Kernel-Elastic Autoencoder for Molecular Design

Haote Li, Yu Shee, Brandon Allen, Federica Maschietto, Victor Batista
Department of Chemistry, Yale University, New Haven, Connecticut 06520, USA.

Contributing authors: haote.li@yale.edu; yu.shee@yale.edu; brandon.allen@yale.edu; federica.maschietto@yale.edu; victor.batista@yale.edu;

Abstract
We introduce the Kernel-Elastic Autoencoder (KAE), a self-supervised generative model based on the transformer architecture with enhanced performance for molecular design. KAE is formulated based on two novel loss functions: modified maximum mean discrepancy and weighted reconstruction. KAE addresses the long-standing challenge of achieving valid generation and accurate reconstruction at the same time. KAE achieves remarkable diversity in molecule generation while maintaining near-perfect reconstructions on the independent testing dataset, surpassing previous molecule-generating models. KAE enables conditional generation and allows for decoding based on beam search resulting in state-of-the-art performance in constrained optimizations. Furthermore, KAE can generate molecules conditional to favorable binding affinities in docking applications as confirmed by AutoDock Vina and Glide scores, outperforming all existing candidates from the training dataset. Beyond molecular design, we anticipate KAE could be applied to solve problems by generation in a wide range of applications.

1 Introduction
The advent of generative models has precipitated a revolutionary shift in the development of methods for drug discovery, revealing new opportunities to swiftly identify ideal candidates for specific applications [1–7]. The Variational Autoencoder (VAE) model has emerged amongst these models as an approach with extraordinary capabilities that can be adapted for molecule generation via character, grammar, and graph-based representations [8–11]. Here, we introduce the Kernel-Elastic Autoencoder (KAE) which is a self-supervised generative model based on a modified maximum mean discrepancy loss and weighted reconstruction loss functions. With these improvements, KAE demonstrates its capabilities as applied to molecular design and molecular docking.

Autoencoders (AEs) typically encode the input data by compression into a low-dimensional latent space [12]. However, such latent space can include regions devoid of physical meaning when the encoded data does not map to those regions, thereby limiting the generative capabilities of the model. VAEs mitigate that challenge by sampling from encoded latent vectors in known distributions. Upon decoding samples from those distributions, VAEs generate outputs mirroring the training data, such as molecules with desired properties. An outstanding challenge of great interest is to harness the power of VAEs to generate molecular candidates with optimal properties during the initial screening phase of molecular development, a process that can facilitate early success for drug discovery.

VAEs are typically evaluated for molecule generation using novelty (N), uniqueness (U), validity
(V), and reconstruction (R) metrics. NUVR metric, which is the product of them, captures the trade-off between these four factors—i.e., the so-called NUV to R tradeoff [13], as a model with high reconstruction ability usually does not achieve high metrics for novelty, uniqueness, and validity. Kernel-Elastic Autoencoder (KAE) model outperforms state-of-the-art methods on the NUVR metric, achieving nearly perfect reconstruction while maintaining high novelty, uniqueness, and validity scores (Figure 1).

Generative models can be used to optimize the design of molecular motifs with respect to a desired property by sampling from latent space near a reference molecule, conditioned on the property of interest. This requires robust reconstruction, as proximity in latent space should correlate with proximity in the value of the desired property. Accurate reconstruction also allows for interpolation between molecular motifs with intermediate properties between promising lead compounds [19–22]. The enhanced performance of KAE is therefore particularly valuable for both molecular optimization and interpolation since it can be used to generate new molecular motifs with desired properties or to interpolate between existing molecular motifs to find new lead candidates.

Graph-based VAE methodologies take the lead in ensuring chemical validity [11, 23]. By representing molecules as graphs with distinct motifs, these models enforce grammatical rules, ensuring the validity of generated molecules. Yet, when building motifs are absent in the training dataset, they stumble to reconstruct or generate molecules similar to those motifs.

Flow-based generative models have been demonstrated to excel in reconstruction [14, 16]. These models employ invertible mapping to transform input data to a latent space of identical dimensions, thereby precisely mapping latent vectors back to their origins. However, flow-based models have the same input and latent dimensions which lead to out-of-distribution sampling problems with limited reconstruction performance [24].

KAE (Figure 2) effectively overcomes the NUV-R tradeoff by combining the advantages of both autoencoder (AE) and variational autoencoder (VAE) models with an attention-based transformer architecture [25]. KAE’s loss function is a modified version of the Maximum Mean Discrepancy (MMD) [26] that shapes the latent space and enables better performance than using Kullback-Leibler (KL) divergence loss commonly used in VAEs. When coupled to weighted reconstruction and beam search [27, 28] decoding techniques, KAE, without any checking for molecular grammar or chemical rules, outperforms both string and graphical-based models (Figure 1) in generation tasks while exhibiting nearly flawless reconstruction, as demonstrated on 24,000 molecules from the ZINC250k testing sets.

Beam search also enhances sample diversity, showing that multiple valid interpretations of the same latent vectors can be derived through the beam search process.

When implemented to solve optimization problems, KAE outperforms the state-of-the-art by
Fig. 2: KAE transformer architecture. KAE consists of 6 encoder and 6 decoder components (highlighted in red and blue, respectively) gradient color in the latent space represents the connection between the encoder and decoder. The condition is depicted in grey. The vector obtained after the compression layers is referred to as the latent vector. During training, a Gaussian noise is added, resulting in the final encoder output. The $\lambda$ and $\delta$ terms are used to control the KAE loss. If the Conditional KAE (CKAE) structure is employed, condition-scaled embeddings are concatenated with both the latent vector after applying noise and the encoder input. In KAE, all conditions are set to zero during training. The decoder output is then passed through a linear layer and softmax function, producing the probabilities of output tokens for each character in the dictionary of size $T$.

2 Results and Discussions

2.1 KAE Performance

The overall performance of the KAE (Figure 2) compared to state-of-the-art generative models is shown in Table 1. As described in the Methods section, KAE combines a modified-MMD (m-MMD) loss and the Weighted Cross Entropy Loss (WCEL), with hyperparameters $\lambda$ and $\delta$, and exhibits the generative capabilities of VAEs as well as the exact reconstruction objectives of AEs. KAE was evaluated according to the fraction of generated molecules that are novel (N), unique (U), and valid (V). A molecule is considered novel if it is not included in the training dataset. Uniqueness is defined as the absence of duplicates in the set of generated molecules. A molecule is counted as valid if its SMILES representation is syntactically correct and passes the RDKit chemical semantics checks [32]. Additionally, reconstruction is successful if and only if the decoder regenerates the input SMILES sequence matching every single character.

Maximum validity and reconstruction was achieved by using the Weighted Cross-Entropy Loss $L_{WCEL}(\lambda, \delta)$ defined by Eq. (4) where the hyperparameter $\delta$ controls the AE-like objective (see S.I. for a discussion of the effect of changing $\lambda$ and $\delta$). The best results for the NUVR metric were obtained by using a combination of $\lambda = 3.5$ and $\delta = 1$. A substantial 28% [18]. Additionally, KAE tackles the problem of molecular docking by finding suitable binding ligands by conditional generation, as demonstrated for the training dataset of GFlowNet [5]. Superior candidates from the baseline and the training data are independently verified by both Autodock Vina [29] and Glide [30, 31], demonstrating its efficacy and practicality.
Table 1: Comparison of performance of molecular generative models trained with the ZINC250K dataset. Assessment of the capabilities of the models to generate novel (N), unique (U), valid (V), and properly reconstructed (R) molecules. Validity (V w/o) indicates that the generated strings have not been post-processed using chemical knowledge to enforce corrections. NUV results were obtained from averaging over 5 iterations of sampling 10,000 random vectors from latent space while the reconstruction rate was calculated using all molecules from the testing dataset. The two KAE models in the table were trained using loss functions with \( \lambda = 1 \) and 3.5 and \( \delta = -1 \) and 1. The choice of \( \delta = -1 \) is a special case of WCEL and is equivalent to not using any AE objectives. The validity check selects alternative candidates from the beam search process.

