Autoencoding undirected molecular graphs with neural networks

Jeppe Johan Waarkjær Olsen, Peter Ebert Christensen, Martin Hangaard Hansen, and Alexander Rosenberg Johansen*

Department of Computing, Technical University of Denmark

E-mail: aler@dtu.dk

Abstract

We propose a machine learning model, inspired by language modeling from natural language processing, which can automatically correct molecules in discrete representations using a structure rule learned from a collection of undirected molecular graphs. Using discrete representations of molecules allows cheap, fast, and coarse grained insights. We introduce an adaption on a modern neural network architecture, the Transformer, which can learn relationships between atoms and bonds. The algorithm thereby solves the unsupervised task of recovering partially observed molecules represented as undirected graphs. This is to our knowledge, the first work that can automatically learn any discrete molecular structure rule with input exclusively consisting of a training set of molecules. In this work the neural network successfully approximates the octet rule, relations in hypervalent molecules and ions when trained on the ZINC and QM9 dataset. These results provides encouraging evidence that neural networks can learn advanced molecular structure rules and dataset specific properties, as the transformer surpasses a strong octet-rule baseline.

Introduction

In drug discovery,[1,2] catalysis,[3,4] combustion and other areas of chemistry, the number of possible relevant molecules grows exponentially with size of the molecule or size of the reaction network. Modeling and exploring large data sets of molecules therefore benefit from fast coarse grained methods to generate, select, filter and correct molecules represented as data.

A trade-off exists between accuracy and cost from accurate, but expensive, quantum chemical methods to highly scalable learning algorithms which provide coarse grained predictions on the target properties. In machine learning models for atomic structure data, an important distinction lies between models on continuous real-space representations, such as force fields, as opposed to models on discrete graph based representations, the latter of which is a coarse grained, but far more scalable representation.[5,6]
Machine learning methods for predicting properties based on discrete encodings have only recently begun to train and predict on undirected graphs, as opposed to directed linear graphs or sequences such as SMILES.[6–11] A requirement for any model is that molecules are considered invariant under permutation, translation and rotation, which calls for undirected graph representations and models that are inherently permutation invariant.[12]

The tasks of correcting, completing and generating molecules in discrete representations usually rely on the octet rule as the fundamental structure rule. This alone defines the validity, i.e. the stability of molecules.[6,13–16] Despite the success of the octet rule, it has limitations such as not allowing hyper-valent molecules, ions, and inability to capturing structure phenomena in liquid environments.

In this paper, we introduce an unsupervised task, known as masked language modeling or denoising autoencoder,[17–19] over an undirected discrete graph representation of a given

molecule. We define the unsupervised task as applying a corruption to a molecule and learning how to revert such corruption to recover the valid molecule. This objective allows us to learn the underlying structure rule without any hard-coded heuristic, but exclusively by observing valid molecular graphs.[20]

This paper presents several models trained on the QM9 and ZINC.[16,21,22] The ground truth we approximate, and benchmark against, is either that the original atom is correctly predicted (sample F1), or that the prediction is correctly predicted or satisfies the octet rule.

In Natural Language Processing (NLP) the a goal of statistical and probabilistic language modeling is to learn the joint probability mass function of sequences.[23] Historically, this has been accomplished by calculating the probability of observing a word given the sentence that precedes it.[24] Methods exploiting the sequential relationship between words in text has been ranging from probabilistic finite automaton[24] to distributed word embeddings[25] and recurrent neural networks (RNN).[26–28] To cover the most recent development in language modeling, adapted to fit undirected graphs with high degrees, we test the following methods of increasing complexity:

- `unigram` — unconditional probabilities of the atoms
- `bag-of-neighbors` — neural network that counts the neighboring atoms
- `bag-of-atoms` — neural network that counts all atoms in the molecule
- `binary/bond-transformer` — neural network architecture with attention using either binary representations of bond types or full bond type information

The `binary/bond-transformer` is inspired by a recent trend in NLP, called masked language modeling, where the sequential requirement can be relaxed.[18] Most noticeably, we modify masked language modeling to work with molecules by masking atoms and using graph adjacency matrices to model intermolecular relationships.

Methods

In this section we present several methods to restore molecules that have been corrupted. We formally define this as an unsupervised learning task over discrete molecular graphs, where we use a simple corruption function that masks atoms. We introduce four unsupervised models with increased modeling complexity: counting, nearest neighbor, all atoms, and the transformer model; which we compare against the octet-rule.

Unsupervised learning of discrete molecular graphs

Autoencoders are a type of neural network that use unsupervised learning. They create an efficient representation of data by extracting important features.[29] However, when the autoencoder has a larger hidden than input dimension the neural network risk learning the identity function, which makes the autoencoder useless. The denoising autoencoder[19] alleviates this problem by corrupting the input data on purpose.

In our case the input is a molecule, which we represent as an undirected graph with discrete edges $G = (V, E)$. Here V is a set of vertices (atoms), such that $(a, i) \in V$ where $a \in A$ is the element, $i \in \mathbb{N}$ is the index. E is the set of undirected bonds between atoms in the molecule, such that $E \subseteq \{x, y, b\} \mid (x, y) \in V^2 \wedge (x, y) = (y, x) \wedge x \neq y$, where $b \in \{1, 2, 3\}$ is the bond type: single, double, or triple.

In the denoising autoencoder we corrupt the input with the corruption function $\kappa : V \rightarrow \tilde{V}$. By recovering a corrupted molecule we force the neural network to learn the underlying rules governing molecules.

For the experiments we use a corruption function that mask atoms in a molecule with bond type intact. This method of corruption is inspired by the masked language model

presented in BERT.[18] To apply the corruption function we replace a set of vertices with the `<MASK>` token as described in equation (1)

$$\tilde{V} = V - V_{\texttt{subset}} \cup \kappa(V_{\texttt{subset}}), V_{\texttt{subset}} \subseteq V \tag{1}$$

$$\kappa(a, i) = (\texttt{<MASK>}, i) \tag{2}$$

Given the corrupted graph, $\tilde{G} = (\tilde{V}, E)$, we want to maximize the probability of recovering the original graph, G, which equals maximizing the probability of the masked atoms.

$$\max P(G|\tilde{G}) = \max P(V_{\texttt{subset}}|\tilde{G}) \tag{3}$$

In the following subsections we present four models maximizing this objective with increasing modeling complexity.

Counting: atomic frequencies

A counting-based model obtains the distribution of atom types by calculating their frequencies over a dataset. Counting-based models will by intuition have high accuracy when the dataset is biased, which we find the `QM9` and `ZINC` are (see Table 1).

