Supplementary Information (SI) for Digital Discovery.

Spectra To Structure: Contrastive Learning Framework for Library Ranking and Generating Molecular Structures for Infrared Spectra

Ganesh Chandan Kanakala Bhuvnaesh Sridharan U. Deva Priyakymar *

International Institute of Information Technology Hyderabad
*deva@iiit.ac.in

October 6, 2024

Supporting Information

Contents

1 Data and code availability **3**

2 Model Architecture **3**
- 2.1 Spectra Encoder architecture . . . 3
- 2.2 Molecule Encoder architecture . . . 3
- 2.3 Molecule Decoder architecture . . . 4

3 Training **5**
- 3.1 Effect of window size . . . 5
- 3.2 Effect of Batch size . . . 5

4 Embedding Space **6**

5 SMEN Ranking examples **7**

6 SMEN-SD Decoded examples **7**

7 Chemprop-IR for QM9 **8**

8 Effect of Normalization **8**

9 Exploring SMEN's latent space **8**

10 Evaluating on NIST Quantitative Infrared Dataset **9**

1 Data and code availability

The code used for this study is available in github.com/devalab/Spectra2Structure, attached with the supplementary information available. The processed data is available in link.

2 Model Architecture

The entire model architecture comprises of a Spectra encoder, a molecule encoder, and a molecule decoder. The details for the architectures and parameters are shown as follows.

2.1 Spectra Encoder architecture

Figure S1 shows the spectra encoder architecture. First, a spectrum is divided into smaller windows; each window is featurised using a simple Feed forwarded neural network. The obtained features are used to train a transformer model followed by a feed-forward neural network to obtain the embedding in the shared latent space.

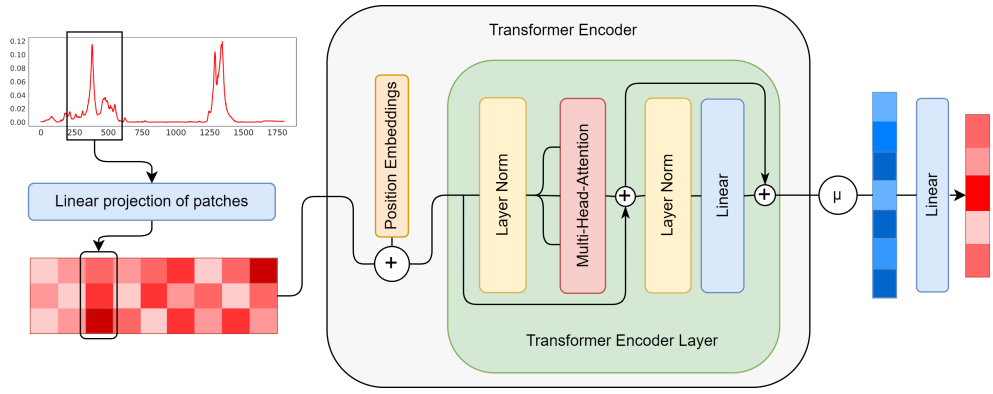

Figure. S1: Spectra encoder architecture

2.2 Molecule Encoder architecture

We used the original authors' egnn code provided in https://github.com/vgsatorras/egnn. We take the EGNN architecture and augment the model with a projection head (a simple feed-forward neural network) that projects the sum pooling of all the node embeddings in a molecule graph to the latent space to match the corresponding spectra embedding dimension.

We consider a molecular graph $G = (V, E)$ with nodes $v_i \in V$ and edges $e_{ij} \in E$. Each node has a node feature representation $h_i \in \mathbb{R}^{nf}$ where h_i is a nf-dimensional node feature vector for i^{th} node and $x_i \in \mathbb{R}^n$ for n-dimensional coordinate for each graph node. EGNN consists of several Equivariant Graph Convolutional Layer (EGCLs), which take input as the set of node embeddings $h^l = \{h^l{}_0, h^l{}_1, ...h^l{}_{k-1}\}$ and coordinate embeddings $x^l = \{x^l{}_0, x^l{}_1, ..x^l{}_{k-1}\}$ and edge embeddings and outputs a new transformation h^{l+1}. Each EGCL layer does the following transformations.

$$m_{ij} = \phi_e(h_i^l, h_j^l, \|x_i - x_j\|^2, e_{ij}) \quad (message\ passing)$$

$$m_i = \sum_{j \neq i} m_{ij} \quad (message\ aggregation)$$

$$h_i^{l+1} = \phi_h(h_i^l, m_i) \quad (node\ feature\ update)$$

ϕ_h and ϕ_e are node and edge operations approximated by Multi Layer perceptions. The outputs of the final EGCL layer are aggregated together and given to a projection head, similar QM9 regression taskss@ in EGNN[1]. We have used EGNN[1] implementation and QM9 data processing for this study from https://github.com/vgsatorras/egnn.

2.3 Molecule Decoder architecture

To generate molecules from the multimodal latent space, we use a GPT-like approach. We use similar approach to MolGPT[2] .We use the spectra embedding obtained from the spectra encoder as the start token, which is passed along with the tokenized molecular SMILES. We apply a causal mask on the input to train the transformer and prevent the model from looking into the upcoming tokens. This ensures that the model learns to predict the next token using only the start token and the tokens generated so far. During the sampling, we first feed the start token and iteratively use the predicted tokens as the input for the next token prediction task till the model generates an end token or reaches a maximum sequence length of 70 characters (upper bound for largest SMILES molecule in the dataset). We use different sampling methods for the next token prediction and compare the results.