<table>
<thead>
<tr>
<th>Method</th>
<th>N</th>
<th>U</th>
<th>V w/o</th>
<th>V</th>
<th>NUV</th>
<th>R</th>
<th>NUVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVAE [9]a</td>
<td>0.980</td>
<td>0.021</td>
<td>0.007</td>
<td>N/A</td>
<td>0.0001</td>
<td>0.446</td>
<td>5e-6</td>
</tr>
<tr>
<td>GVAE [10]a</td>
<td>1.000</td>
<td>1.000</td>
<td>0.072</td>
<td>N/A</td>
<td>0.072</td>
<td>0.537</td>
<td>0.039</td>
</tr>
<tr>
<td>JTVAE [11]a</td>
<td>1.000</td>
<td>1.000</td>
<td>0.935</td>
<td>1.000</td>
<td>0.935</td>
<td>0.767</td>
<td>0.177</td>
</tr>
<tr>
<td>MoFlow [14]</td>
<td>1.000</td>
<td>0.999</td>
<td>0.818</td>
<td>1.000</td>
<td>0.817</td>
<td>1.000b</td>
<td>0.817</td>
</tr>
<tr>
<td>Rebalanced [15]</td>
<td>1.000</td>
<td>1.000</td>
<td>0.907</td>
<td>0.938</td>
<td>0.907</td>
<td>0.927</td>
<td>0.841</td>
</tr>
<tr>
<td>GraphDF [16]</td>
<td>1.000</td>
<td>0.992</td>
<td>0.890</td>
<td>1.000</td>
<td>0.883</td>
<td>1.000b</td>
<td>0.883</td>
</tr>
<tr>
<td>ALL SMILES [17]a</td>
<td>1.000</td>
<td>1.000</td>
<td>N/A</td>
<td>0.985</td>
<td>N/A</td>
<td>0.874</td>
<td>N/A</td>
</tr>
<tr>
<td>( \beta )-VAE [18]</td>
<td>0.998</td>
<td>0.983</td>
<td>0.983</td>
<td>0.988</td>
<td>0.964</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>KAE (( \lambda = 1, \delta = -1 ))</td>
<td>0.998</td>
<td>0.994</td>
<td>0.863</td>
<td>N/A</td>
<td>0.856</td>
<td>0.992</td>
<td>0.849</td>
</tr>
<tr>
<td>KAE (( \lambda = 3.5, \delta = 1 ))</td>
<td>0.996</td>
<td>0.973</td>
<td>0.997</td>
<td>1.000</td>
<td>0.966</td>
<td>0.997</td>
<td>0.963</td>
</tr>
</tbody>
</table>

\( a \)Results obtained from sampling 1,000 vectors from latent space.

\( b \)Reconstruction rates were obtained on training datasets.

With the same Transformer architecture, KAE is compared to approaches using different loss functions (SI, Figure S1) by assessing the validity and reconstruction. KAE trained with Gaussian noise added to latent space vectors exhibited the highest percentage of valid SMILES strings, while models trained with the KL divergence exhibited much lower validity and significantly slower improvements for validity during training. The analysis of novelty and uniqueness showed that models with noise (i.e., with Gaussian noise added to latent space vectors) performed much better than the corresponding models without noise when trained with the standard MMD (s-MMD) or modified MMD (m-MMD, see method section). Additionally, the NUV metric showed that models trained with m-MMD outperformed models trained with s-MMD. In summary, the KAE model trained with m-MMD loss and Gaussian noise added to latent space vectors achieved the best performance on all three metrics (validity, reconstruction, and NUV).

2.2 Conditional KAE

In this section, the performance of the Conditional-KAE (CKAE) (Figure 2) on the constraint optimization task is investigated. CKAE generates new candidates conditioned on the property of interest. Here, we demonstrate the capabilities of CKAE as applied to the PLogP values defined, as follows: [9, 11]:

\[
P\text{LogP}(m) = \log\text{P}(m) - \text{SA}(m) - \text{ring}(m),
\]

where \( \log\text{P} \) is the octanol-water partition coefficient of molecule \( m \) calculated using Crippen’s approach from the atom contributions [37]. SA is the synthetic accessibility score, [38] while \( \text{ring}(m) \) corresponds to the number of rings with more than six members in the molecule.

To demonstrate that CKAE generates molecules that are strongly correlated to the conditioned value, we analyzed the correlation between the properties of CKAE-generated molecules and the specified input condition. Figure 3 shows the mean PLogP value obtained from 1,000 CKAE-generated molecules, strongly correlated to the PLogP value used as a condition (correlation
Table 2: Comparison of performance of various conditional generative models. The table presents the average PLogP improvements computed for the set of 800 lowest ranking molecules from the ZINC250K dataset, as well as the mean Tanimoto similarities of the best candidate molecules compared to their respective starting molecules. The success rate indicates the percentage of molecules for which the algorithm successfully achieved modifications resulting in higher PLogP values within the specified similarity constraint. The ZINC250K result corresponds to the highest PLogP improvement obtained by searching within the ZINC250K dataset itself. Our approach outperforms the search against the training data and demonstrates the highest performance when combining our model with the SES method.

<table>
<thead>
<tr>
<th>Method</th>
<th>PLogP-Improvement</th>
<th>Tanimoto Similarity</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>JT-VAE [11]</td>
<td>0.84 ± 1.45</td>
<td>0.51 ± 0.1</td>
<td>83.6%</td>
</tr>
<tr>
<td>MHG-VAE [33]</td>
<td>1.00 ± 1.87</td>
<td>0.52 ± 0.11</td>
<td>43.5%</td>
</tr>
<tr>
<td>GCPN [34]</td>
<td>2.49 ± 1.30</td>
<td>0.47 ± 0.08</td>
<td>100%</td>
</tr>
<tr>
<td>Mol-CycleGAN [1]</td>
<td>2.89 ± 2.08</td>
<td>0.52 ± 0.10</td>
<td>58.75%</td>
</tr>
<tr>
<td>MOLDQboot [35]</td>
<td>3.37 ± 1.62</td>
<td>N/A</td>
<td>100%</td>
</tr>
<tr>
<td>ZINC250K (This work)</td>
<td>4.64 ± 2.33</td>
<td>0.48 ± 0.16</td>
<td>97.88%</td>
</tr>
<tr>
<td>MoFlow [14]</td>
<td>4.71 ± 4.55</td>
<td><strong>0.61 ± 0.18</strong></td>
<td>85.75%</td>
</tr>
<tr>
<td>Random Sample (This work)</td>
<td>4.78 ± 2.08</td>
<td>0.43 ± 0.03</td>
<td>81.75%</td>
</tr>
<tr>
<td>MNCE-RL [36]</td>
<td>5.29 ± 1.58</td>
<td>0.45 ± 0.05</td>
<td>100%</td>
</tr>
<tr>
<td>β-VAE [18]</td>
<td>5.67 ± 2.05</td>
<td>0.42 ± 0.02</td>
<td>98.25%</td>
</tr>
<tr>
<td>CKAE (This work)</td>
<td><strong>7.67 ± 1.61</strong></td>
<td>0.42 ± 0.02</td>
<td><strong>100%</strong></td>
</tr>
</tbody>
</table>

a) Tanimoto similarity constraint = 0.4

The distribution of PLogP values of the training set, rendered as a histogram in Figure 3, shows the range of PLogP values used for CKAE training.