The count-based model is motivated by the probability chain-rule, where we can model the joint probability of the atoms $v_i = (a, i) \in V$ in a molecule.

$$P(v_1, v_2, \ldots, v_n) = P(v_1)P(v_2|v_1) \ldots P(v_n|v_1, \ldots, v_{n-1}) \tag{4}$$

While equation 4 allows us to exactly estimate the conditional atom distribution, the condition grows exponentially with the amount of vertices and becomes infeasible due to the exponential requirement of data and compute. In NLP the directionality of the sentence allows for clipped, n-gram, versions of equation 4 where the prediction of the word distri-

bution is only conditioned on the last k tokens $P(v_n|v_1,\ldots,v_{n-1}) = P(v_n|v_{n-k},\ldots,v_{n-1})$. Using N-grams significantly reduces required computation and data while exploiting locality in language.[24]

In molecules, the degree of vertices and lack of directionality makes such n-gram models cumbersome as each atom can have a tree of recursive n-grams. Because of such, we limit ourselves to only consider unigram models (1-grams) for the counting case. A unigram model splits the probability of different terms in a context into a product of individual terms, disregarding the condition of equation 4.

$$P_{\texttt{unigram}}(v_1, v_2, \ldots, v_n) = P(v_1)P(v_2)\ldots P(v_n) \tag{5}$$

$$P(a_j) = \frac{\texttt{count}(a_j)}{\sum_a \texttt{count}(a)} \tag{6}$$

Unigram models have the benefit of being relatively simple to implement and interpret as they merely count the occurrence of elements in the trainingset. The unigram distribution on the `QM9` and `ZINC` trainingset are shown in Table 1.

Table 1: Unigram probabilities for QM9 and ZINC trainingset. The unigram probabilities corresponds to the distribution of elements in the dataset.

Elements	**QM9**	**ZINC**
P(H)	0.519	0.47407
P(C)	0.347	0.38691
P(O)	0.078	0.05416
P(N)	0.054	0.06109
P(F)	0.002	0.00856
P(P)	0	0.00001
P(S)	0	0.00913
P(Cl)	0	0.00452
P(Br)	0	0.00144
P(I)	0	0.00011

This indicates a bias in the data towards the elements H and C. Note that the unigram model will always predict with the same probability distribution of elements for any vertice

or atom as it does not use context.

Using our objective from equation 3 we calculate our unigram probability of the corrupted molecule as

$$\max P(V_{\texttt{subset}}|\tilde{G}) = \max \prod_{\tilde{v} \in V_{\texttt{subset}}} P(\tilde{v}) \tag{7}$$

Bag of vectors: neighbors and atoms

In a `bag-of-vectors` model a molecule is represented as a multiset of its tokens (elements and/or bonds),[5,30] disregarding structure but keeping multiplicity (i.e. multiple occurrences of the same token). Each token, x, is embedded as a trainable vector of real numbers $x \in \mathbb{R}^d$. By summing the n tokens of a molecule over the d features we obtain the `bag-of-vectors` : $\mathbb{R}^{n \times d} \rightarrow \mathbb{R}^d$ representation (sum is used instead of mean to keep multiplicity). The `bag-of-vectors` representation is used as input to a neural network that learns to predict the masked tokens $V_{\texttt{subset}}$. The token vectors, also known as embeddings, and the neural network are jointly optimised with stochastic gradient descent.[25,31] Using eq. 3 we define two `bag-of-vector` models for our study: a bag of neighboring atoms (eq. 8) and a bag of all atoms in the corrupted molecule (eq. 10).

$$\max_{\theta} P_{\theta}^{\texttt{bag-of-neighbors}}(V_{\texttt{subset}} \mid \tilde{G}) = \max_{\theta} \prod_{v \in V_{\texttt{subset}}} P_{\theta}(v|V_{neighbors}) \tag{8}$$

$$V_{neighbors} = \{v_j \mid (v_j, v) \in E\} \tag{9}$$

$$\max_{\theta} P_{\theta}^{\texttt{bag-of-atoms}}(V_{\texttt{subset}}|\tilde{G}) = \max_{\theta} \prod_{v \in V_{\texttt{subset}}} P_{\theta}(v|\tilde{V}) \tag{10}$$

$$P(x_j|\tilde{X})_{\theta} = \texttt{softmax}(Wh_{\theta}(\tilde{X}))_j = \frac{\exp((Wh_{\theta}(\tilde{X})_j)}{\sum_{i=0}^{|\Sigma|-1} \exp((Wh_{\theta}(\tilde{X}))_i)} \tag{11}$$

$$h_{\theta}(\tilde{X}) = \texttt{NN}(z_{\theta}(\tilde{X})) \tag{12}$$

$$z_{\theta}(\tilde{X}) = \sum_{\tilde{x} \in \tilde{X}} \texttt{embedding}(\tilde{x})) \tag{13}$$

To represent our corrupted tokens in equation 13, $\tilde{X}$, we use an embedding function. Embedding functions, $\texttt{embedding}(x) \in R^{d_{emb}}$, are a popular way to represent input tokens in NLP.[25] The embedding function uses a dense vector representation for each token class, which allows the embedding function to learn relations between token classes. As we want to model all the tokens in the molecule with a neural network we need to have a fixed feature space. A convenient way to achieve such is the `bag-of-vectors`, which sums all tokens to achieve a fixed-sized distributed feature representation of $\tilde{X}$. Given a `bag-of-vectors` representation, z_{θ}, we want to model the corrupted atoms. We choose to use a feed forward neural network in equation 12, $NN : R^{d_{emb}} \rightarrow R^{d_{nn}}$. A NN is a powerful non-linear function approximator that can learn relations between tokens. To map the NN output onto probabilities for the element classes we use trainable a linear projection, $W \in R^{|\Sigma| \times d_{nn}}$, followed by the `softmax` function (eq. 11), which squeezes the output to the probability domain. $|\Sigma|$ denotes the amount of elements we predict over for each atom (e.g in `QM9` that would be five: H, C, N, O and F).[24]

The bag-of-vector models are trained end-to-end with stochastic gradient descent using

a cross-entropy loss function given the set of correctly labelled atoms $V_{\texttt{subset}}$.[24]

$$L(V_{\texttt{subset}}, \tilde{G}) = \sum_{v \in V_{\texttt{subset}}} \log P_\theta(v \mid \tilde{G}) \tag{14}$$

Where the conditional probability, $P_\theta(v \mid \tilde{G})$, is calculated accordingly to; equation 8 for the `bag-of-neighbors` and equation 10 for the `bag-of-atoms`.

Since these models rely on either pairs of atoms (neighbors) or mere counts (atoms) they can work with a broad family of corruption functions. However, only including compositional information is a coarse representation of a molecule, e.g. we have several large subsets of molecules in QM9 with fixed element compositions, which have varied structures but identical bag-of-atoms representations.
Moreover, in equation 10 for the all-atom based model we have the same condition for all the predictions, $V_{\texttt{subset}}$. As such, it will always predict the same distribution for the masked atoms, given the composition.