The transformer decoder network comprises 3 "decoder" modules, where each decoder module consists of a Layer norm layer, a Masked multi-head attention layer, and a feed-forward layer. Given the latent space of any embedding of size 512, it is concatenated with the token embeddings of the SMILES string to create an input representation of $70 * 512$. Each input is added with a positional encoding[3] to aid the model in understanding the relative positions of the SMILES characters. The newly created input representation is given to the transformer decoder network. Each decoder block processes the input data by applying a layer norm first followed by a Masked multi-head attention module, with a feed-forward neural network at the end, giving a feature representation with the same size as input. The attention mechanism used here is the Scaled-dot product attention, the same as that used by Vaswani et al.[3] represented by:

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

where Q, K, and V are Query, Key, and Value vectors. d_k is the dimension of the input representation. However, the attention mentioned above method "attends" to all the tokens in the input representation while training the neural network. Ideally, we would not want the network to attend to tokens to be predicted while predicting the current token. For this, we use a causal mask, which makes the attention weights of future tokens (tokens on the right side of the current token) 0. This ensures that the model trains to predict the next tokens by only looking at previously predicted tokens. Complete details are provided by Vaswani et al.[3]

The output of the last decoder layer is given to a final projection layer, which has the output size of the vocabulary size, which is used for the final next-token classification. The entire network is trained with cross-entropy loss, where each next token prediction can be considered a multi-class classification.

3 Training

We use an EGNN with 5 Equivariant Graph Convolution layers (EGCL) for the molecule encoder with 256 hidden features. As for the spectra encoder, we settled with a window size of 7, 7 attention heads, and 5 Transorfmer encoder layers. For the generative transformer, we used 4 attention heads and 3 transformer decoder layers, each with 512 hidden neurons. The entire network was trained with 2 NVIDIA GeForce RTX 3090 GPUs with a batch size of 400, for 1000 epochs on an 8:2 train-test split, with AdamW optimizer, a learning rate of 0.0001 equipped with gradient clipping.

3.1 Effect of window size

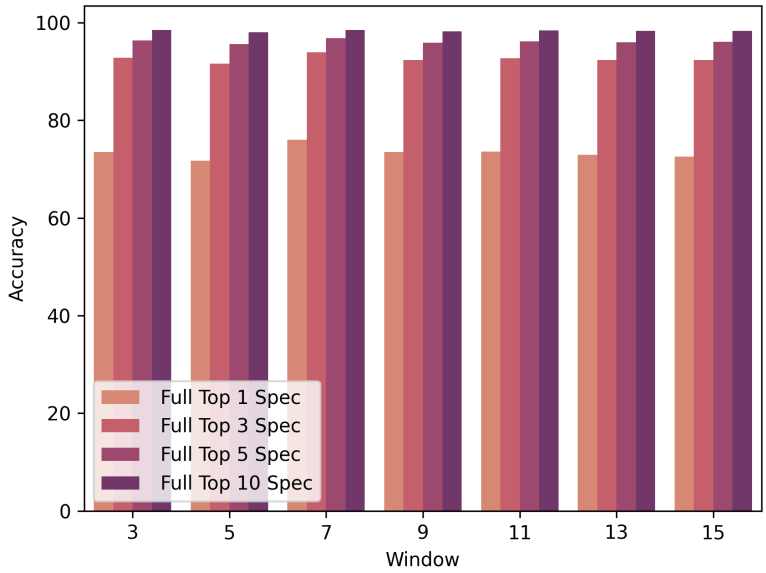

Figure. S2: Variation of model performance across various window sizes

One interesting parameter we use to featurize the IR spectra is the window or the *patch size*. This determines the resolution of the spectral data fed into the transformer model. We experiment with various window sizes for window size $w \in [3, 5, 7, 9, 11, 13, 15]$ reported in Figure S2. Although we do not notice any specific trend in performance, a window length of 7 worked the best for our case. We expected the accuracy to be higher for smaller window lengths due to their higher resolution, but that was not the case. We believe that a window length of 7 here can capture relevant spectral features with sufficient details.

3.2 Effect of Batch size

Several studies have shown the significance of batch size as a parameter for contrastive learning tasks. Following the same tradition, we explore the performance of our model across various batch sizes