Instead of using regressors to navigate in the latent space[11, 15, 18, 39], a procedure called Similarity Exhaustion Search (SES) was developed for constraint optimizations. SES aims to find molecules that are both similar to the target molecule and have higher desired properties (e.g., PLogP) by using the same or slightly perturbed latent vector representations with gradually increasing conditions. Importantly, the CKAE output is robust even if the condition or latent vector is slightly changed. Formally, \( f(z, c) \approx f(z + \Delta_z, c + \Delta_c) \) for small values of \( \Delta_z \) and \( \Delta_c \) when \( f(x, c) \) is the decoding output function of latent vector \( z \) subject to the constraint of \( c \) (e.g., PLogP = \( c \)). When the generative model has high enough NUVR values, it is able to pinpoint the exact latent vector location and perform an exhaustive search for all possible \( \Delta_c \). Therefore, SES combines beam search with iterative sampling under various conditions to identify chemically similar molecules that closely resemble the target compound in the latent space. The details of SES can be found in Section 3.5.

Table 2 shows 1. the results of optimizing the 800 lowest PLogP-valued molecules from the ZINC250K dataset to generate similar molecules (Tanimoto similarity < 0.4) with larger PLogP values; [35]; 2. The mean difference in PLogP values; 3. The Tanimoto similarity between the best candidate molecules and their starting molecules for each method. The success rate measures the percentage of molecules that achieved modifications with higher PLogP values within their similarity constraints. The CKAE model exhibits significant improvements, with an average increase of 7.67 ± 1.61 PLogP units. CKAE achieves a 100% success rate, indicating that modifications leading to higher PLogP values were successfully achieved for all molecules within the defined similarity constraints.

Additionally, CKAE performance was assessed as compared to direct search from the ZINC250K training set. For each of the 800 molecules, its similarity value with respect to all other 250K entries was calculated, and the compound with the highest PLogP value that remained within the 0.4 Tanimoto similarity constraint was identified. This particular outcome is labeled “ZINC250K” in Table 2.
2.3 CKAЕ for Ligand Docking

2.3.1 Comparison to GFlowNet

Table 3 shows the performance of the CKAЕ model as applied to the generation of small molecule inhibitors that bind to the active site of the enzyme soluble epoxide hydrolase (sEH), as compared to results obtained with GFlowNet for the same active site [5, 40].

CKAE was trained using the same dataset of 300,000 molecules used for training GFlowNet [41], each with a binding energy calculated using AutoDock [29] (see Section 3.6). Binding energies were converted to a reward metric, using a custom scaling function. Results in Table 3 correspond to the mean reward for the top 10, 100 and 1000 best scoring molecules from a pool of \(10^6\) NUV molecules generated by the CKAЕ model. Rewards were computed from the Autodock Vina binding scores. Average Tanimoto similarities were computed using a Morgan Fingerprint with a radius of 2.

Table 3 shows that CKAЕ achieves similar performance to GFlowNet in molecular docking, and generates molecules with higher rewards at the top 10, 100, and 1000 thresholds, without significantly sacrificing the similarity score. In fact, CKAЕ was able to generate molecules scoring as high as 11.45, which exceeds the maximum reward of 10.72 in the training database. This demonstrates the capabilities of CKAЕ for generative extrapolation, which allows for applications to generative dataset augmentation including molecules with scoring values beyond the range of the original dataset.

2.3.2 Glide Analysis

A comparison of the ligand-receptor interactions established by the top-scoring CKAЕ, training dataset (TD) and GFlowNet candidates, respectively is shown in Figure 4a. KAE’s top candidate exhibits superior docking performance compared to top-scoring candidates in both the training dataset and GFlowNet. In terms of fitting within the pocket, the top CKAЕ candidate occupies a substantially larger volume within the receptor binding region when compared to the other two. The improved fit is also evidenced by the broader array of stabilizing interactions. These interactions include a series of \(\pi-\pi\) stacking and \(\pi\)-cation interactions. In addition to occupying the pocket

---

**Fig. 3: CKAЕ correlation performance.** The blue dots represent the mean PLogP values of 1,000 molecules generated by CKAЕ, as a function of the condition PLogP value. The black line shows the ground truth values strongly correlated with the mean PLogP values. The histogram shows the underlying distribution of the training dataset over the entire range of PLogP values.

It is worth noting that the CKAЕ model exhibits an impressive reconstruction rate of nearly 100%. This indicates that the encoder is able to determine the most accurate representation of the molecule in latent space, which makes searching around the molecules much more efficient. To highlight this advantage, we further compared CKAЕ to direct search using randomly sampled latent vectors with different conditions (e.g., a range of PLogP values from -10 to 10 scanned with a step size of 0.1). At each step, instead of using encoder provided latent vectors, 800 vectors were randomly sampled from the latent space and decoded using beam search with a beam size of 15. The outcomes of this search are marked as “Random Search” in Table 2.

CKAE yielded better results than simply searching over the entire ZINC250K dataset, demonstrating the advantage of using a generative model. Additionally, the CKAЕ protocol samples ten times fewer molecules per target than random search, while achieving better performance, demonstrating the strength of exact reconstructions.
Table 3: Performance of the CKAE model on molecular docking as compared to GFlowNet.
Top 10, 100, and 1000 rewards are the averages of the docking scores of molecules generated at the corresponding thresholds. The Top-1000 similarity is the mean of all pair-wise similarities. Lower similarity between generated molecules indicates greater diversity, which is desirable. For docking, the higher rewards the better.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top 10 reward</th>
<th>Top 100 reward</th>
<th>Top 1000 reward</th>
<th>Top-1000 similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFlowNet</td>
<td>8.36</td>
<td>8.21</td>
<td>7.98</td>
<td>0.44</td>
</tr>
<tr>
<td>Training Data</td>
<td>9.62</td>
<td>8.78</td>
<td>7.86</td>
<td>0.58</td>
</tr>
<tr>
<td>CKAE (This work)</td>
<td>11.15</td>
<td>10.46</td>
<td>9.63</td>
<td>0.63</td>
</tr>
</tbody>
</table>

entirely, the CKAE-generated molecules are devoid of unfavorable clashes, further underscoring the effectiveness of the model in generating effective candidates in the context of molecular docking.

Figure 4b shows the analysis of the best scoring molecules generated by CKAE and direct search from the training dataset (TD), as assessed by the Glide molecular docking program that is an integral part of the Schrödinger Suite of software [30, 31]. Figure 4b thus provides an independent assessment of the quality of the best-scoring CKAE-generated molecules, showing that CKAE-generated molecules outperform the TD counterparts in terms of ranking as determined by the nature of the interactions established at the binding site.

The docking procedure employed an identical receptor grid size as used for Autodock Vina [29] calculations, and the candidates sourced from both the training dataset and CKAE, were docked onto the same receptor structure, using the highest scoring pose derived from Autodock Vina [29] calculations, as described in Section 3.6.1.

A dataset comprised of 869 tautomers was curated with high structural similarity, including the top ten CKAE-derived molecules and the top ten TD molecules, as well as a set of tautomers of the same molecules generated by changing protonation and enantiomeric states to analyze the quality of the top-performing hits relative molecular tautomers (molecules with different arrangements of atoms and bond). The results shown in Figure 4c revealed that the top-ranking candidates from both CKAE and TD outperformed other contenders (tautomers) when compared against the dataset of tautomers. These results confirmed that the highest-scoring molecular structures obtained from CKAE and TD remained superior, even when compared to a large number of structurally similar alternatives, confirming the reliability and quality of molecules generated by CKAE.

As examined by both Autodock Vina [29] and Glide [30, 31], it is clear that CKAE generated molecules that bind better to the active site of sEH than those of the training dataset. The generated higher-scoring molecules can then be used for dataset augmentation, for retraining purposes, allowing the model to generate even higher-scoring molecules.