While these models are limited in representational power, they provide a rudimentary baseline for our introduction of the transformer model on undirected molecular graphs.

Tranformer: atomic context

The ideal discrete representation of a molecule must have permutation, translation and rotation invariance as well as allowing branched and aromatic molecules, in other words, an undirected graph with vertices of high degree.[12]

We adapt the transformer[32] to fulfill all of the above requirements. The transformer is a neural network architecture that uses repeated adaptive receptive fields (known as attention[33]) to model relations between words in a text given their context. The original

transformer, like many other NLP models, uses sequence information to build context from relative word positioning. Instead of a sequence representation we use an adjacency matrix to give a complete description of the molecule.[34]

We test two approaches for encoding bond information: the `binary-transformer`, where all bonds are binary, and the `bond-transformer`, where bonds type (1, 2, or 3) is given.

Using eq. 3, the transformers take the entire graph representation as input and learn a parameterized function that we train to maximize eq. 15.

$$\max_{\theta} P_{\theta}^{\texttt{transformer}}(V_{\texttt{subset}} \mid \tilde{G}) = \max_{\theta} \prod_{v \in V_{\texttt{subset}}} P_{\theta}(v|\tilde{G}) \tag{15}$$

$$P_{\theta}(v_j|\tilde{G}) = \texttt{softmax}(W\ \texttt{transform}_{\theta}(\tilde{G})^{L})_j \tag{16}$$

Similar to the `bag-of-vectors` we use a `softmax` function to learn class (atomic element) probabilities. The transformer consist of L $\texttt{transform}_{\theta}$ layers. Each layer applies a non-linear function to build molecular context. The final layer, $\texttt{transform}_{\theta}(\tilde{G})^{L}$, is used for classification. Each layer consist of an attention mechanism with layer normalization,[35] skip-connections,[36,37] and a feed forward neural network;[38] which allows the transformer to model structures and dependencies for each atom using the entire molecule. Where the atomic representation of each layer is defined as $h^l, z^l \in \mathbb{R}^{|V| \times d_{transform}}$.

$$\texttt{transform}_{\theta}(V, E)^l = h^l = \texttt{layer-norm}(z^l + \texttt{FFN}(z^l)) \tag{17}$$

$$z^l = \texttt{layer-norm}(h^{l-1} + \texttt{Attention}(h^{l-1}, E)) \tag{18}$$

$$h^0 = \texttt{atom-embedding}(V) \tag{19}$$

The `atom-embedding`, h^0, is identical to the `embedding` in equation 13. To represent either the full bond type or just the binary edge information we set the adjacency matrix $E_{i,j} \in$

$\{0, 1\}$ for the `binary-transformer` or $E_{i,j} \in \{0, 1, 2, 3\}$ for the `bond-transformer`. Like the `bag-of-vector` models, this is trained with stochastic gradient descent using the cross-entropy loss function (see equation 14).

Attention

As with the original transformer, we use the key-value lookup `Attention` function. This layer can adaptively align information between atoms conditioned on the context of other atoms.[33,39] Our implementation takes a layer of hidden representations, h^l, and an adjacency matrix of edges, E, as input. Notice that we have separate trainable `bond-embedding` functions for the key, e^K, and value, e^V, edge representations.

$$\texttt{Attention}(h, E)_i = \sum_{j=1}^{n} \alpha_{ij} \left(h_j W^V + e_{ij}^V \right) \tag{20}$$

$$\alpha_{ij} = \frac{\exp \phi_{ij}}{\sum_{k=1}^{n} \exp \phi_{ik}} \tag{21}$$

$$\phi_{ij} = \frac{\left(h_i W^Q \right) \left(h_j W^K + e_{i,j}^K \right)^T}{\sqrt{d_{transform}}} \tag{22}$$

$$e = \texttt{bond-embedding}(E) \tag{23}$$

where $a_{ij} \in [0, 1]$ is the attention weights, n is the number of vertices, W^Q, W^V, and $W^K \in \mathbb{R}^{d_{transform} \times d_{transform}}$ are trainable weights. The $\texttt{bond-embedding} : E^{|V| \times |V|} \rightarrow \mathbb{R}^{|V| \times |V| \times d_{transform}}$ takes an adjacency matrix and returns a three dimensional tensor with a distributed representation for each edge.

To have a more expressive attention function we use the multi-head attention mechanism by concatenating k attention layers. The k attention layers are projected to the hidden size of the network $\mathbb{R}^{d_{transform} \times k} \rightarrow \mathbb{R}^{d_{transform}}$, such that

$$\texttt{Multi-Head-Attention}(h, E)_i = [\texttt{C\_}1, \texttt{C\_}2, \ldots \texttt{C\_}k] W_{multi} \tag{24}$$

where `C_i` corresponds to an instance of `Attention` (eq. 20) and $W_{multi} \in \mathbb{R}^{(d_{transform} \cdot\ k) \times d_{transform}}$ is a trainable weight. This is further illustrated in Figure 1

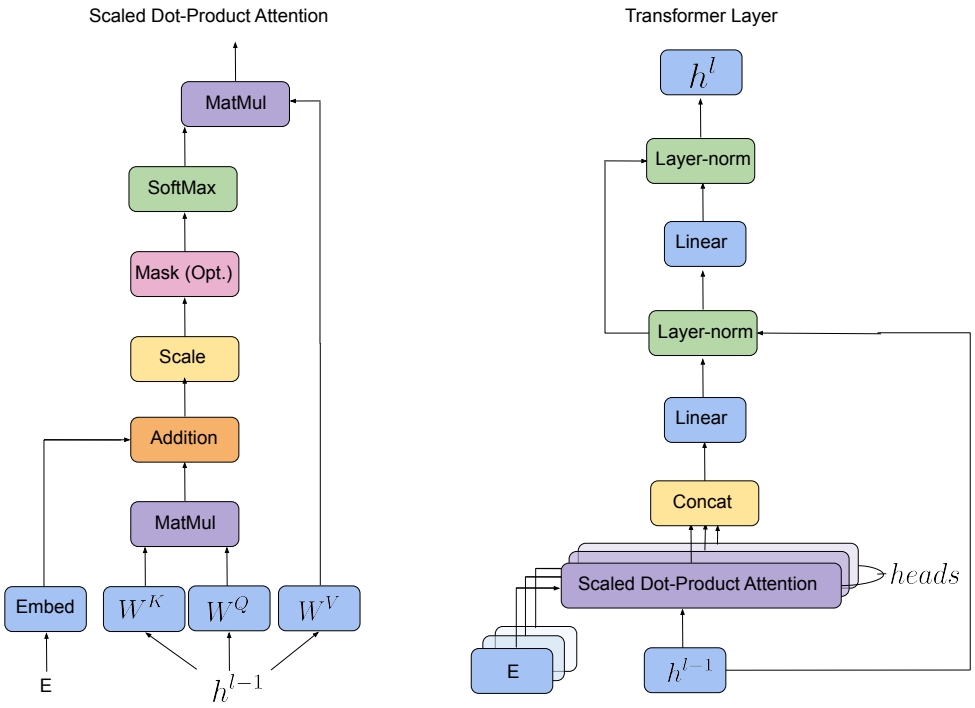

Figure 1: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention with multiple layers consisting of several attention layers running in parallel. Figure modified from Vaswani et al. [32]

.