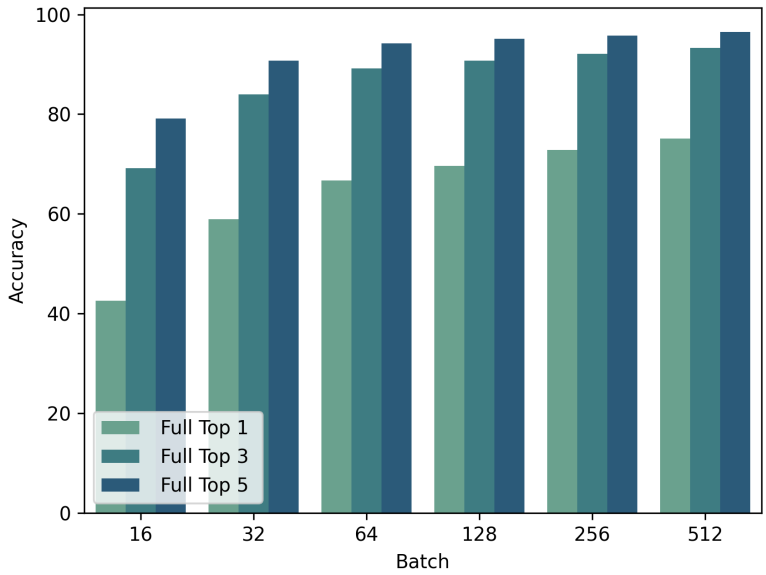

Figure. S3: Variation of model performance across various Batch sizes

$B \in [16, 32, 64, 128, 256, 512]$ as reported in Figure S3. We clearly see the trend in increasing top-k scores with the increase in batch size. Due to computing resource limitations, we could not explore the effect of an even bigger batch size. However, the figure does suggest that the increase in performance appears to reach a saturation point with the increase in batch size. Such results were expected because the contrastive optimization algorithm heavily depends on batch size; the bigger the batch size, the more accurately the model can contrast them in higher dimensional space.

4 Embedding Space

High Top-K accuracies suggest that the model has learned sound representations, grouping similar entities together in latent space. For this study, we used an embedding dimension of 512. Since embeddings are unit normalized and hence represent a higher dimensional unit sphere. To aid the visualization of the embedding space, we project the embeddings onto a plane using UMAP(dimensionality reduction algorithms) showcased in Figure S4. The colored dots with larger marker sizes are a pair of molecules and their corresponding spectra. The smaller blue dots are other samples of molecules and spectra embeddings obtained by SMEN. We can see that the colored dots with larger marker sizes are closer to each other in the embedding space. This highlights SMEN's contrasting capabilities to embed a molecule and its spectra. Although in the Figure, it may seem that there are several other blue dots closer to the molecule-spectra pair, but this is simply a limitation of visualizing embeddings in a low-dimensional 2D space.

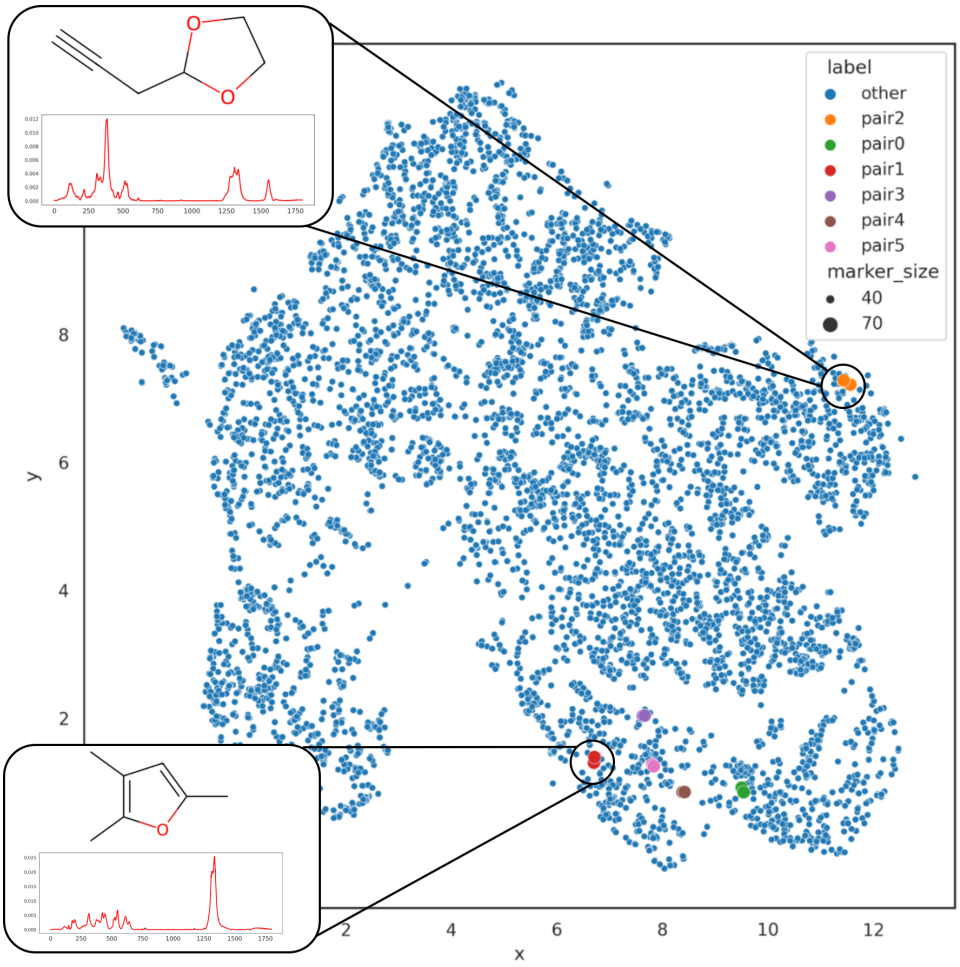

Figure. S4: visualization of embedding space

5 SMEN Ranking examples

Some examples of top-ranked molecules examples are shown in Figure S10.

6 SMEN-SD Decoded examples

Some examples of SMEN-SD molecule generation are shown in Figure S11 and S12. These figures illustrate that random sampling helps us generate a more diverse set of molecules.