In summary, KAE allows the integration of the strengths of both VAE and AE frameworks in applications to molecular design. The KAE loss, with hyperparameters $\lambda$ and $\delta$, controls varying degrees of VAE and AE features as needed for the specific applications.

It was found that in the context of molecule generation, KAE outperformed VAE approaches in terms of generation validity without the need for additional chemical knowledge-based checks, while achieving reconstruction performance close to 100% accuracy akin to an AE model. The KAE generative performance is further enhanced through beam search decoding. [42–44] In the context of conditional generation, we found that CKAE generates molecules that exhibit excellent correlation with the input condition, including molecules with a desired property (e.g., specific value of PLogP, or reward value upon docking to a specific binding site of a biological target).

CKAE achieves state-of-the-art performance in molecular design, outperforming direct search from its training dataset by over 65% in PLogP constrained optimization, and 6.8% in molecular docking. Validation of the top-scoring CKAE-generated molecules with Glide [30, 31], reveals
Fig. 4: Glide analysis of molecular inhibitors docked at the active site of sEH. (a) Binding interactions of top scoring molecules generated by CKAE (left), searched from the training dataset (middle), and generated by GFlowNet (right). (b) Extra Precision (XP) Glide score Boltzmann factors for the top ten candidates obtained from the CKAE and training dataset (TD) show that the top ranking CKAE-generated outperform the top molecules from the TD ensemble. (c) Histogram of Glide XP docking scores, showing that top scoring inhibitors generated by CKAE or TD outperform 869 tautomers generated from the top ten candidates of the two datasets.

that they consistently outperform those from the training dataset as well as those from a curated dataset of structurally similar tautomers, demonstrating robust performance of CKAE and high quality of the CKAE-generated molecules. More generally, KAE and CKAE architectures should exhibit powerful generative capabilities with potential applications to solution search by generation in a wide range of problems with pair-wise labeled datasets.
3 Methods

3.1 Model Architecture

KAE treats molecule generation as a natural language processing task. Phrases in the “source language” (i.e., SMILES strings) are encoded and compressed into latent vectors, and then decoded into the target output with corresponding labels. Attention mechanisms are implemented for both the encoding and decoding processes, as depicted in Figure 2. KAE includes multi-head self-attention for the encoder (using 4 heads), a compression layer, an expansion layer, and decoding.

Source and target masks are created with specified padding tokens to ensure that the encoder and decoder do not attend to padding tokens during training. The SMILES tokens are passed through embedding layers in both the encoder and decoder by transforming each token into a vector of embedding size \( E = 128 \).

In the CKAE variant, the conditions (i.e., molecule properties) are attached with additional embeddings. Condition-scaled embeddings are concatenated with the input of the encoder and the latent representation along the sequence length dimension. This allows the model to generate molecules by either interpolating or extrapolating with a desired condition value.

The input is encoded by the Transformer encoder and compressed into latent space. During training, this latent tensor has dimensions of \( 10 \times E \) as resulting from compressing the sequence length to 10 dimensions and keeping the embedding size \( E \). The entries of this tensor are then perturbed with Gaussian noise. Additionally, in the case of CKAE, it is concatenated with the property-scaled embedding vector condition. The resulting latent tensor is then expanded back to dimensions of \( M \times E \) by using the expansion layer, where \( M \) represents the maximum sequence length in the relevant dataset. This expanded tensor serves as the input to be fed into the decoder without supplying encoder masks.

Each decoder layer attends to the encoder outputs through encoder-decoder multi-head attention operations. The decoder outputs are contracted by a linear layer along the embedding dimension, producing a \( T \)-dimensional vector per token. This \( T \)-dimensional vector is then softmaxed, resulting in a probability distribution \( P \) for each possible character (\( c \)).

3.2 KAE Loss

The KAE loss function is defined, as follows:

\[
\mathcal{L}(\lambda, \delta) = \mathcal{L}_{WCEL}(\lambda, \delta) + m\text{-MMD}(\lambda),
\]

where \( m\text{-MMD}(\lambda) \) is a modified version of the regularizing MMD loss, discussed in Sec. 3.3, and \( \mathcal{L}_{WCEL} \) is a weighted cross-entropy loss (WCEL) obtained from outputs generated by decoding the latent vector with and without Gaussian noise added to the latent vector. Based on the original definition of the cross-entropy loss (CEL):

\[
\mathcal{L}_{CEL} = - \sum_{s} \sum_{c} Y_{s,c} \log (P_{s,c}),
\]

where \( P_{s,c} \) is the predicted softmax probability of token \( c \) at sequence position \( s \) and \( Y_{s,c} \) is the ground truth label equal to one if the token belongs to class \( c \) at position \( s \), or zero otherwise. Accordingly, we define \( \mathcal{L}_{WCEL} \), as follows:

\[
\mathcal{L}_{WCEL}(\lambda, \delta) = \frac{-1}{\lambda + \delta + 1} \left[ \sum_{s} \sum_{c} Y_{s,c} \log (P_{s,c}) \right] + (\lambda + \delta) \sum_{s} \sum_{c} Y_{s,c} \log (P^{*}_{s,c}) \right).
\]

where \( P_{s,c} \) and \( P^{*}_{s,c} \) are the predicted softmax values obtained upon decoding the latent vector with and without added Gaussian noise, respectively.

The hyperparameters \( \lambda \) and \( \delta \) control the significance of the second term in the r.h.s. of Eq. (4) (AE behavior) as well as the relative weight between the m-MMD term and the weighted cross-entropy loss, according to Eq. (2). By adjusting \( \lambda \) and \( \delta \), the learning objective can be positioned between the VAE and AE objectives. At the extremes, the objective becomes VAE-like (or AE-like) upon weighting more the term with (or without) noise. For example, when \( \lambda = 1 \) and \( \delta = -1 \), \( \mathcal{L} \) is like the VAE loss except that we use m-MMD instead of the KL-divergence. For AE-like behavior, we choose \( \lambda = 0 \) and \( \delta = 1 \).

The inclusion of \( \lambda \) in the second term of Eq. (4) allows larger \( \lambda \) values to restrict the latent vectors closer together, penalized by the m-MMD loss. This effect increases the probability of sampling valid latent vectors but reduces distinctions.
between vectors. Further details on the effect of \( \lambda \) in Section S4.

During training, both the latent vector and the decoder outputs with and without noise are necessary for the calculation of the KAE loss. The teacher forcing method [45] is implemented for training by penalizing the decoder outputs relative to the labels. On the other hand, the latent vectors are penalized based on their differences from 1000 randomly sampled Gaussian vectors \( \{G_i\} \) using kernel-based metrics. During training, a noise vector \( \epsilon \in \mathbb{R}^D \), with \( D \) the dimension of the latent space, is added to the latent vector before passing it to the decoder. The noise vector is generated from a Gaussian normal distribution \( \mathcal{N}(\mu, \sigma^2) \) with zero mean \( \mu = 0 \) and unit variance \( \sigma = 1 \).

The center of the Figure 2 captures the training procedure, where latent information from the encoder is passed to the decoder twice. One pass resembles an AE-like behavior without noise, while the other pass resembles a VAE-like behavior with added noise to the latent vector before decoding. The parameter \( \lambda \) controls the shape of the latent vector distribution and the relative weights between the MMD term and the cross-entropy loss. The parameter \( \delta \) controls the relative weights between the AE and VAE objectives.