Experimental setup

In our experiments we test the described models of the `unigram`, `bag-of-neighbors`, `bag-of-atoms`, `binary-transformer`, and `bond-transformer` as denoising autoencoders on the `QM9` and `ZINC` datasets. [16,21]

Pre-processing

The `QM9` dataset has 134 000 organic molecules with of five types of atoms; $A = \{$H, C, N, O, F$\}$. Similarly, the `ZINC` dataset has 250 000 drug-like molecules with 10 type of atoms,

$A = \{\text{H, C, N, O, F, P, S, Cl, Br, I}\}$. The molecules are represented as a SMILE strings[40] corresponding to their discrete graph representations. We obtain an adjacency matrix for each molecule from the SMILES string using Rdkit.[41] Since we use the `QM9` dataset to benchmark our ability to approximate the octet rule, we discard any molecules that contains net charges (1808 molecules). In the ZINC dataset, we keep all molecules including molecules with charges and hypervalent molecules.

The resulting set of adjacency matrices are split using scaffolding to homology partition the molecules. We make a 15% test, 15% validation, and 70% training set split. In Figure 2 we show the distribution of elements for different sizes of molecules. Here we see that in both the `QM9` and `ZINC` dataset, the size of the molecules are not uniformly distributed, with few small and large molecules. Furthermore, the molecules in `ZINC` are generally larger than the ones in `QM9`. The distribution of different elements depends somewhat on the size of the molecule, especially for smaller molecules.

To stress test the models we generate several validation-/ and tests sets with increasing complexity. For `ZINC`, the datasets have either 1, 10, 20, 30, 40, 50, 60, 70, or 80 atoms randomly masked in the molecule, denoted by $n_{corrupt}$. For $n_{corrupt} = 1$, we oversample the molecules, by generating five unique different maskings per molecule. This is done to reduce the variance of our estimated performance, especially on molecules with few atoms, since there only exists few of these in the dataset.

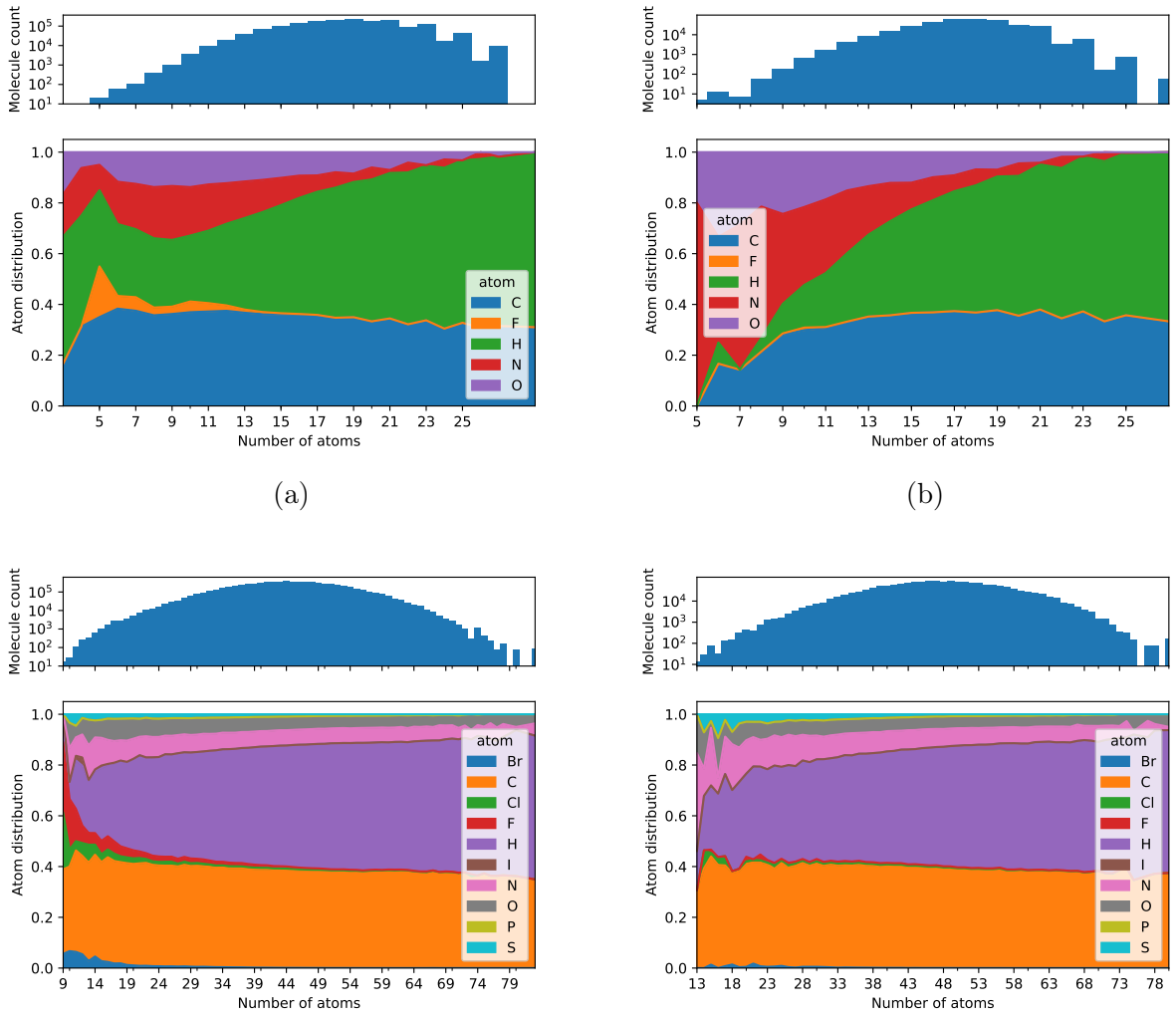

Figure 2: Count (top) and distribution of elements per molecule size (number of atoms) for (a) `QM9` training set, (b) `QM9` test set, (c) `ZINC` training set and (d) `ZINC` test set.

