\text{prior}}$ are matrices representing the linear transformations for the nodes and edges respectively. The final hidden vector $h^L(v_i)$ of each node is passed through a two layer neural network with hidden size d_f, whose output $\tilde{h}^L(v_i)$ is transformed into the mean and variance vectors of a Normal distribution with a diagonal covariance matrix:

$$\mu_i = W_\mu^{\text{prior}} \tilde{h}^L(v_i) + b_\mu^{\text{prior}}; \tag{10}$$

$$\sigma_i^2 = \exp\left\{ W_\sigma^{\text{prior}} \tilde{h}^L(v_i) + b_\sigma^{\text{prior}} \right\}, \tag{11}$$

where W_μ^{prior} and W_σ^{prior} are the weight matrices and b_μ^{prior} and b_σ^{prior} are the bias terms of the transformations. These are used to form the prior distribution:

$$\log P(Z|G) = \sum_{i=1}^{N} \sum_{j=1}^{3} -\frac{(\mu_{i,j} - z_{i,j})^2}{2\sigma_{i,j}^2} - \log \sqrt{2\pi\sigma_{i,j}^2}, \tag{12}$$

where $\mu_{i,j}$ and $\sigma_{i,j}^2$ are the j-th components of the mean and variance vectors respectively. In other words, we parameterize the prior distribution as a factorized Normal distribution factored over the vertices and the dimensions in the 3-D coordinate.

Likelihood Parameterization We use a similar MPNN to model the likelihood distribution, $P(X|Z, G)$ in Equation 6 (b). The only difference is that this distribution is conditioned not only on the molecular graph $G = (V, E)$ but also on the latent set $Z = \{z_1, \ldots, z_M\}$. We incorporate the latent set Z by adding the linear transformation of the node feature vector v_i to its corresponding latent variable z_i. This result is used to initialize the hidden vector:

$$h^0(v_i) = U_{\text{node}}^{\text{likelihood}} v_i + z_i; \quad h(e_{ij}) = U_{\text{edge}}^{\text{likelihood}} e_{ij}, \tag{13}$$

where $U_{\text{node}}^{\text{likelihood}}$ and $U_{\text{edge}}^{\text{likelihood}}$ are matrices representing the linear transformations for the nodes and edges respectively. From there on, we run neural message passing as in Eqs. (8–11), with a new set of parameters, $\theta_{\text{likelihood}}$, $W_\mu^{\text{likelihood}}$, $b_\mu^{\text{likelihood}}$, $W_\sigma^{\text{likelihood}}$ and $b_\sigma^{\text{likelihood}}$. The final mean and variance vectors are now three dimensional, representing the 3-D coordinates of each atom, and we can compute the log-probability of the coordinates using Equation 12.

Posterior Parameterization As computing the exact posterior $P(Z|G, X)$ is intractable, we resort to amortized inference using a parameterized, approximate posterior $Q(Z|G, X)$ in Equation 6 (c). We use a similar approach to our parameterization of the prior distribution above. However, we replace the input to the MPNN with the concatenation of an edge feature vector e_{ij} and the corresponding distance (proximity) matrix $D(X^*)$ of the ground-truth 3-D conformation X^*:

$$h(e_{ij}) = U_{\text{edge}}^{\text{posterior}} \begin{bmatrix} e_{ij} \\ D(x_i^*) \end{bmatrix}. \tag{14}$$

With a new set of parameters, $\theta_{\text{posterior}}$, $W_\mu^{\text{posterior}}$, $b_\mu^{\text{posterior}}$, $W_\sigma^{\text{posterior}}$ and $b_\sigma^{\text{posterior}}$, and similarly to the process in Eqs. (8–11), the MPNN outputs the mean and diagonal covariance of a normal distribution for each latent variable z_i. Linear weight embeddings of nodes U_{node} are shared between prior, likelihood and posterior.

B TRAINING THE CONDITIONAL VARIATIONAL GRAPH AUTOENCODER

(1) Post-Alignment Likelihood An important property of conformation generation over a usual problem of regression is that we must take into account rotation and translation. Let R be an alignment function that takes as input a a target conformation and a predicted conformation, aligns the reference conformation to the predicted conformation and returns the aligned reference conformation. $\hat{X} = R(X, X^*)$ is the conformation obtained by rotating and translating the reference conformation X^* to have the smallest distance to the predicted conformation X according to a predefined metric such as root-mean-square deviation (RMSD):

$$\text{RMSD}(\hat{X}, X^*) = \sqrt{\frac{1}{M}\sum_{i=1}^{M} \|\hat{x}_i - x_i^*\|^2}. \tag{15}$$

This alignment function R is selected according to the problem at hand, and we present below its use in a general form without exact specification.