### 3.3 KAE m-MMD Loss

The MMD loss, between distributions \( \bar{x} \) and \( \bar{y} \) having \( N_x \) and \( N_y \) samples, is defined as their squared distance calculated in a space \( \mathcal{F} \) through the transformer \( \phi \):

\[
\text{MMD}(\bar{x}, \bar{y}) = \| \mu_x - \mu_y \|_F^2,
\]

\[
= \mu_x^T \cdot \mu_x + \mu_y^T \cdot \mu_y - 2 \mu_x^T \cdot \mu_y,
\]

where \( \mu_x = \frac{1}{N_x} \sum_{i}^{N_x} \phi(x_i) \). Introducing the kernel

\[
\mathcal{K}(\bar{x}_i, \bar{y}_j) = \phi(x_i)^T \cdot \phi(y_j),
\]

we can write the inner products, as follows:

\[
\mu_x^T \cdot \mu_y = \frac{1}{N_x N_y} \sum_{i}^{N_x} \sum_{j}^{N_y} \mathcal{K}(x_i, y_j),
\]

so

\[
\text{MMD}(\bar{x}, \bar{y}) = \frac{1}{N_y^2} \sum_{i}^{N_x} \sum_{j}^{N_y} \mathcal{K}(y_i, y_j)
\]

\[
+ \frac{1}{N_x} \sum_{i}^{N_x} \sum_{j}^{N_y} \mathcal{K}(\bar{x}_i, \bar{x}_j)
\]

\[
- \frac{2}{N_x N_y} \sum_{i}^{N_x} \sum_{j}^{N_y} \mathcal{K}(x_i', y_j'),
\]

where all \( \bar{y} \) are sampled from the target Gaussian distribution, and the kernel is defined as follows:

\[
\mathcal{K}(\bar{a}, \bar{b}) = \exp\left( -\frac{1}{D} \sum_{d=0}^{D-1} (\alpha_d - \beta_d)^2 \right),
\]

where \( D = 10 \times E \) is the size of the latent dimension and \( \sigma = \sqrt{0.32} \) has been empirically chosen (see S.I. for model performance with different sigma values).

The first term in the r.h.s. of Eq. (8) corresponds to \( \mu_y^T \cdot \mu_y \). It is typically dropped in the loss evaluations since this term does not contribute to the gradients of the loss with respect to the weights during backpropagation. So, the standard-MMD (s-MMD) loss is defined, as follows:

\[
s-MMD(\lambda) = \lambda \left[ \frac{1}{N_x^2} \sum_{i}^{N_x} \sum_{j}^{N_x} \mathcal{K}(\bar{x}_i, \bar{x}_j)
\]

\[
- \frac{2}{N_x N_y} \sum_{i}^{N_x} \sum_{j}^{N_y} \mathcal{K}(x_i', y_j') \right].
\]

For a zero-minimum inner product, the minimum of the first term is achieved \( \mu_x \) equal zero. So, minimizing the first term promotes all \( \phi(x_i) \) to spread out in the space \( \mathcal{F} \) while the second term brings \( \phi(\bar{x}) \) to be similar to the distribution of \( \phi(\bar{y}) \).

Based on the s-MMD loss, introduced by Eq. (10), we define the m-MMD loss, as follows:

\[
m-MMD(\lambda) = \lambda \left[ 1 - \frac{1}{N_x N_y} \sum_{i}^{N_x} \sum_{j}^{N_y} \mathcal{K}(\bar{x}_i, \bar{y}_j) \right].
\]
3.4 Decoding Methods

The KAE generation of a molecule involves sampling a vector, \( \vec{u} \in \mathbb{R}^{10 \times E} \) from a \( D \)-dimensional Gaussian distribution and decoding it. For conditional generation (CKAE), the sampled vector is concatenated with a condition \( C \), following its multiplication by its corresponding embedding vector. The resulting vector is subsequently mangled by a fully connected linear layer, yielding \( \vec{L} \) in the same dimension \( \mathbb{R}^{10 \times E} \). The decoder then translates the SMILES string sequence, character by character, with decoder-encoder attention applied to \( \vec{L} \).

During decoding, the token “<SOS>” is initially supplied. The decoder subsequently generates a probability distribution across \( T \) possible tokens for each input. The common practice is to continue the predictions using the token possessing the maximum probability, incorporating the token into the next-round input sequence and reiterating the procedure to obtain the next most probable token. This process is repeated until the “<EOS>” token is produced or the sequence length limit is achieved.

Contrary to retaining only the token of highest probability, KAE employed beam search, guided by the hyper-parameter beam size, to derive a broader array of interpretations of the same vector, \( \vec{L} \). With a beam size, \( B \), where \( B \leq T \), a maximum of \( B \) outputs are produced from a single decoding procedure.

The beam search algorithm logs the probability of each step for each of the \( B \) sequences. For the first step, the top \( B \) most probable tokens are selected. In subsequent steps, the model decodes from \( B \) input sequences concurrently. Given that each of the \( B \) sequences has \( T \) potential outcomes for the succeeding token, the total number of potential next-step sequences equates to \( B \times T \). These sequences are then ranked according to the sum of their probabilities for all \( S \) characters.

In a beam search, the probability of a sequence of tokens indexed from \( s, s - 1, s - 2 \ldots \) to 0 can be represented, as follows:

\[
P(s, s - 1, s - 2, \ldots, 0) = P(s|s - 1, s - 2, \ldots, 0) \times P(s - 1, s - 2, \ldots, 0)
\]

This can be interpreted as the product of individual probabilities,

\[
P(s, s - 1, s - 2, \ldots, 0) = P(s|s - 1, s - 2, \ldots, 0) \
\times P(s - 1|s - 2, s - 3, \ldots, 0) \times \ldots P(0)
\]

However, calculations of long sequences based on this equation yield impractically small numbers as every term is smaller than one. Therefore, we sum the log probabilities instead.

For the \( B \times T \) sequences with equal sequence length \( S \), the probability of the \( i \)th sequence at each position \( s \) is denoted as \( P_{i,s} \). Excluding the probabilities of padding tokens, the sum of log probabilities, \( P_i \) for the \( i \)th sequence is computed as:

\[
P_i = \frac{1}{\sqrt{N_i}} \sum_{s \neq pad}^{S} \log(P_{i,s})
\]

Here, \( N_i \) represents the quantity of non-padding tokens in sequence \( i \).

To foster diversity in decoding, sequence lengths are factored into the computation of \( P_i \). The \( \frac{1}{\sqrt{N}} \) term counteracts the preference for shorter sequences over longer ones, as longer sequences typically yield smaller sums of log probabilities.

The top \( B \) most probable tokens are selected and serve as the inputs for the subsequent iteration, which continues until the maximum sequence length \( M \) is attained or all top \( B \) candidates have produced the “<EOS>”, signaling the cessation of decoding.

3.5 Similarity Exhaustion Search Procedures

The hyperparameters of SES include the beam size \( (B) \), the interval \( (\delta_i) \), the maximum increase in condition \( (\Delta) \), and the number of repetitions in
Phase-two ($R$). In our implementation, the parameters were set as $B = 15$, $\delta_s = 0.1$, $\Delta = 20$, and $R = 4$.

**Condition Search:** The Condition Search, the initial stage of SES, begins by assigning each molecule to be optimized, denoted as $m_i$, with its corresponding PLogP value as the initial condition $c_i$. The index $i$ represents the molecule’s position. The latent vector $z_i$ is obtained through the encoding process.

During each step $s_j$, where $j$ starts from zero, a search is conducted for the vector $z_i$ with an adjusted condition $c_i + j\delta_s$. The concatenated vector of $z_i$ and the updated condition vector are then passed to the decoder. By utilizing beam search, a set of $B$ results is generated at each step. This process continues until the increment $j\delta_s$ reaches the maximum allowed value, $\Delta$. In total, $B + B\Delta$ candidates are produced for the molecule $m_i$ through this procedure. Subsequently, all candidates are filtered, retaining only those that exhibit a Tanimoto similarity within the range of 0.4 compared to the original molecule. The PLogP values of the remaining candidates are calculated and ranked. The optimization process is deemed successful if the highest PLogP value among the candidates for the $i^{th}$ molecule surpasses its original value. In such cases, the corresponding PLogP value and the SMILES representation of the candidate are recorded.