Training details

To train the model we optimize the objective for each of the methods (equation 7, 8, 10, and 15) by corrupting the atoms, with masking, and reversing the corruption. When increasing the masking we have an exponentially growing combination of corruptions, for which reason we sample the atom modifications in an online manner for training. To make the model

robust towards different levels of corruption we employ an ϵ-greedy corruption scheme.[42]

$$\Pr(\text{no. of corruptions} = k) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|V|} & k = n_{\texttt{corrupt}} \\ \frac{\epsilon}{|V|} & k \neq n_{\texttt{corrupt}} \end{cases} \tag{25}$$

in the first case, with probability $1 - \epsilon$, we corrupt $n_{\texttt{corrupt}}$ atoms and in the second case, with probability $\epsilon - \frac{\epsilon}{|V|}$, we uniformly corrupt between 1 to $|V|$ where $|V|$ is the amount of atoms in the molecule. We use $n_{corrupt} = 1$ for training. The models are trained on an Nvidia Tesla V100 GPU, using Adam optimization,[43] and a learning rate of 0.001 for all models. The experiments are implemented in PyTorch.[44]

Evaluation

When predicting the true value of a masked atom in a molecule, several solutions might be equally correct. In NLP this is often handled by considering sample exact match. However, for molecular structures we know that multiple elements could exist in the same position. This is formalized by the octet rule, which allows the prediction of elements with a similar number of valence electrons. We define the union of an exact match and elements that are correct with respect to the octet rule as octet accuracy (e.g H and F could both be possible when one bond is present). Given that the `QM9` dataset is generated by the octet rule, correctly understanding the octet rule would result in 100% octet accuracy, which is why we use it as our first dataset - to see how difficult it is to learn the octet rule. The `ZINC` dataset on the other hand does not conform to the octet rule as it contains ions and hypervalency. This tests our models ability to go beyond the octet rule and learn other underlying structure rules of molecules present in `ZINC`.

Since the distribution of atoms in the data is heavily biased, we use the F1-micro and F1-macro scores, which are a weighted average of the precision and recall.[45]

While the octet rule becomes increasingly ambigious when more elements are allowed, understanding what underlying structures are more common, exact match is of interest. This is important to evaluate if we can fit the specific distribution of a dataset. We define exact match as sample accuracy and F1. Moreover, we supply sample perplexity measures, which is a more fine grained way of assessing certainty in model prediction.

$$\text{Perplexity} = \exp\left(-\frac{1}{|V_{subset}|}\sum_{v\in V_{subset}} \log P(v|\tilde{G})\right) \tag{26}$$

We benchmark our proposed models against an octet rule model. The octet rule model counts the number of covalent bonds of the masked atom and predicts the unigram probabilities of the elements of the corresponding group in the periodic table. We denote this model as the `octet-rule-unigram`. When predicting elements with ambiguity (e.g hydrogen and fluor in the `QM9` dataset) the `octet-rule-unigram` will therefore not obtain perfect perplexity. As no predictions exsists for hypervalent elements (five and six covalent bonds), the `octet-rule-unigram` predicts uniform probability. Notice that as opposed to using a unigram model, this actually gives better perplexity as S is underrepresented in the dataset (see Table 1).

Results

We test all proposed models on octet and sample accuracy, F1, and perplexity. First, we evaluate the models on the `QM9` dataset, where the purpose is to learn an Octet-rule approximation. Next, we measure the models on the `ZINC` dataset and attempt to extend the octet approximation with hypervalent molecules and ions. Finally, we provide a qualitative insight into model prediction by analyzing six different samples (three correct, three incorrect) from the `binary-transformer`.

QM9 - approximating Octet rule

In Table 2, we evaluate our models on octet rule accuracy, octet rule F1-(micro/macro) and sample perplexity.

As expected, the `bond-transformer` achieves almost perfect performance (**99.99%** octet accuracy), since the task becomes a matter of counting covalent bonds, once you include the order of the bonds. The `binary-transformer` also achieves excellent performance (**99.73%** octet accuracy), even though it is not given any information about bond types. With 1 masked atom, the problem of recovering the corrupted atom, without any bond types, can be seen as a combinatorial problem. This suggests that the `binary-transformer` is able approximately solve this problem by inferring the bond orders from the remaining molecule.

By only using neighborhood information, the `Bag-of-neighbors` model gets 90%, which serves as a very strong baseline, but without the full structural context, the model cannot approximate the octet-rule. Similar, by only providing compositional information, the `Bag-of-Atoms` model, performs significantly worse, showing that structural and neighboring information is important.

Finally, the `Unigram`, relies purely on the frequency of occurrence of elements in the dataset, thus always guessing the masked atom is hydrogen and performs poorly.

We provide extended results on masking multiple atoms, transformer model sizes, and accuracy by length in Supporting information.

Table 2: Performance of our models for 1 masked atoms per molecule. The uncertainty corresponds to the standard deviation of ten models, trained with different start seed.

Model	Octet Accuracy	Octet F1 (micro/macro)	Perplexity
`bond-transformer`	**99.99**±0.01	**99.99**±0.01 / **99.99**±0.01	**1.002**±0.001
`binary-transformer`	99.73 ±0.01	99.73 ±0.01 / 93.44 ±4.20	1.009 ±0.002
`bag-of-neighbors`	90.67 ±0.01	90.67 ±0.01 / 77.18 ±0.01	1.281 ±0.004
`bag-of-atoms`	65.77 ±4.48	65.77 ±4.48 / 44.30 ±4.92	3.310 ±0.478
`Unigram`	47.32	47.32 / 32.85	3.104
`octet-rule-unigram`	100	100 / 100	1.002

ZINC - going beyond the octet rule

We consider the `ZINC` dataset as it cannot be fully explained by the octet rule and has a more of ambiguous elements and a larger quantity of them than `QM9`. E.g. with $n_{corrupt} = 1$, our `ZINC` testset contains 924 Fluors to be predicted as opposed to 9 Fluors in `QM9`.

Given some elements, namely ions and hypervalent molecules, cannot be predicted by the octet rule we add k-smoothing[24] to the `octet-rule-unigram` model. This avoids the case of 0 probability, which would result in infinite perplexity loss. We optimize k on the validation set and found the optimum at k=1842.

From Table 3 we see that the `octet-rule-unigram` model no longer has 100% octet F1, which emphasizes to what extend that the dataset cannot be fully explained by the octet rule, due to molecules with charges and hypervalency. Both our transformer models performs similar or better than the `octet-rule-unigram`, when evaluated on Octet F1, sample F1 and sample perplexity. This is especially the case with with F1 macro, that puts more emphasis on the underrepresented cases, which in our case are the most interesting. This indicates that the transformer models also have learned to discriminate between elements that should be equally likely from the perspective of the octet rule, but might have higher likelihood under a given structure.