We implement this invariance to rotation and translation by parameterizing the output of the likelihood distribution above to be aligned to the target molecule. That is,

$$\log p(X|G, Z) = \sum_{i=1}^{M}\sum_{j=1}^{3} -\frac{(\mu_{i,j} - \hat{x}_{i,j}^*)^2}{2\sigma_{i,j}^2} - \log\sqrt{2\pi\sigma_{i,j}^2}, \tag{16}$$

where $\hat{x}_i^*$ is the coordinate of the i-th atom aligned to the mean conformation $\{\mu_1, \ldots, \mu_N\}$. That is,

$$\{\hat{x}_1^*, \ldots, \hat{x}_M^*\} = R(\{\mu_1, \ldots, \mu_M\}, X^*). \tag{17}$$

In other words, we rotate and translate the reference conformation X^* to be best aligned to the predicted conformation (or its mean) before computing the log-probability. This encourages the model to assign high probability to a conformation that is easily aligned to the reference conformation X^*, which is precisely the goal of maximum log-likelihood.

(2) Unconditional Prior Regularization The second term in the lower bound in Equation 6, which is the KL divergence between the approximate posterior and prior, does not have a point minimum but an infinitely long valley consisting of minimum values. Consider the KL divergence between two univariate Normal distributions:

$$\text{KL}(\mathcal{N}(\mu_1, \sigma_1^2)\|\mathcal{N}(\mu_2, \sigma_2^2)) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. \tag{18}$$

When both distributions are shifted by the same amount, the KL divergence remains unchanged. This could lead to a difficulty in optimization, as the means of the posterior and prior distributions could both diverge.

In order to prevent this pathological behavior, we introduce an unconditional prior distribution $P(Z)$ which is a factorized Normal distribution:

$$P(Z) = \prod_{i=1}^{M} \mathcal{N}(z_i|0, I), \tag{19}$$

where $\mathcal{N}$ computes a Normal probability density, and I is a $d_z \times d_z$ identity matrix. We minimize the KL divergence between the original prior distribution $P(Z|G)$ and this unconditional prior distribution $P(Z)$ in addition to maximizing the lowerbound, leading to the following final objective function for each molecule:

$$\mathcal{L} = \log p(X|Z^1, G) - \text{KL}(Q(Z|G, X)\|P(Z|G)) - \alpha \cdot \text{KL}(P(Z|G)\|P(Z)), \tag{20}$$

where we assume $K = 1$ and introduce a coefficient $\alpha \geq 0$.

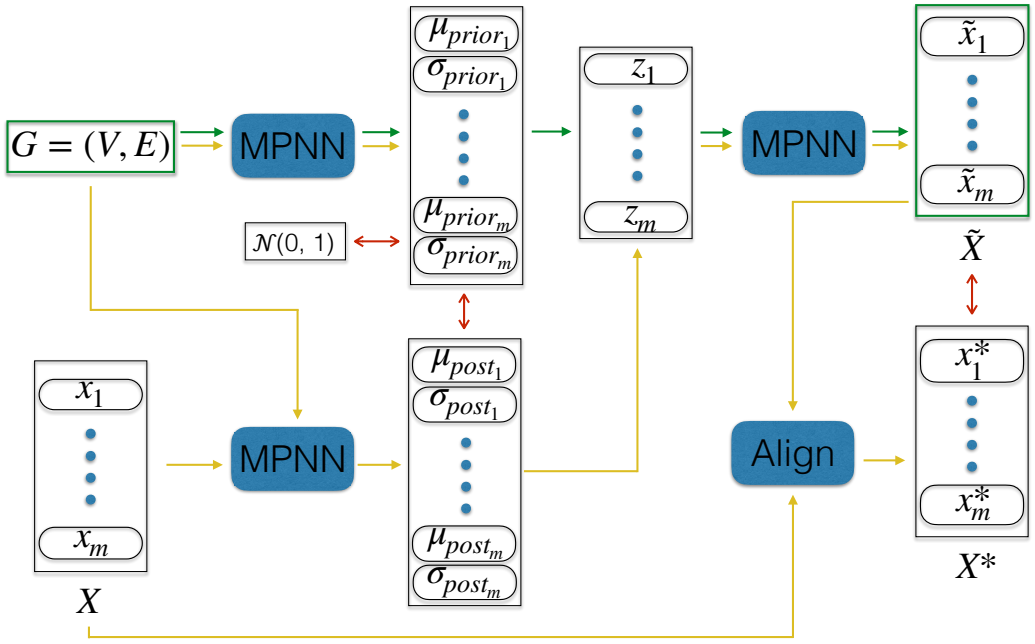


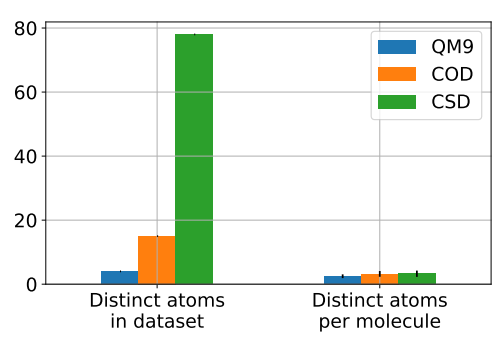
Figure 1: A diagram of the CVGAE model. Yellow arrows show the training process, green arrows show the testing process and double headed red arrows show directly interacting terms in the loss function. Clear, rounded boxes are vectors. The input and output at test time are in highlighted, green boxes.