The purpose of condition search is to look for a set of candidates with similar encoder-estimated $z_i$ but with higher PLogP conditions. However, this procedure does not guarantee good samplings around all candidates. This means the decoded molecules would be dissimilar or even out of the similarity constraints from the encoded targets. In addition, despite the correct reconstruction, because these molecules represent the tail of the distribution of the PLogP conditions in the training data, they could have poor latent space definitions around them. This can cause a similar problem to reconstruction where better candidates within the constraint cannot be found due to a decrease in factors such as validity, uniqueness, and novelty. Therefore, a repositioning step is developed to ensure all molecules, especially for those $z_i$ that cannot be reconstructed correctly, can explore possibly better-starting points in the later search.

**Repositioning:** Repositioning is used to encourage sampling from regions farther away from the encoded latent vectors. To achieve this, sampling around the vector $z_i$ at the corresponding condition $c_i$ is performed. The sampling process involves adding a noise term $\epsilon$ drawn from the same Gaussian distribution employed during training.

If the sampled vector $\tilde{z}_i$ yields a superior outcome compared to the previous search, it is recorded. Whenever a recorded $\tilde{z}_i$ exists, the subsequent sampling iteration starts from this repositioned vector. This repositioning step aims to expand exploration towards molecules that exhibit a greater separation from $z_i$, especially for vectors that display limited or no improvement during the condition search. This repositioning procedure is iterated 100 times to enhance the exploration process. A visual representation of this procedure can be seen in Figure 5.

![Fig. 5: Repositioning. A $\tilde{z}_i$ is selected around $z_i$ if the generated molecule falls within the similarity threshold ($\sigma$) and exhibits an improvement in the optimized property. The subsequent search repetition is then conducted around $\tilde{z}_i$. Through repositioning, the search space expands for molecules that showed little improvements during the condition search.](image)

**Phase Two:** The preceding stages of the SES, namely the condition search and repositioning, yield two distinct sets of latent vectors. The first set comprises the original encoded $z$ vectors, while
the second set consists of the repositioned \( \tilde{z} \) vectors. In the second phase, the search process is performed in parallel using a combination of the condition search and repositioning approaches.

For each set, noise is added in a similar manner as during the repositioning stage. However, in this phase, every \( c_i \) is adjusted by \( c_i + j\delta_\sigma \), following the same procedure as the condition search.

By applying the filtering and selection criteria identical to those used in the condition search, new molecules with the highest PLogP values are recorded for both sets. The phase two process is repeated \( R \) times. After completing the \( R \) repetitions, for each molecule candidate, the superior result between the two sets is chosen.

3.6 Docking Methods

The generated molecular structures were evaluated using Autodock Vina [29], following a procedure that ensures meaningful comparisons to other molecular generation models, such as GFlowNet [5]. All results were independently tested by using Glide docking from Schrodinger Inc. [30, 31] to ensure the results are robust across different docking software packages.

Autodock Vina is known for its efficiency and speed, making it suitable for high-throughput screening. It employs an empirical scoring function for accurate prediction of binding affinities. On the other hand, Glide utilizes a force field-based scoring function that is widely recognized for its accuracy. In particular, Glide excels at predicting binding poses with high precision and has undergone extensive validation. Its efficacy in handling large and flexible ligands has established it as the gold standard in the field. To ensure meaningful comparisons to GFlowNet [5], we followed the same procedure implemented for Autodock Vina calculations. Specifically, 20 conformers were used per ligand, exhaustiveness was set to 32, and a maximum of 10 binding modes were generated.

3.6.1 Glide calculations

The model protein receptor (PDB ID: 4jnc) was prepared by using the adept Protein Preparation Wizard tool in Maestro [46]. The protonation states were defined at a neutral pH = 7.0. The protein was subsequently refined via energy minimization using the OPLS4 force field [47].

The 3D grid representation of the receptor binding site was prepared by using the Maestro Grid Generation tool, ensuring that the grid size and positioning was perfectly aligned with those used in GFlowNet calculations. All model structures for docking were prepared using the LigPrep tool of Maestro. Utilizing the Pre-Dock tool in Maestro, the docked molecules were prepared and assigned charges and protonation states via the OPLS4 force field [47]. The XP (extra precision) [30, 31] flexible docking protocol was then implemented, employing a range of settings designed to optimize the docking accuracy and precision. These included a selection of all predefined functional groups for biased torsional sampling, the addition of Epik state penalties [48] to the docking scores [46], and the enhanced planarity setting for conjugated pi groups.

In the initial step of the docking procedure, 10,000 poses were filtered through the Glide screens, and the top 1,000 poses were selected for energy minimization. The expanded sampling option was utilized to maximize pose flexibility during the search. Ultimately, a single pose was retained for each ligand. The final stage involved refining the best docking poses. Two consecutive refinement steps were performed, each consisting of a post-docking energy minimization on the selected pose, eliminating the need for additional sampling. As a result, highly optimized and reliable docking poses were obtained and compared against those obtained with Autodock Vina [29] calculations.

Acknowledgments

This work was supported by the NIH Grant GM106121 (VSB). We acknowledge NERSC for the generous computational resources and technical help that make this work possible.

References


S1 Training Datasets

The KAE model has been trained on 90% (225,000) of the entries of the ZINC250K dataset [49]. Within the other split, 1,000 molecules were used for validation and 24,000 were used for testing. In CKAE, the training data included the molecular properties from the ZINC250K library. For the dataset with 300,000 docking candidates from [5], all entries were used for training.

S2 Data Preparation

We used the ZINC250K dataset consistent with [9]. During dataset preparation, all SMILES strings were added to the start of sequence tokens “<SOS>” and the end of sequence tokens “<EOS>”. The two tokens are used in the testing phase of the model to determine if the translation is completed. There were 41 unique characters from the database. They were extracted and put into a character-to-token dictionary that allows conversions from characters to tokens. The padding was added at the end as the 42nd token, making the dictionary size T. A token-to-character dictionary was created at the same time for the interpretation of the model output in tokens. With the character-to-token dictionary, all SMILES representations were converted to the corresponding tokens. Since we use the Transformer architecture, model inputs were made into the same shape for batch training by adding paddings to all sequences. After padding, all sequences have the same length. The numerical values of the penalized octanol-water partition coefficient (PLogP) were concatenated to the end of the corresponding tokenized molecules. This adds one extra dimension in the sequence length. The maximum sequence length for each molecule in the dataset is denoted as M. The tokenized dataset is then partitioned into 256-size batches.