Table 3: Performance of our models for 1 masked atoms per molecule. The uncertainty corresponds to the standard deviation of ten models, trained with different start seed.

Model	Octet F1 (micro/macro)	Sample F1 (micro/macro)	Perplexity
`bond-transformer`	**99.52**±0.04 / **97.97**±3.17	**98.64**±0.03 / **62.67**±3.19	**1.047**±0.001
`binary-transformer`	99.13 ±0.05 / 91.38 ±4.94	98.18 ±0.06 / 55.76 ±4.89	1.063 ±0.002
`octet-rule-unigram`	99.17 / 88.65	97.22 / 38.48	1.164
`bag-of-neighbors`	90.73 ±0.03 / 76.50 ±0.45	89.00 ± 0.03 / 29.47 ±0.37	1.412 ±0.004
`bag-of-atoms`	50.84 ±0.30 / 56.75 ±0.58	49.06 ±0.32 / 9.84 ±0.53	3.135 ±0.073
`Unigram`	48.05 / 56.40	46.10 / 6.31	3.221

Since our model has the ability to corrupt multiple atoms in a molecule, we investigate how the amount of corruption affects the performance. This is shown in Figure 3 (see Supporting information for F1 metrics, and accuracy/F1 by number of atoms). Here we see that the accuracy of `Bond-Transformer` barely is affected, even when all the atoms in the molecule are masked. This suggests that the model primarily uses the structural information (bond type and connections). The `Binary-Transformer` however drops slightly in accuracy as the molecule is corrupted. This makes sense, as without bond type information, the model can use the label of the remaining atoms to infer the bond types, but as we corrupt more, we limit the available information in the molecule. The same is the case for the `Bag-of-neighbors`. In the case of `Bag-of-atoms`, the model seem to converge to the `Unigram`.

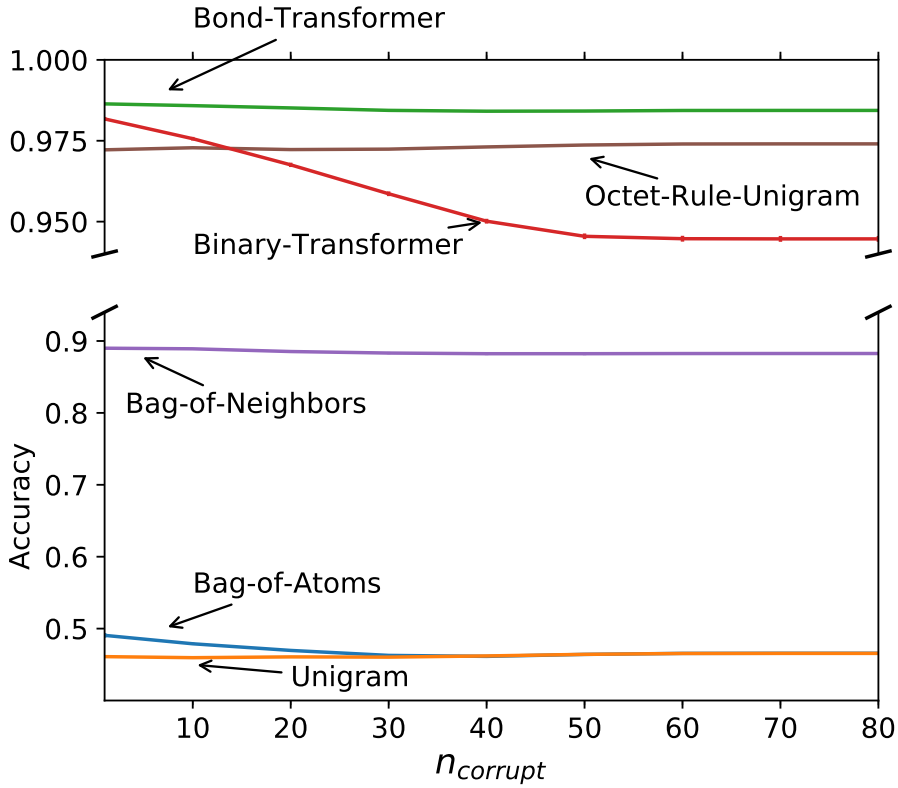

Figure 3: Sample accuracy of the models, evaluated by different number of masked atoms on ZINC dataset. Errors bar corresponds to standard deviation of 10 models trained with different start seed.

To investigate if our model can understand ions we have visualized the confusion matrix for atoms with four covalent bonds in Figure 4 (other bond order confusion matrices can be found in Supporting information). In this case, our dataset contains examples of C, but also N^+ ions and hypervalent S. The confusion matrix shows that while our `Octet-rule-unigram` model only predicts C, both the `Binary-transformer` and `Bond-transformer` has learned, that both S and N can have four covalent bonds and how to discriminate between them. Thus the models seems to have successfully learned a more complex structure rule, than the octet rule.

To better understand the models success in predicting hypervalent elements we visualize the confusion matrix for five and six covalent bonds in Figure S1 and S2 (see Supporting

information). With five covalent bonds we only have one occurrence of P, which is correctly predicted by the `bond-transformer`. For six covalent bonds, both transformers correctly predict all elements with S.

To assess the models ability for predicting ambiguous elements we visualize the confusion matrix for one covalent bond in Figure S2. In particular, we find that both transformer models (`binary-transformer`/`bond-transformer`) can successfully predict a large number of F molecules (279/270) while only misclassifying a small amount of H (23/21) as F.

For future investigations, we find that the `QM9` and `ZINC` datasets are heavily biased towards H and C. This might make training difficult due to dataset imbalances and could be improved by oversampling rare elements.[46]

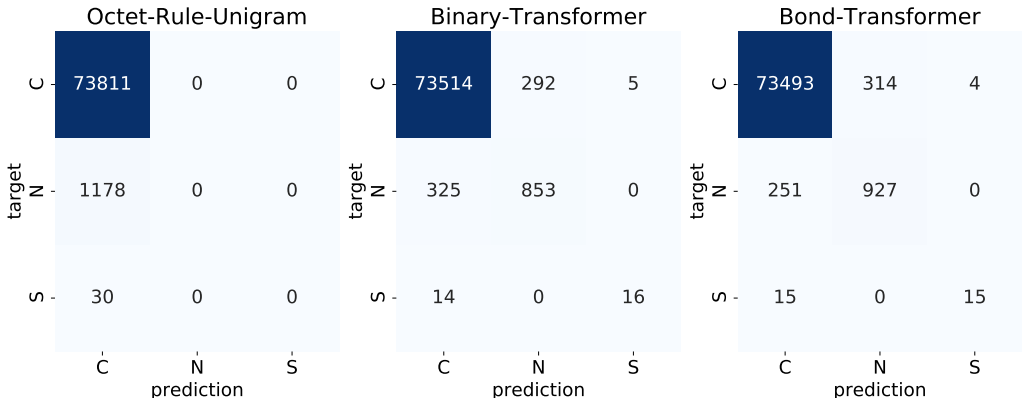

Figure 4: Confusion matrix for the testset, with $n_{\texttt{corrupt}} = 1$, where the masked atom has four covalent bonds. We provide this matrix for the `octet-rule-unigram`, `binary-transformer`, and `bond-transformer`.