C DATA

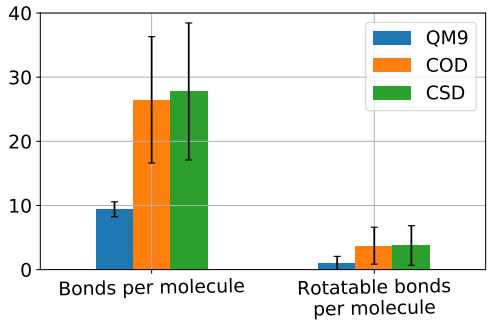
QM9 The filtered QM9 dataset contains 133,015 molecules, each of which contains up to 9 heavy atoms of types `C`, `N`, `O` and `F`. Each molecule is paired with a ground-truth conformation obtained by optimizing the molecular geometry with density functional theory (DFT) at the B3LYP/6-31G(2df,p) level of theory, which implies that these ground-truth conformations may not necessarily correspond to the lowest energy configurations of the molecules. We hold out separate 5,000 and 5,000 randomly selected molecules as validation and test sets, respectively.

COD We use the organic part of the COD dataset. We further filter out any molecule that contains more than 50 heavy atoms of types `B`, `C`, `N`, `O`, `F`, `Si`, `P`, `S`, `Cl`, `Ge`, `As`, `Se`, `Br`, `Te` and `I`. This results in 66,663 molecules, out of which we hold out separate 3,000 and 3,000 randomly selected ones respectively for validation and test purposes. Ground-truth conformations are voluntarily contributed to the dataset and are often determined either experimentally or by DFT (Hautier et al., 2012).

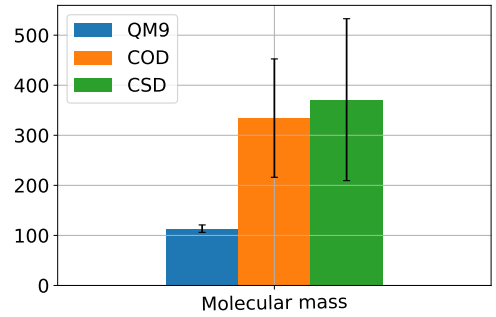
CSD Similarly to COD, we remove any molecule that contains more than 50 heavy atoms, resulting in a total of 236,985 molecules. This dataset (Groom et al., 2016) contains organic and metal-organic crystallographic structures which have been verified experimentally. The atom types in this dataset are `S`, `N`, `P`, `Be`, `Tc`, `Xe`, `Br`, `Rh`, `Os`, `Zr`, `In`, `As`, `Mo`, `Dy`, `Nb`, `La`, `Te`, `Th`, `Ga`, `Tl`, `Y`, `Cr`, `F`, `Fe`, `Sb`, `Yb`, `Tb`, `Pu`, `Am`, `Re`, `Eu`, `Hg`, `Mn`, `Lu`, `Nd`, `Ce`, `Ge`, `Sc`, `Gd`, `Ca`, `Ti`, `Sn`, `Ir`, `Al`, `K`, `Tm`, `Ni`, `Er`, `Co`, `Bi`, `Pr`, `Rb`, `Sm`, `O`, `Pt`, `Hf`, `Se`, `Np`, `Cd`, `Pd`, `Pb`, `Ho`, `Ag`, `Mg`, `Zn`, `Ta`, `V`, `B`, `Ru`, `W`, `Cl`, `Au`, `U`, `Si`, `Li`, `C` and `I`. We hold out separate 3,000 and 3,000 randomly selected molecules for validation and test purposes respectively. See Figure 2 for more detailed characteristics of datasets.



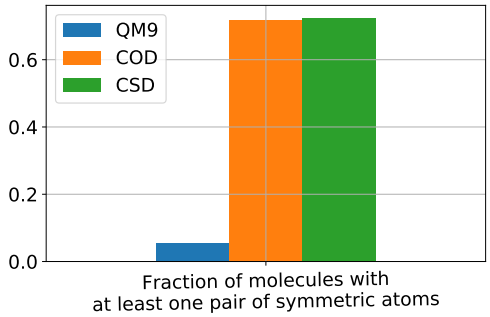
(a) Diversity of atoms overall and per molecule



(b) Number and rotatability of bonds per molecule



(c) Molecular mass per molecule



(d) Proportion of molecules with symmetry

Figure 2: Dataset Characteristics: information regarding the atoms, bonds, molecular mass and symmetry of molecules in each dataset.