7 Chemprop-IR for QM9

For this study, we used Chemprop-IR[4] as a baseline to compare our model ranking performance. Chemprop-IR has shown state-of-the-art performance in predicting the spectra for a given molecule. To address the inverse problem, one can use Chemprop-IR to generate spectra for a large library of molecules and use the SID score between the predicted spectra and target spectra proposed in Chemprop-IR as the metric for ranking. We employ the same technique for this study and compare the ranking accuracy. We used the publicly available codebase for Chemprop-IR to reproduce the results and compare.

8 Effect of Normalization

To validate the effect of normalization of the broadened spectra, we perform the same experiments for lesser parameters and 500 epochs without normalizing the spectra and min-max normalization. However, we notice that there is no significant difference in the performance of the model 1. This indicates that normalization has little to no effect on the results or model performance.

	top-1	top-5	top-10	Greedy decoding Accuracy
Unit Normalization	72 %	96 %	98 %	51 %
Min-max Normalization	71 %	95 %	97 %	49 %
No Normalization	74 %	97 %	98 %	44 %

Table 1: Top K scores for scoring Test set molecules and entire QM9 molecules(Full) against test set spectra

9 Exploring SMEN's latent space

Since, SMEN's embeddings are unit normalized to form a high dimensional unit sphere, we would like to explore this higher dimensional latent space. For this, we perform walks along two anchor spectral embeddings (Example Figure S5). We sample the molecules along the walk using the Decoder network. This allows us to see the molecule's transition from one point in the latent space to another, which may lead to exciting insights into how going from one spectra to another changes the decoded molecules gradually.

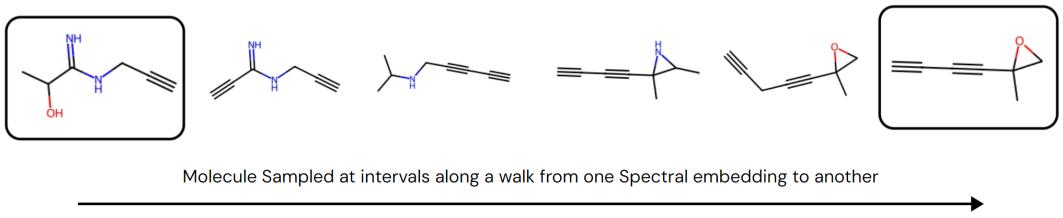

Figure. S5: Sampling molecules along a line in the latent space.

10 Evaluating on NIST Quantitative Infrared Dataset

Due to lack of enough experimental IR data, we test the SMEN's ranking capabilities on ranking the molecules present in Quantitative Infrared Dataset https://webbook.nist.gov/chemistry/quant-ir/. We remove all the molecules with atoms other than C,N,O,H and F so that we end up having QM9-like molecules. We test SMEN on the remaining 27 molecules. We expected a poor performance due to the differences in experimental and simulated spectra, and also because SMEN has not seen any experimental spectra and has to rank the 27 molecules solely based on the data it learned from simulated data. The results are shown in Figure S6, S7. The results indicate that 7 out of 27 were ranked correctly, 17 out of 27 were ranked in the top 5. This indicates that although SMEN is not trained on experimental data, to some extent, it still is capable of ranking them. We believe that upon fine-tuning on experimental datasets, this will improve significantly, making SMEN readily usable for real-world examples.

References

[1] V. G. Satorras, E. Hoogeboom and M. Welling, *Proceedings of the 38th International Conference on Machine Learning*, PMLR, 2021.

[2] V. Bagal, R. Aggarwal, P. K. Vinod and U. D. Priyakumar, *J. Chem. Inf. Model.*, 2022, **62**, 2064–2076.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin, *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017.

[4] C. McGill, M. Forsuelo, Y. Guan and W. H. Green, *J. Chem. Inf. Model.*, 2021, **61**, 2594–2609.