S3 Learning Behavior

We have analyzed the KAE learning rate by comparing its performance to the performance of models with the same architecture but various different loss functions (Figure S1). The reconstruction performance was evaluated from 1000 randomly selected molecules from the validation set over increasing numbers of epochs. Figure S1 shows the evolution of validity, uniqueness, novelty and reconstruction along the training process for models based on a loss that combines the weighted cross-entropy \( L_{WCEL}(\lambda, \delta) \), defined by Eq. (4), with modified-MMD (m-MMD(\lambda), Eq. 11), standard-MMD (s-MMD(\lambda), Eq. 10), or KL divergence. All models were trained with the ZINC250K dataset for 200 epochs, with \( \lambda = 1 \) and \( \delta = -1 \). When \( \lambda = -\delta \), the weighted cross-entropy loss \( L_{WCEL}, \) Eq. 4) reduces to the standard cross-entropy loss \( L_{CEL} \). Additionally, we examine the effect of adding noise to the latent space while training with the m-MMD loss. The results (Figure S1) indicate that the KAE model based on the m-MMD loss and Gaussian noise added to the latent space exhibits the best performance. Figure S1a shows the percentage of valid strings regenerated as a function of the number of epochs. Figure S1b shows the percentage of input molecules that were correctly regenerated by the models. The models exhibit significant differences in their ability to generate valid SMILES strings and reconstruct input molecules. The KAE model trained with noise in latent space generated the highest percentage of valid SMILES strings, making it preferable to other models. For example, the model trained with KL divergence exhibited much lower validity and significantly slower learning rate. The assessment of novelty and uniqueness also shows that s-MMD and m-MMD models trained with Gaussian noise added in latent space (noisy models) performed much better than the corresponding models without noise. In summary, the models fully trained with m-MMD loss outperformed those trained with the s-MMD loss in terms of the NUV (Novelty, Uniqueness, and Validity) values after 200 epochs.
Fig. S1: Comparison of learning rates for models trained with m-MMD loss, s-MMD loss, and KL divergence loss. (a) Validity evaluated at each epoch. (b) Fraction of molecules properly reconstructed as a function of epochs. (c) Novelty evaluated at each epoch. (d) The uniqueness at each epoch. The model labeled as KL includes an extra layer that estimates the standard deviation of each latent vector. The models labeled with m-MMD are trained with the loss $L_{CEL} + m\text{-MMD}(\lambda = 1)$, s-MMD with $L_{CEL} + s\text{-MMD}(\lambda = 1)$, and KL with $L_{VAE}$. "No noise" indicates that noise has not been added to the latent vectors during training. Validity, uniqueness, and novelty are calculated at the end of each epoch from 1000 randomly sampled latent vectors. The reconstruction rate is calculated using 1000 molecules from the validation set.

S4 Impact of the $\lambda$ Parameter

The reason for increasing $\lambda$ is similar to that of increasing $\beta$ in the case for $\beta$-VAE [50]. Both $\lambda$ in KAE and $\beta$ in $\beta$-VAE encourage the model to learn more efficient latent representations and to construct smoother latent space. However, since KAE has different architecture and loss objectives from VAE, the
aforementioned regularisation does not lead to the same result in terms of the NUVR metric when both $\lambda$ and $\beta$ are set to one for KAE and $\beta$-VAE.

For the best model using m-MMD in Figure S1, all validities are lower than 90%. This is improved by increasing the $\lambda$ value for the m-MMD term as shown in Table S1. The models in Table S1 were first trained with $\lambda = 1$ for 85 epochs then with higher values for an additional 1 epoch. $\delta$ values were set to $-\lambda$ throughout the training process to exclude any effects from WCEL in the comparison.

Increasing the $\lambda$ value tightens the placement of latent vectors together according to the m-MMD loss. This is reflected in the increase in the probability of sampling valid molecules when the latent vectors are drawn from the same distribution. However, as the latent vectors become closer, it becomes more challenging for the decoder to differentiate them, resulting in a decrease in reconstruction. The decrease in uniqueness and novelty with increased $\lambda$ values can be attributed to the decoder more frequently identifying different molecules with overlapping latent representations as the same ones.

The overall effects of $\lambda$ are shown by the product of N, U, and V (NUV) as well as the one including reconstruction (NUVR).

Table S1 shows the trend of NUV and NUVR as $\lambda$ is adjusted. It is observed that validity peaks with larger $\lambda$ and the model trained with $\lambda = 24.5$ has the highest NUV. However, the reconstruction rate decreases significantly with increasing $\lambda$ values.

### Table S1: Model performance with varying $\lambda$. The result shows sampling 1k latent vectors by continued training of the model from the same checkpoint (85 epochs) with the loss function being $\mathcal{L}(\lambda = 1, \delta = -1)$, but then followed by an additional epoch with different $\lambda$ values (loss functions are then $\mathcal{L}(\lambda = \lambda, \delta = -\lambda)$).

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Validity</th>
<th>Novelty</th>
<th>Uniqueness</th>
<th>NUV</th>
<th>Reconstruction</th>
<th>NUVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.782</td>
<td>1.000</td>
<td>0.995</td>
<td>0.778</td>
<td>0.988</td>
<td>0.769</td>
</tr>
<tr>
<td>2.0</td>
<td>0.802</td>
<td>1.000</td>
<td>1.000</td>
<td>0.802</td>
<td>0.978</td>
<td>0.784</td>
</tr>
<tr>
<td>5.0</td>
<td>0.849</td>
<td>1.000</td>
<td>1.000</td>
<td>0.849</td>
<td>0.933</td>
<td>0.792</td>
</tr>
<tr>
<td>10.0</td>
<td>0.847</td>
<td>0.999</td>
<td>0.999</td>
<td>0.845</td>
<td>0.792</td>
<td>0.669</td>
</tr>
<tr>
<td>15.0</td>
<td>0.913</td>
<td>0.998</td>
<td>1.000</td>
<td>0.911</td>
<td>0.527</td>
<td>0.480</td>
</tr>
<tr>
<td>20.0</td>
<td>0.929</td>
<td>1.000</td>
<td>1.000</td>
<td>0.929</td>
<td>0.246</td>
<td>0.229</td>
</tr>
<tr>
<td>24.5</td>
<td>0.961</td>
<td>0.999</td>
<td>0.998</td>
<td>0.958</td>
<td>0.060</td>
<td>0.057</td>
</tr>
<tr>
<td>25.0</td>
<td>0.940</td>
<td>0.998</td>
<td>1.000</td>
<td>0.938</td>
<td>0.043</td>
<td>0.040</td>
</tr>
<tr>
<td>25.5</td>
<td>0.943</td>
<td>1.000</td>
<td>1.000</td>
<td>0.943</td>
<td>0.039</td>
<td>0.037</td>
</tr>
<tr>
<td>26.0</td>
<td>0.965</td>
<td>0.998</td>
<td>0.999</td>
<td>0.962</td>
<td>0.029</td>
<td>0.028</td>
</tr>
<tr>
<td>27.5</td>
<td>0.962</td>
<td>0.996</td>
<td>0.999</td>
<td>0.957</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>30.0</td>
<td>0.970</td>
<td>1.000</td>
<td>0.987</td>
<td>0.957</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

We seek to find a solution that can increase validity while maintaining other metrics at the same level. Further controlling the model via WCEL was the key to this problem. We compared model performance with a range of $\delta$ values in S.I. and choose $\delta = 1$ in WCEL; we compared different $\lambda$ values with $\delta$ fixed to 1. In Figure S2, models are trained with the loss function $\mathcal{L}(\lambda = 1, \delta = 1)$ throughout the training process for 200 epochs. After each epoch, we evaluated the NUV and measured the reconstruction rates. It can be observed in Figure S2a that higher $\lambda$ values lead to better final validity. However, uniqueness in Figure S2b breaks down for the case of when $\lambda = 4$ while novelty and reconstruction rates converge to around 100% in (Figure S2c and Figure S2d). Therefore, the $\lambda = 3.5$ model is trained for additional 200 epochs (total 400 epochs) to give final performance metrics in Table 1.
Fig. S2: The performance comparison of models trained with different $\lambda$ values, while keeping $\delta = 1$, using the m-MMD loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. The evaluation metrics include validity, uniqueness, and novelty, which are computed at the end of each epoch based on 1000 randomly generated molecules from each model. Additionally, the reconstruction rate is calculated using 1000 molecules from the validation set. In the legend, the notation $L_{xD_y}$ represents a model trained with $\lambda = x$ and $\delta = y$. For instance, the model labeled L3D1 corresponds to $L(\lambda = 3, \delta = 1)$.

S5 Latent Space And Model Performance

In m-MMD, with the RBF-kernel function, removing the $\mu_x^T \mu_x$ term is believed to be helpful since as it allows the distributions of individual molecules to be closer together. This makes the sampling region have fewer places where the decoder cannot infer valid molecules. A demonstration and a comparison with the latent spaces of s-MMD and m-MMD is presented in Figure S3.