D HYPERPARAMETERS

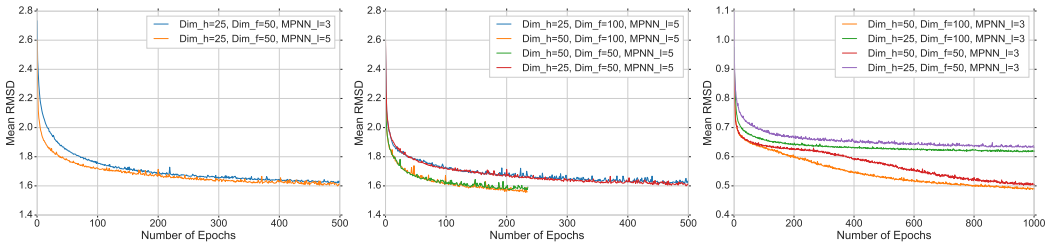
We build one conditional variational graph autoencoder for each dataset. We use $d_h = 50$ hidden units at each layer of neural message passing (Eq. 8) in each of the three MPNNs corresponding to the prior, likelihood and posterior distributions. We use $d_f = 100$ in the two layer neural network that comes after the MPNN. As described earlier, we fix the variance of the output in the likelihood distribution to 1. We use $L = 3$ layers per network for QM9 and $L = 5$ layers per network for COD and CSD. We chose these hyperparameter values by carrying out a grid-search and choosing the values that had the best performance on the validation set. For all models, we use dropout (Srivastava et al., 2014) at each layer of the neural network that comes after the MPNN with a dropout rate of 0.2 to regularize learning. We set the coefficient α in Equation 20 to 10^{-5}. We train each model using Adam (Kingma & Ba, 2015) with a fixed learning rate of 3×10^{-4}. All models were trained with a batch size of 20 molecules on 1 Nvidia GPU with 12 GB of RAM.

D.1 HYPERPARAMETER SEARCH

Below are the hyperparameters we tried for the QM9 and COD datasets. We picked the hyperparameters to ensure that a model trained with a batch size of 20 molecules could fit on 1 GPU with 12 GB of RAM.

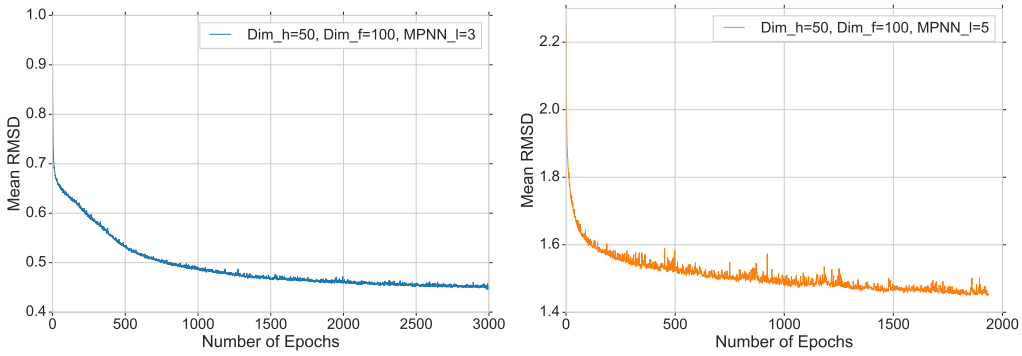
We experimented with the following values of hyperparameters on the QM9 dataset: $d_h = [25, 50]$, $d_f = [50, 100]$. The number of MPNN layers L was fixed to 3 according to previous preliminary experiments. On Figure 3c we can see that the model with $d_h = 50$ and $d_f = 100$ significantly outperforms models with a smaller number of hidden units.

On the COD dataset we experimented with the following values of hyperparameters: $d_h = [25, 50]$, $d_f = [50, 100]$ and number of MPNN layers $L = [3, 5]$. In Figure 3a we can see that the model with 5 MPNN layers slightly outpeforms the model with 3 MPNN layers. Similarly to the QM9



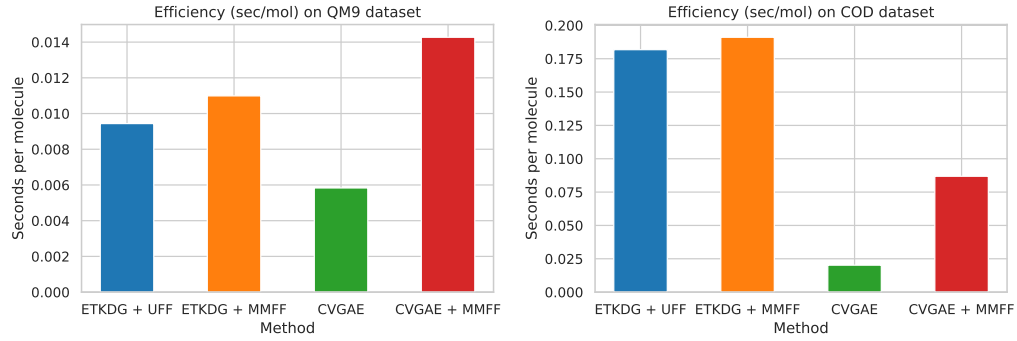
(a) COD dataset (b) COD dataset (c) QM9 dataset

Figure 3: Investigation of number of different hyperparameters on QM9 and COD datasets over different number of epochs. Mean RMSD over number of epochs of best performing model on valid set of corresponding dataset. Mean RMSD was calculated given 10 conformations per molecule.



(a) QM9 dataset (b) COD dataset

Figure 4: Performance of the best performing model over the number of epochs. Mean RMSD over number of epochs of best performing model on valid set of corresponding dataset. Mean RMSD was calculated given 10 conformations per molecule.



(a) QM9 dataset (b) COD dataset

Figure 5: Computational efficiency of various approaches on QM9 and COD datasets

dataset, we can see in Figure 3b that a larger number of hidden units results in significantly faster convergence and better performance.