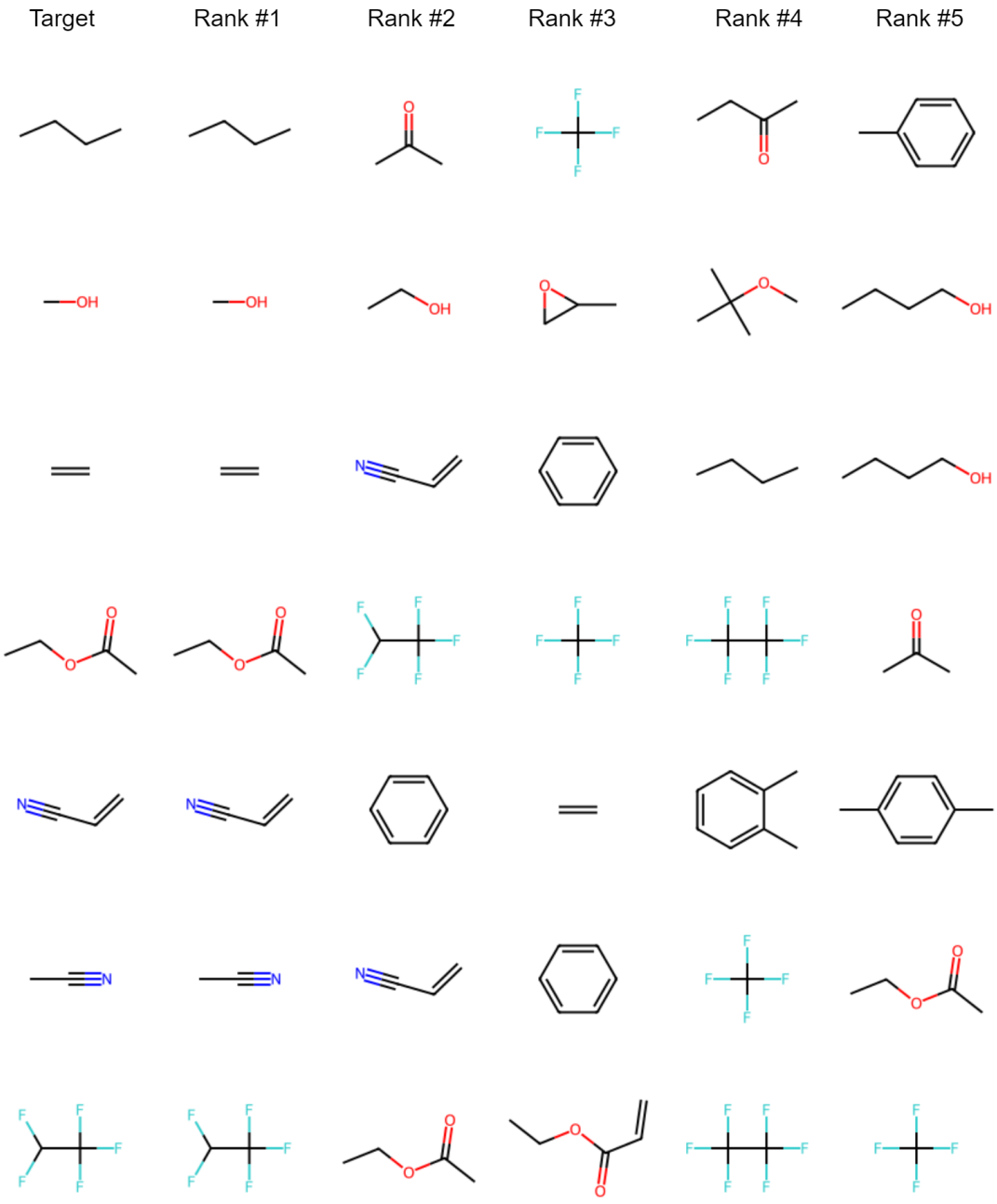

Figure. S6: Seven out of the 27 molecules ranked, SMEN ranked the correct molecule in rank 1.