Qualitative results

To investigate the `binary-transformer` corrections of atoms in a molecule, we inspect a few interesting predictions on the `ZINC` dataset. We show the molecules with the predicted conditional probabilities of the possible element labels on the masked atoms. Figure 5a

illustrates an example where the model correctly predicts N, even though N^- ions are very rare in the dataset. It also puts a reasonable amount of probability of the target being O, which could be a valid guess assuming the octet rule applies. In Figure 5b we see an example of a hypervalent S, which our model correctly predicts, with a very high certainty. The hypervalent S often appears in the dataset with the two double bonded O, which might be a giveaway for the model. The example in Figure 5c would however most likely not have a immediate explanation, but the model is very certain of it prediction, which is also correct. The context of the elements with one covalent bond is expected to be identical, under the octet rule, since both have one neighbor to any of the other elements in the data, but since the data is heavily biased towards hydrogen it is worth checking if the predicted probabilities are also biased. From Figure 5d, we see that even though the model incorrectly predicts H, the second most likely guess of Cl is correct, even though F appears twice as often in the dataset. A similar case can be seen in 5e where the model is in doubt between two elements, that both could be considered correct under the octet rule. Finally, in Figure 5f we have an example where the model is very certain, but makes a completely wrong prediction.

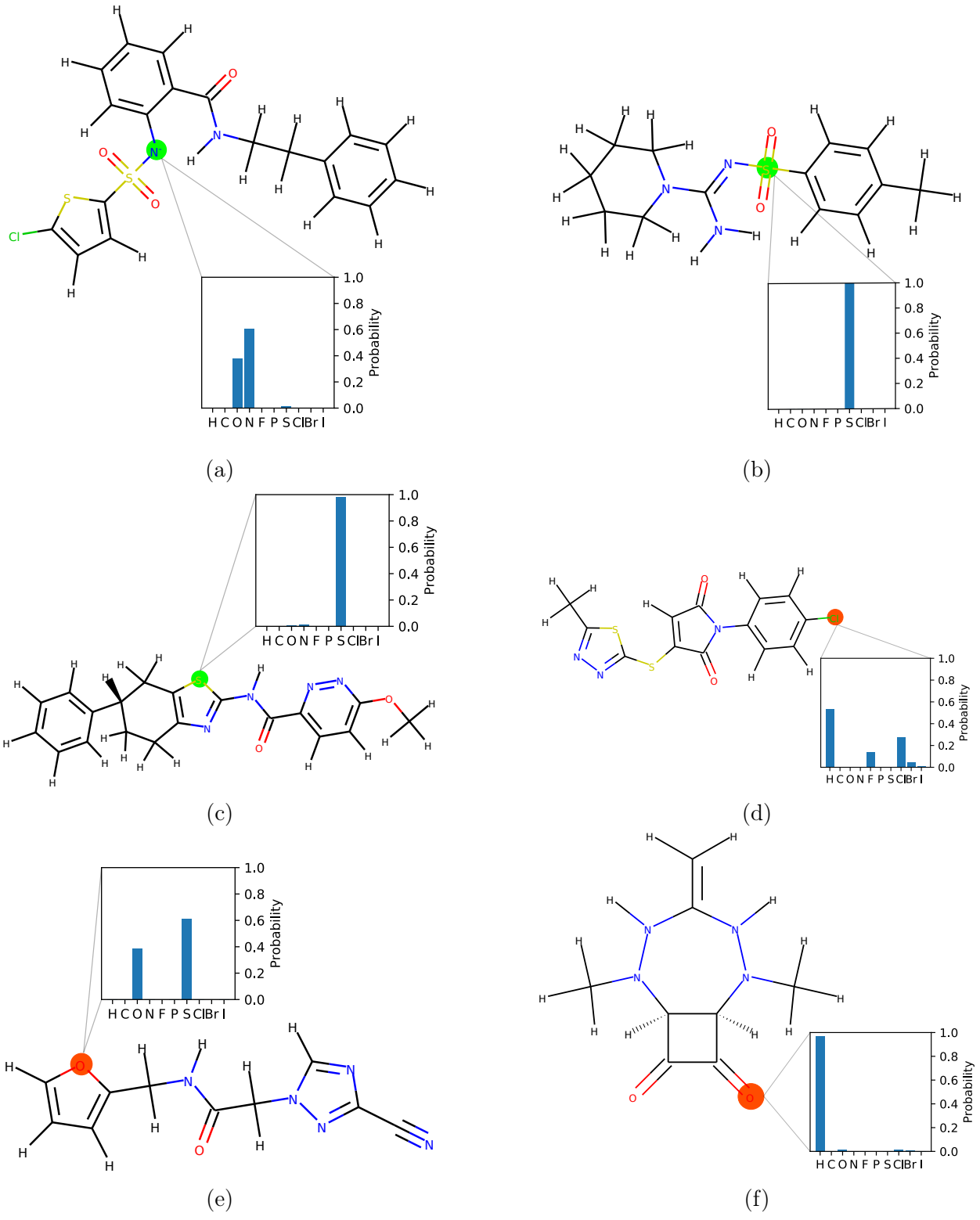

Figure 5: Predicted atom probabilities. The molecule corresponds to the true molecule, where the colored atom is the target we want to predict. Green corresponds to correct, and red to wrong predictions.

Conclusion

In this work we have introduced the `binary-transformer` and `bond-transformer` models, and evaluated their ability to recover masked atoms in an undirected molecular graph with discrete representations of bonds. The models achieves 99.73 ± 0.01 % and 99.99 ± 0.01 octet F1-micro on the `QM9` dataset, while masking 1 atom per molecule, suggesting that the model is capable of learning the octet rule, which is the underlying selection criteria for the QM9 data set.