We selected the model with the best hyperparameter values given by our grid-search. Figure 4 shows the RMSD of this model on the validation set as a function of number of epochs on the QM9 and COD datasets.

Dataset		ETKDG + MMFF	CVGAE	CVGAE + MMFF
QM9	mean	0.400	0.106	0.238
	std. dev.	0.254	0.061	0.209
COD	mean	1.148	0.239	1.619
	std. dev.	0.699	0.181	0.537
CSD	mean	1.244	0.567	1.665
	std. dev.	0.733	0.339	0.177

Table 2: Conformation Diversity. Mean and std. dev. represents the corresponding mean and standard deviation of pairwise RMSD between at most 100 generated conformations per molecule.

E EXPERIMENTAL RESULTS (CONT.)

The computational efficiency of each of the evaluated approaches on the QM9 and COD datasets is shown in Figure 5. For consistency, we generated one conformation for one molecule at a time using each of the evaluated methods on an Intel(R) Xeon(R) E5-2650 v4 CPU. On the QM9 dataset, CVGAE is $2\times$ more efficient than ETKDG + UFF/MMFF, while CVGAE + MMFF is slightly slower than ETKDG + UFF/MMFF. On the COD dataset, which contains a larger number of atoms per molecule, CVGAE is almost $10\times$ as fast as ETKDG + UFF/MMFF, while CVGAE + MMFF is about $2\times$ as fast as ETKDG + UFF/MMFF. This shows that CVGAE scales much better than the baseline ETKDG + UFF/MMFF methods as the size of the molecule grows.

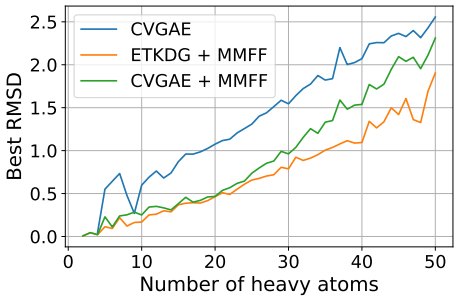
Figures 6 and 7 visualize the median, standard deviation and best RMSD results as a function of the number of heavy atoms in a molecule on the QM9 and COD/CSD datasets respectively. For all approaches, we can see that the best and median RMSD both increase with the number of heavy atoms. The standard deviation of the median RMSD for CVGAE and CVGAE + MMFF is lower than that for ETKDG + MMFF across molecules of almost all sizes. The standard deviation of the best RMSD is slightly higher for CVGAE and CVGAE + MMFF than for ETKDG + MMFF on molecules with at most 12 atoms, but is lower for larger atoms, particularly for CVGAE. Overall, CVGAE yields a lower or similar median RMSD compared to ETKDG + CVGAE across molecules of all sizes but a lower standard deviation, whereas ETKDG + MMFF provides a lower best RMSD particularly for larger molecules observed in the COD/CSD datasets.

Figures 8 and 9 qualitatively compare the results of CVGAE against MMFF and CVGAE + MMFF against CVGAE respectively. For each dataset, each figure shows the three molecules for which the first method in each figure outperforms the second method by the greatest amount, and the three molecules for which the second method outperforms the first by the greatest amount. The reference molecules are shown alongside the conformations resulting from each of the methods for comparison.

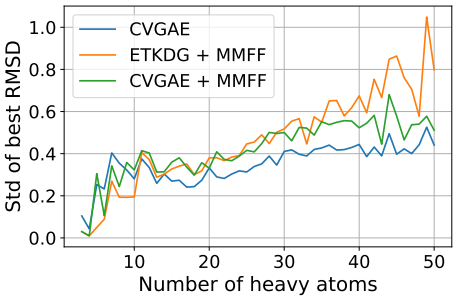
We can see some general trends from both these figures. The conformations produced by the neural network are qualitatively much more similar to the reference in the case of the QM9 dataset than in the cases of the COD and CSD datasets. In the case of the COD and CSD datasets, the CVGAE predictions appear to be squashed or compressed in comparison to the reference molecules. For example, in almost every case we can see the absence of visible rings and the absence of bonds protruding from the lengthwise dimension of the molecule. At the same time we can see that on COD and CSD, CVGAE does better than ETKDG + MMFF in cases where ETKDG + MMFF creates loops and protrusions in the wrong places.