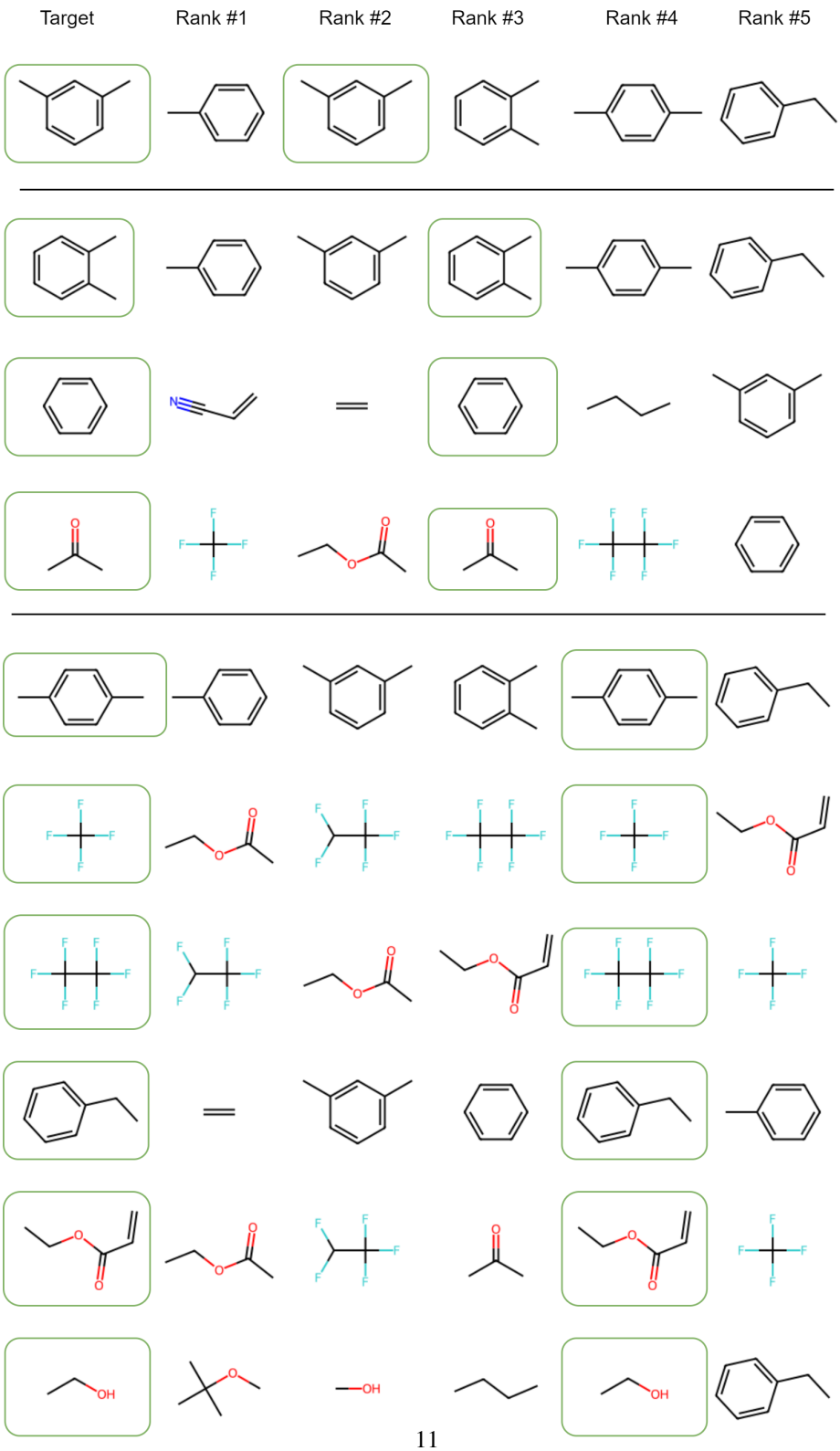

Figure. S7: Ranking results for molecules ranked in 3rd, 4th and 5th positions.

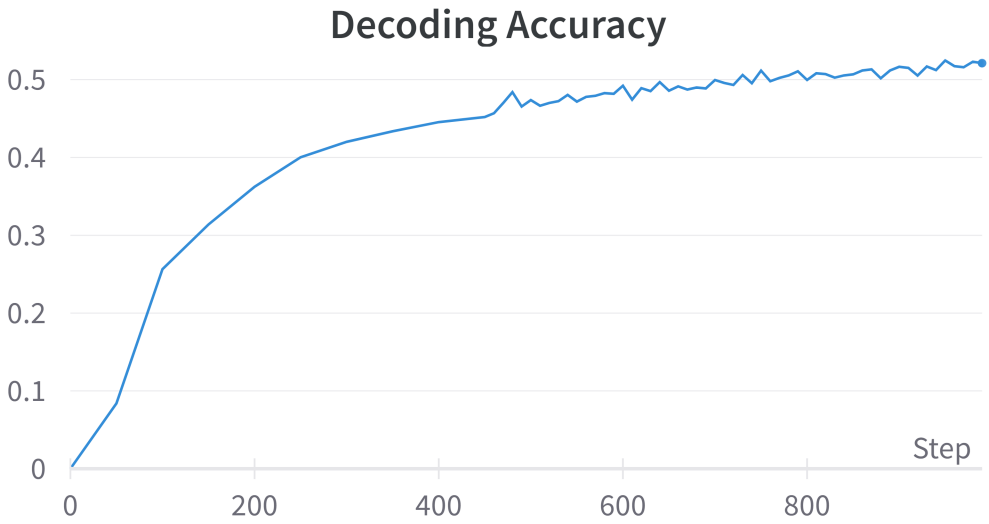

Figure. S8: SMEN-SD target molecule generation accuracy across epochs.

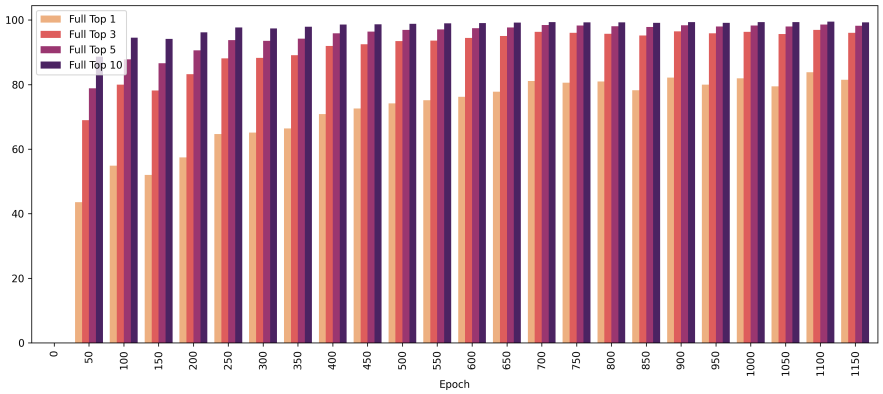

Figure. S9: Model performance across epochs.