When evaluated on the `ZINC` dataset, which contains more complex structure rules, our transformer models outperforms the `octet-rule-unigram` model in all metrics, including achieving 99.52 ± 0.04 and 99.13 ± 0.05 octet F1-micro, when masking 1 atom per molecule. When paired with the analysis of the confusion matrix, this indicates that the models has learned rules that exceed the octet rule, like ions and hypervalent molecules.

Deep learning models are extremely flexible and we have shown that the transformer architecture, which makes no assumption of the amount of atoms or bonds in a molecule, and could in theory be able to model a wide variety of molecular rules. With the high accuracy on the `QM9` and `ZINC` datasets we hypothesize that the transformer models, both the bond and binary based versions, could be well suited for learning other molecular rules, such as structure rules related to properties. As inference with the transformer is cheap, correcting billions of molecules is therefore possible.

The transformer model and embeddings made from undirected molecular graphs may furthermore be useful in chemical discovery tasks such as automatically generating and enumerating new molecules.

Moreover, years of progress in language modeling for NLP has given rise to strong contextual vectors of information that is now the defacto standard for state-of-the-art models in close to every popular dataset for benchmarking neural network performance.[18,47,48] In

particularly, these pretrained language models works surprisingly well for areas of limited labeled data, something that is fairly prevalent in many molecular chemistry tasks as data might be expensive to gather.

Acknowledgments

This research is funded by the Innovation Foundation Denmark through the DABAI project

References

(1) Ertl, P.; Rohde, B.; Selzer, P. Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. *Journal of medicinal chemistry* **2000**, *43*, 3714–3717.

(2) Lo, Y.-C.; Rensi, S. E.; Torng, W.; Altman, R. B. Machine learning in chemoinformatics and drug discovery. *Drug discovery today* **2018**, *23*, 1538–1546.

(3) Ulissi, Z. W.; Medford, A. J.; Bligaard, T.; Nørskov, J. K. To address surface reaction network complexity using scaling relations machine learning and DFT calculations. *Nature communications* **2017**, *8*, 14621.

(4) Boes, J. R.; Mamun, O.; Winther, K.; Bligaard, T. Graph Theory Approach to High-Throughput Surface Adsorption Structure Generation. *The Journal of Physical Chemistry A* **2019**,

(5) Hansen, K.; Biegler, F.; Ramakrishnan, R.; Pronobis, W.; Von Lilienfeld, O. A.; Müller, K.-R.; Tkatchenko, A. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *The journal of physical chemistry letters* **2015**, *6*, 2326–2331.

(6) Elton, D. C.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep learning for molecular generation and optimization-a review of the state of the art. *arXiv preprint arXiv:1903.04388* **2019**,

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

(8) You, J.; Liu, B.; Ying, Z.; Pande, V.; Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. Advances in Neural Information Processing Systems. 2018; pp 6410–6421.

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

(10) Jaeger, S.; Fulle, S.; Turk, S. Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition. *Journal of Chemical Information and Modeling* **2018**, *58*, 27–35, PMID: 29268609.

(11) Zheng, S.; Yan, X.; Yang, Y.; Xu, J. Identifying Structure–Property Relationships through SMILES Syntax Analysis with Self-Attention Mechanism. *Journal of chemical information and modeling* **2019**, *59*, 914–923.

(12) Mater, A. C.; Coote, M. L. Deep Learning in Chemistry. *Journal of Chemical Information and Modeling* **2019**,

(13) Fink, T.; Bruggesser, H.; Reymond, J.-L. Virtual exploration of the small-molecule chemical universe below 160 daltons. *Angewandte Chemie International Edition* **2005**, *44*, 1504–1508.

(14) Blum, L. C.; Reymond, J.-L. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society* **2009**, *131*, 8732–8733.

(15) Ruddigkeit, L.; Van Deursen, R.; Blum, L. C.; Reymond, J.-L. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of chemical information and modeling* **2012**, *52*, 2864–2875.

(16) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data* **2014**, *1*, 140022.

(17) Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.

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

(19) Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.-A. Extracting and Composing Robust Features with Denoising Autoencoders. Proceedings of the 25th International Conference on Machine Learning. New York, NY, USA, 2008; pp 1096–1103.

(20) Gerratt, J.; Cooper, D.; Karadakov, P. a.; Raimondi, M. Modern valence bond theory. *Chemical Society Reviews* **1997**, *26*, 87–100.

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

(22) Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G. ZINC: a free

tool to discover chemistry for biology. *Journal of chemical information and modeling* **2012**, *52*, 1757–1768.

(23) Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.

(24) Jurafsky, D.; Martin, J. H. *Speech and Language Processing (2Nd Edition)*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 2009.

(25) Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. USA, 2013; pp 3111–3119.

(26) Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; Khudanpur, S. Recurrent neural network based language model. INTERSPEECH. 2010; pp 1045–1048.

(27) Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent Neural Network Regularization. *CoRR* **2014**, *abs/1409.2329*.

(28) Merity, S.; Keskar, N. S.; Socher, R. Regularizing and Optimizing LSTM Language Models. 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. 2018.

(29) Hinton, G. E.; Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507.

(30) Hansen, M. H.; Torres, J. A. G.; Jennings, P. C.; Wang, Z.; Boes, J. R.; Mamun, O. G.; Bligaard, T. An Atomistic Machine Learning Package for Surface Science and Catalysis. *arXiv preprint arXiv:1904.00904* **2019**,

(31) Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings. 2013.

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

(33) Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* **2014**,

(34) Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-Attention with Relative Position Representations. *CoRR* **2018**, *abs/1803.02155*.

(35) Ba, L. J.; Kiros, R.; Hinton, G. E. Layer Normalization. *CoRR* **2016**, *abs/1607.06450*.

(36) He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, *abs/1512.03385*.

(37) Srivastava, R. K.; Greff, K.; Schmidhuber, J. Highway Networks. *CoRR* **2015**, *abs/1505.00387*.

(38) Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Fort Lauderdale, FL, USA, 2011; pp 315–323.

(39) Luong, M.-T.; Pham, H.; Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* **2015**,

(40) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences* **1988**, *28*, 31–36.

(41) Landrum, G. RDKit: Open-source cheminformatics. `http://www.rdkit.org`.

(42) Sutton, R. S.; Barto, A. G. *Reinforcement learning: An introduction*; 2018.

(43) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**,

(44) Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in PyTorch. NIPS Autodiff Workshop. 2017.

(45) Yutaka, S. The truth of the F-measure. *Teach Tutor mater* **2007**, *1*, 1–5.

(46) Buda, M.; Maki, A.; Mazurowski, M. A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks : the official journal of the International Neural Network Society* **2018**, *106*, 249–259.

(47) Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* **2018**,

(48) Liu, X.; He, P.; Chen, W.; Gao, J. Multi-Task Deep Neural Networks for Natural Language Understanding. *CoRR* **2019**, *abs/1901.11504*.