E.1 MOLECULAR FEATURES

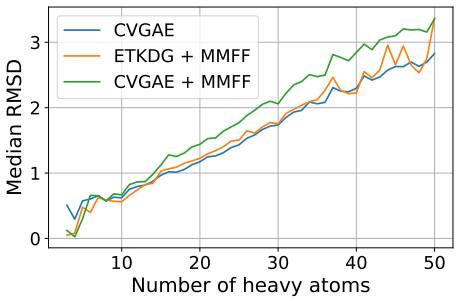
To represent molecules as graph-structured data, each of the nodes and edges in the molecule is represented using the features described in Tables 3 and 4, according to related literature (Kearnes et al., 2016; Ramsundar et al., 2019; Gilmer et al., 2017). We only consider heavy atoms, and do not consider hydrogen atoms as explicit nodes *i.e.* hydrogen atoms are represented as part of the input features and their coordinates are not predicted by the neural network. In Table 4, the first four



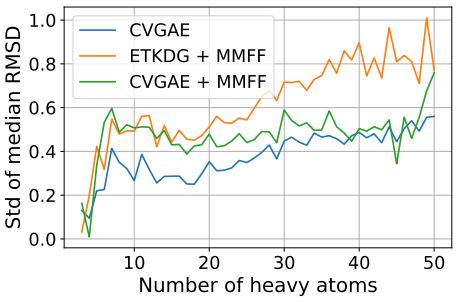
(a) Mean of best RMSD



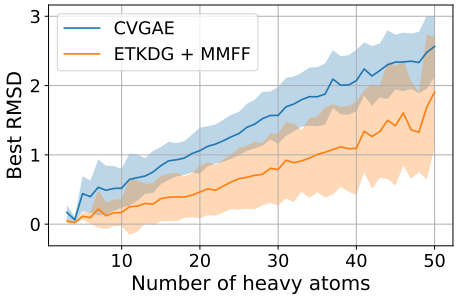
(b) St. dev. of best RMSD



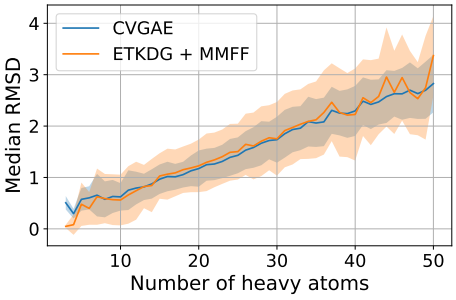
(c) Mean of median RMSD



(d) St. dev. of median RMSD

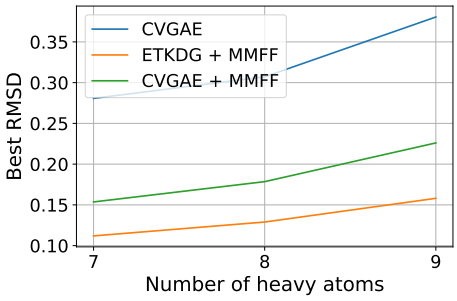


(e) Best RMSD with uncertainty bounds

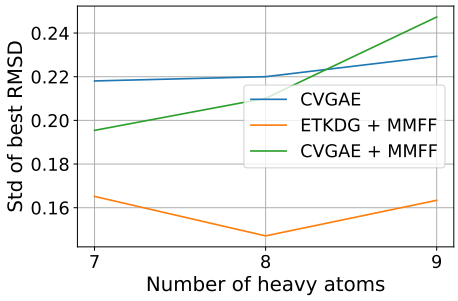


(f) Median RMSD with uncertainty bounds

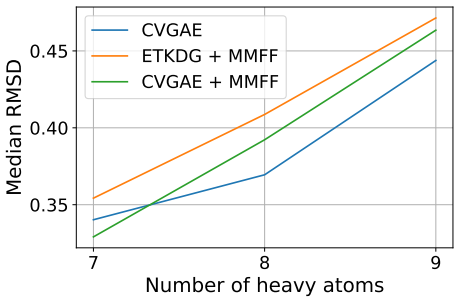
Figure 6: This figure shows the means and standard deviations of the best and median rmsds on the union of COD and CSD datasets as a function of number of heavy atoms. The molecules were grouped by number of heavy atoms, and the mean and standard deviation of the median and best RMSDs were calculated for each group to obtain these plots. Groups at the left hand side of the graph with less than 1% of the mean number of molecules per group were omitted.