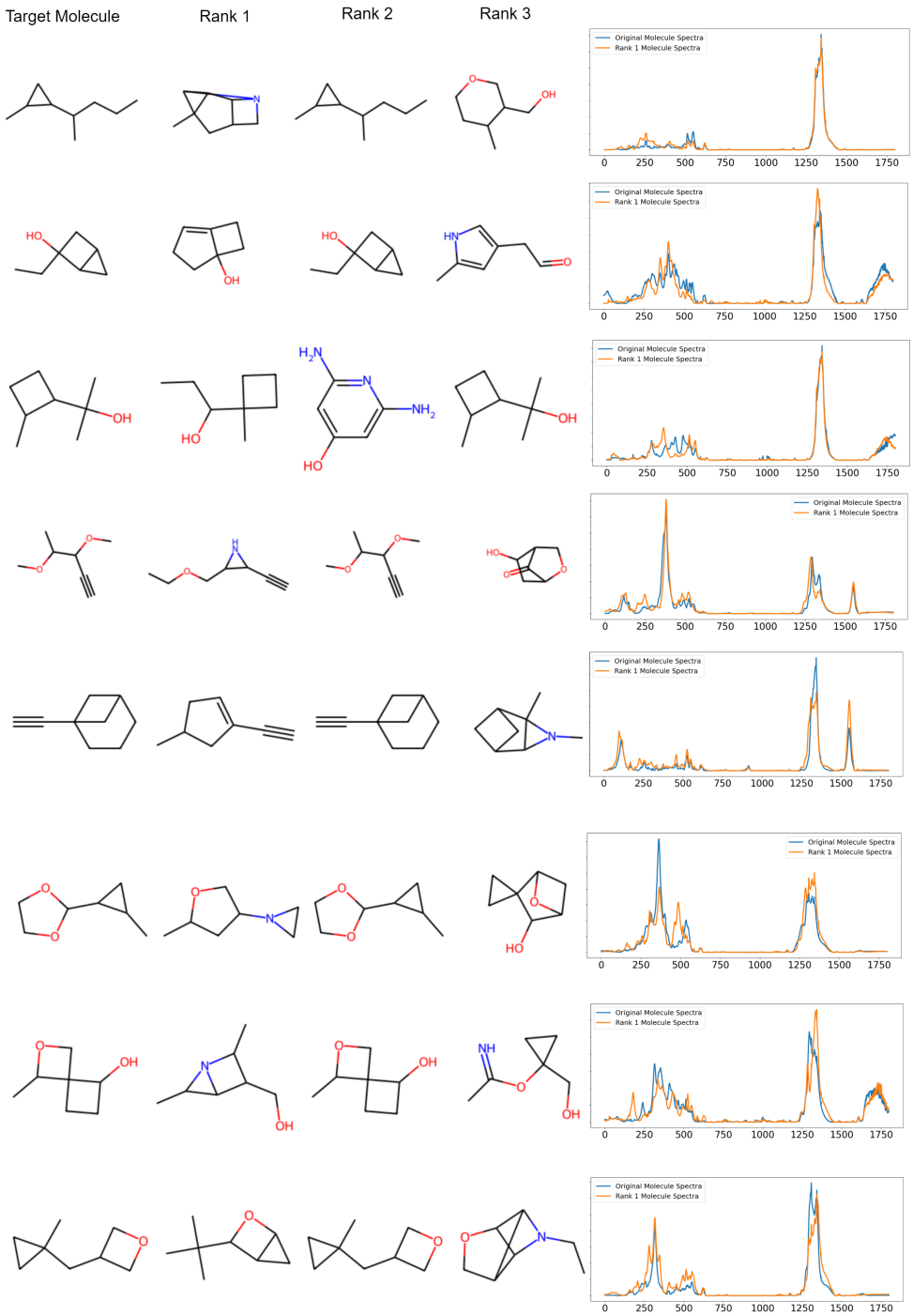

Figure. S10: Examples illustrating that sometimes even if the top-ranked molecule is not similar to the target molecule, they have common characteristic functional groups and end up having very similar spectra

.