It can be seen from Figure S1b that, with or without noise, the models trained with s-MMD have a faster-converging reconstruction rate than the models trained with m-MMD. This is because the extra $\mathcal{K}(\bar{x}, \bar{x})$ term in s-MMD promotes the separation of the latent representations of the data points such that the decoder can easily differentiate the representations. However, since the latent vectors that represent valid molecules are far from each other, the validity is significantly lower for the models trained with s-MMD.

We consider increasing $\lambda$ as an approach to optimize the model performance in N, U, and V, reducing the regions with holes while still making individual molecules distinct from each other.
Fig. S3: Latent vectors obtained by passing 10k ZINC250K molecules to the encoder and transforming under the same principal components extracted from the standard Gaussian distribution. (a) m-MMD results showing all latent vectors well-incorporated in the Gaussian. (b) s-MMD loss makes the latent vectors more scattered relative to the Gaussian, making it less likely to obtain valid output by sampling from the Gaussian. (c) and (d) show the actual vectors passed into the decoder in the training process with latent noise added.

S6 Enhancing Generation Performance through Beam Search

To further improve the performance of our model, we employ beam search, a popular decoding technique in sequence generation tasks. Beam search involves selecting a single output from a set of $B$ potential candidates, based on specific criteria outlined in the decoding method (see Section 3.4).

Similar to previous studies [11, 18] that have employed checking methods to enhance model performance, we propose a novel approach of employing beam search as a post-generation evaluation step. During the molecule generation process, we consider one of the outputs obtained from the beam search results. For
Fig. S4: Molecules obtained by sampling from the 0.1-SD Gaussian distribution centered around a specific latent vector, while varying the beam size. Figure (a) illustrates the molecules found using a beam size of one, where only the original encoded molecule is identified despite the added noise. In contrast, Figure (b) showcases the molecules discovered when a beam size of two is employed, revealing seven distinct molecules out of the ten samples.

Table S2: Model performance with varying beam sizes. This table shows the model’s generation performance with various beam sizes by sampling 10k latent vectors each time. One output is selected out of the interpretations given by all beam search results for each latent vector. The result with a beam size of one is an equivalent measurement to other methods that do not use beam search and grammar checks.

<table>
<thead>
<tr>
<th>Beam Size</th>
<th>Novelty</th>
<th>Uniqueness</th>
<th>Validity</th>
<th>NUV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.996</td>
<td>0.974</td>
<td>0.998</td>
<td>0.968</td>
</tr>
<tr>
<td>2</td>
<td>1.000</td>
<td>0.996</td>
<td>1.000</td>
<td>0.996</td>
</tr>
<tr>
<td>3</td>
<td>1.000</td>
<td>0.998</td>
<td>1.000</td>
<td>0.998</td>
</tr>
<tr>
<td>4</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

instance, with a beam size of two, two possible interpretations are generated for the same latent vector. We iterate through the B results, starting with the top-ranked interpretation, and if any of the generated SMILES strings are novel, unique, and valid, we stop the evaluation process and retain the corresponding SMILES string. Priority is given to retaining valid molecules over those that are unique and novel. By checking and selecting from all the beam-searched outputs, we increase the likelihood of finding SMILES strings that meet the criteria of novelty, uniqueness, and validity. If all the outputs fail to meet the validity criterion, we return the top-ranked result based on summed log probabilities for that decoding step.

With larger beam sizes, it is expected that the output of the top one probable SMILES will have higher validity but lower uniqueness as molecules with specific character combinations appear more often in the dataset. However, this is compensated by more possibilities from the search results. The effect of beam search results in better sampling efficiency but is not due to the sheer increase in the number of candidates. We believe the beam search can help differentiating two latent vectors that are similar by providing alternative interpretations per vector.

In the case where the beam size is equal to one, the method is identical to greedy search which takes only the next-step candidate with maximum probability; We compare the results done at different beam sizes. 10k vectors are sampled for each listed beam size. The result of the beam size of one is used as the control group.

Table S2 presents the model’s generation performance across different beam sizes, as measured by various metrics. The metrics assessed include novelty, uniqueness, validity, and the combined metric NUV. The results clearly indicate that as the beam size increases, the model’s performance improves consistently.
Fig. S5: Performance comparison of the models trained with different sigma values using modified MMD loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. Note that $2\sigma^2$ (2 sigma squared) in the legend represents the value used for $2\sigma^2$ in Eq. 9 and $E$ is the embedding dimension. Validity, uniqueness, and novelty are calculated at the end of each epochs using 1000 randomly generated molecules from each of the models. And the reconstruction rate is calculated using 1000 molecules from the validation set.

Notably, when the beam size exceeds three, the performance reaches a plateau, achieving the highest possible value of 1.0 for the NUV metric.

To further highlight the capabilities of beam search, we conduct additional experiments where we sample from a small distribution around a specific latent vector. We start by selecting a molecule from the training set and encoding it into its corresponding latent vector. Next, we generate 10 noise vectors by sampling from a Gaussian distribution with a standard deviation one-tenth of that used during training (0.1-SD). These noisy latent vectors are then decoded using the beam search approach. The results obtained from beam search demonstrate the ability to find diverse candidates that are similar to the molecule being sampled. Notably, when a beam size of two is employed, six additional candidates are discovered compared to the case where beam search is not utilized (i.e., beam size of one).
In Figure S5 and Figure S6, we compare the model performance of different sigma values of the kernel (Eq. 9). It can be observed that the final uniqueness, novelty, and reconstruction rate are similar, while there are clear differences in validity performance. Therefore, the sigma value that gives the highest final validity rate is considered optimal. It can be observed that lower $2\sigma^2$ values give higher validity rates and $2\sigma^2 = 0.0005 \times E$ is the optimal value for both m-MMD and s-MMD models. At the optimal sigma value, the m-MMD model has higher validity rate than the s-MMD model. Besides, if models are trained with even lower sigma values ($2\sigma^2 = 0.00005 \times E$ for example), the models would break down because they cannot get gradient information from the MMD loss term (results not shown).
Fig. S7: Performance comparison of the models trained with different \( \delta \) values (with \( \lambda = 1 \)) using modified MMD loss and KL loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. Validity, uniqueness, and novelty are calculated at the end of each epoch using 1000 randomly generated molecules from each of the models. And the reconstruction rate is calculated using 1000 molecules from the validation set. Note that LxDy in the legend means that the model is trained with \( \text{lambda} = x \) and \( \text{delta} = y \). For example, the model labeled with L1D-1 is trained with \( \mathcal{L}(\lambda = 1, \delta = -1) \).

S8 \( \delta \) Comparison

We designed the \( \delta \) such that when \( \lambda \) is 1, and \( \delta \) is greater than -1, the AE-like term is contributing to how the model reconstructs the inputs. When \( \delta \) is large, the model ignores the regions with added noise and thus is turned into a pure auto-encoder. When \( \delta \) is equal to -1, the model is VAE-like where each latent vector is treated like a region. When \( \delta \) is in between these two extrema, the model is able to achieve the AE-like reconstruction rate while obtaining better generative performance in NUV metrics. Finally, since the addition of \( \delta \) is the key to the WCEL, we again compare the KL loss combined with WCEL with \( \delta \) of 1, as opposed to the original CEL in VAE. We show that our formalism of the WCEL is capable of bringing significant improvements to KL-based models as well in Figure S7.

In Table S1, results with \( \lambda \) parameters greater than 30 are not listed. The model shows a significant reduction in novelty and uniqueness when \( \lambda \) goes above 30. This is because the regions of individual molecules encoded in the latent space are confined to be more Gaussian-like. This reduces the model’s ability to differentiate one from another. On the contrary, when the molecules’ latent vectors are more
separated, leaving “holes” in between, both uniqueness and novelty tend to be high but validity decreases as a result.