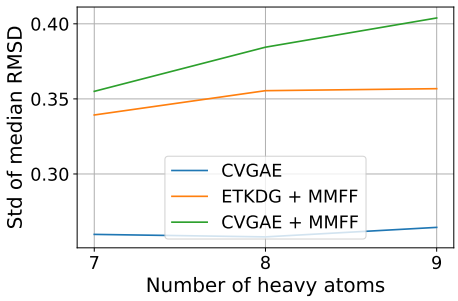
(a) Mean of best RMSD



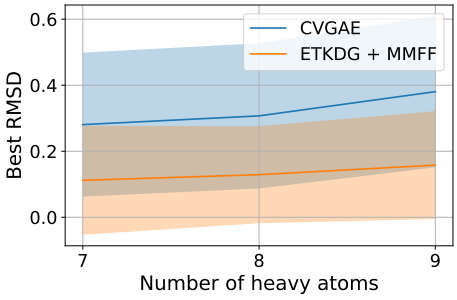
(b) St. dev. of best RMSD



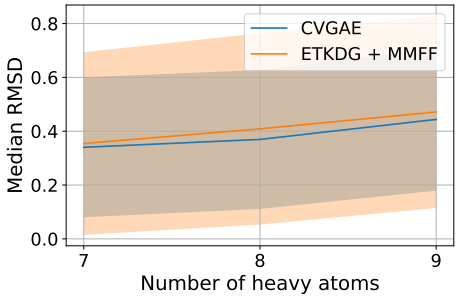
(c) Mean of median RMSD



(d) St. dev. of median RMSD

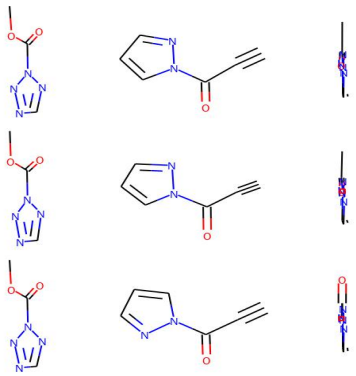


(e) Best RMSD with uncertainty bounds

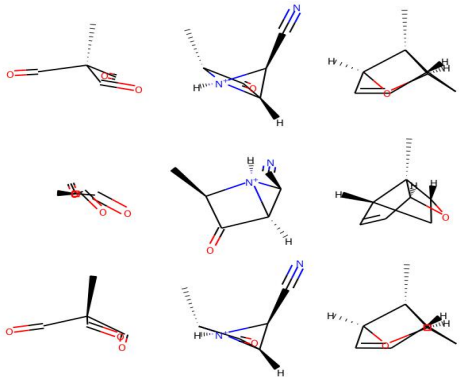


(f) Median RMSD with uncertainty bounds

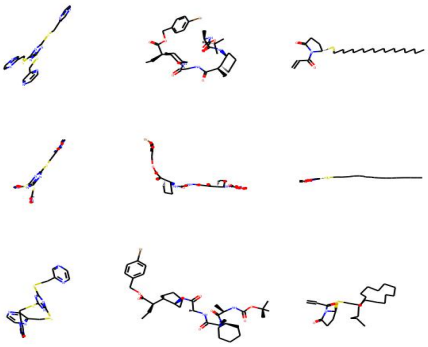
Figure 7: This figure shows the means and standard deviations of the best and median RMSDs on the QM9 dataset as a function of number of heavy atoms. The molecules were grouped by number of heavy atoms, and the mean and standard deviation of the median and best RMSDs were calculated for each group to obtain these plots. Groups at the left hand side of the graph with less than 1% of the mean number of molecules per group were omitted.



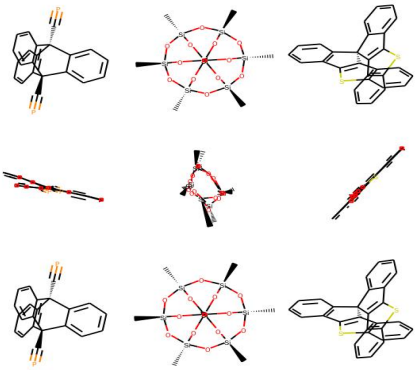
(a) QM9 greatest difference in favour of neural network predictions



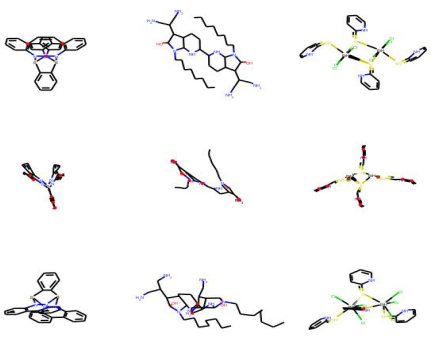
(b) QM9 greatest difference in favour of ETKDG + MMFF predictions



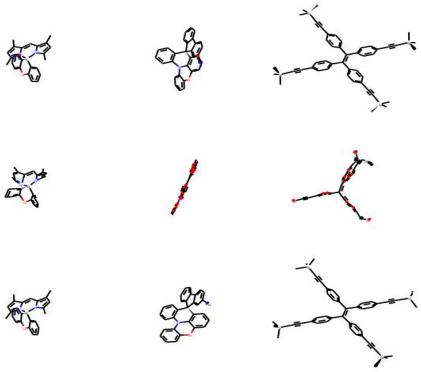
(c) COD greatest difference in favour of neural network predictions