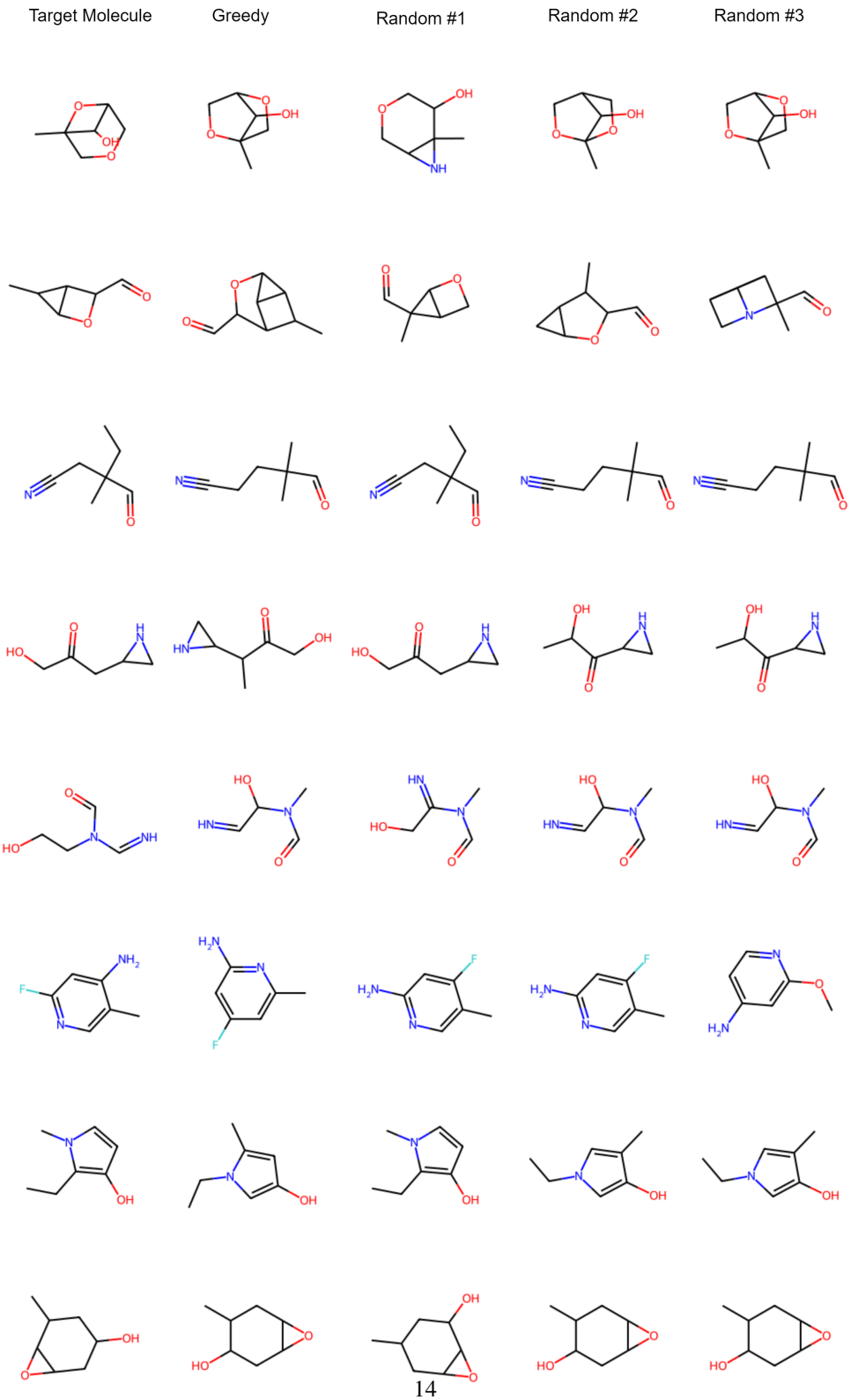

Figure. S11: Examples illustrating the comparison between different sampling methods used for generative transformer

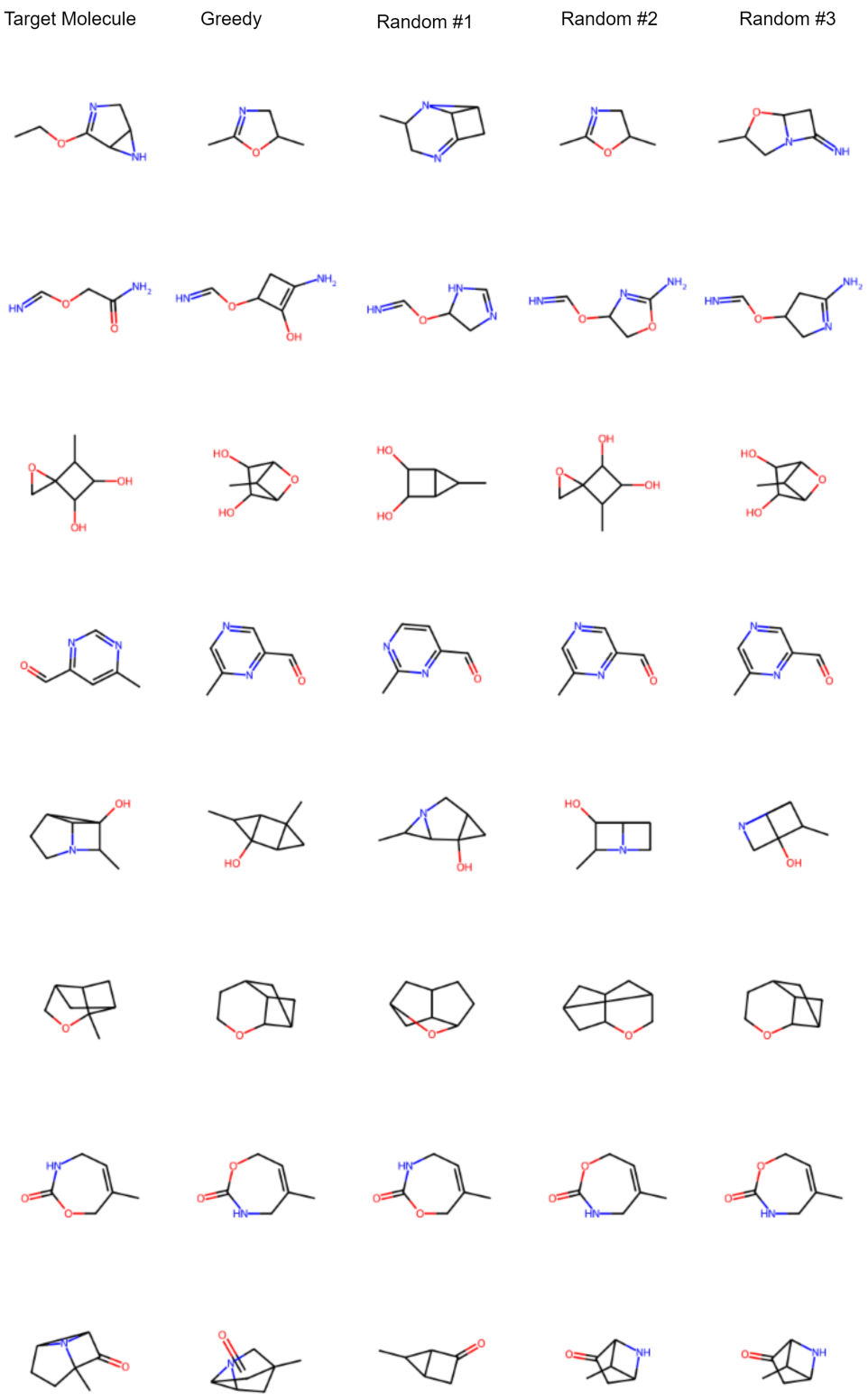

Figure. S12: More Examples illustrating the comparison between different sampling methods used for generative transformer