(d) COD greatest difference in favour of ETKDG + MMFF predictions

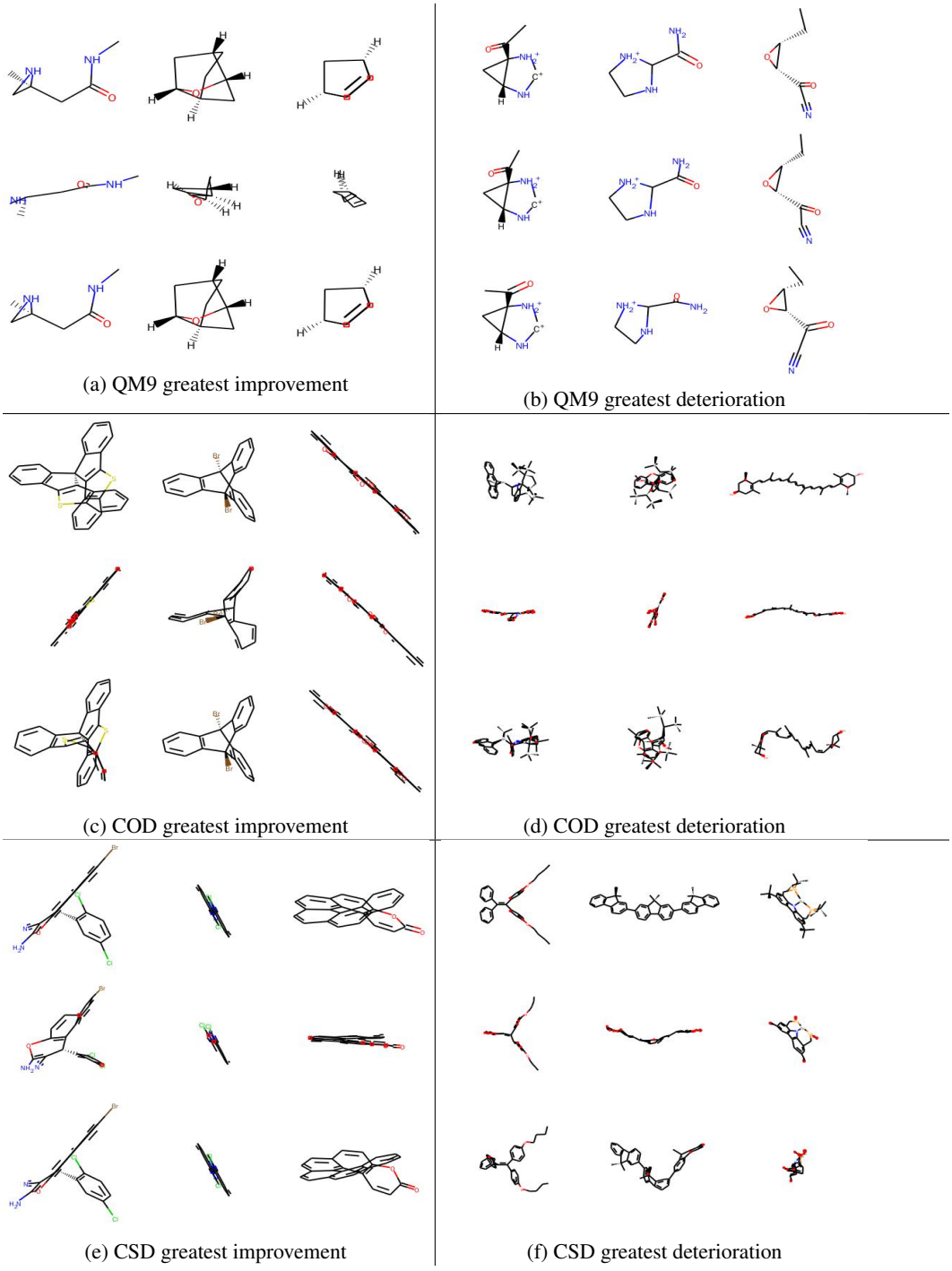


(e) CSD greatest difference in favour of neural network predictions



(f) CSD greatest difference in favour of ETKDG + MMFF predictions

Figure 8: This figure shows the three molecules in each dataset for which the differences between the RMSDs of the neural network predictions and the baseline MMFF predictions were greatest in favour of the neural network predictions ($max\ (RMSD_{CVGAE} - RMSD_{ETKDG+MMFF})$), and the three for which this difference was greatest in favour of the ETKDG + MMFF predictions ($max\ (RMSD_{ETKDG+MMFF} - RMSD_{CVGAE})$). The top row of each subfigure contains the reference molecules, the middle row contains the neural network predictions and the bottom row contains the conformations generated by MMFF using ETKDG as initialization.



(a) QM9 greatest improvement

(b) QM9 greatest deterioration

(c) COD greatest improvement

(d) COD greatest deterioration

(e) CSD greatest improvement

(f) CSD greatest deterioration

Figure 9: This figure shows the three molecules in each dataset whose RMSD decreased the most and the three whose RMSD increased the most on applying MMFF to the conformations predicted by the neural network. The top row of each subfigure contains the reference molecules, the middle row contains the neural network predictions and the bottom row contains the conformations generated by applying MMFF to the neural network predictions.

feature	type	dimension
atom type	one-hot (possible heavy atoms)	vary
atomic number	integer	1
chirality	one-hot (R, S)	2
is aromatic	binary	1
hybridization	one-hot (sp, sp^2, sp^3, sp^3d^1, sp^3d^2)	5
degree	integer	1
formal charge	integer	1
no. hydrogens	integer	1
no. radical electrons	integer	1
implicit valence	integer	1
no. rings for each ring size	integer (ring sizes 3, 4, 5, 6, 7, 8)	6
total		> 20

Table 3: Node features.

feature	type	dimension
bond type (if bond)	one-hot (single, double, triple, aromatic)	4
stereochemistry (if bond)	one-hot (E, Z)	2
is conjugated (if bond)	binary	1
is in ring (if bond)	binary	1
is in same ring	binary	1
graph distance (shortest path)	integer	1
total		10

Table 4: Edge features.

edge features are only calculable if the corresponding atom pair is bonded, while the last two edge features are calculable for every atom pair. All features are generated using RDKit (Landrum).