

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 employs two innovative loss functions: modified maximum mean discrepancy (m-MMD) and weighted reconstruction (\mathcal{L}_{WCEL}). The m-MMD loss has significantly improved the generative performance of KAE when compared to using the traditional KL loss of VAE, or standard MMD. Including the weighted reconstruction loss \mathcal{L}_{WCEL} , KAE achieves valid generation and accurate reconstruction at the same time, allowing for generative behavior that is intermediate between VAE and AE not available in existing generative approaches. Further advancements in KAE include its integration with conditional generation, setting a new state-of-the-art benchmark in constrained optimizations. Moreover, KAE has demonstrated its capability to generate molecules with favorable binding affinities in docking applications, as evidenced by AutoDock Vina and Glide scores, outperforming all existing candidates from the training dataset. Beyond molecular design, KAE holds promise to solve problems by generation across a broad spectrum 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].

Autoencoders (AEs) encode the input data by compression into a low-dimensional space [12]. Though providing a high lower bound for accurate reconstruction, such space is not well-structured and in some regions, the decoder does not generate output that resembles the training data,

thereby limiting its generative capabilities. Sacrificing reconstruction performance, VAEs mitigate this disadvantage by enforcing encoded latent vectors to known prior distributions. Upon decoding samples from those distributions, VAEs generate outputs mimicking the training data. An outstanding challenge of great interest to drug discovery is to harness the power of VAEs to generate molecular candidates with optimal properties during the screening phase of molecular development, while preserving AE’s high reconstruction rate for precise lead candidate optimizations.

Generative models 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, the so-called NUV to R tradeoff, as a model with high

reconstruction ability usually does not achieve high metrics for novelty, uniqueness, and validity.

Optimizing the design of molecules near a reference molecule 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 [13–16].

Kernel-Elastic Autoencoder (KAE) stands out as a new category for a self-supervised generative model based on a modified maximum mean discrepancy and weighted reconstruction loss functions. Leveraged by the Transformer architecture [17–20], KAE (Figure 1) effectively overcomes the NUV-R tradeoff by combining the merits of both autoencoder (AE) and variational autoencoder (VAE) models. KAE’s loss function is a modified version of the Maximum Mean Discrepancy (MMD), inspired by [21–23], that shapes the latent space and enables better performance than using Kullback-Leibler (KL) divergence loss used in VAEs. When coupled to the weighted cross-entropy loss (\mathcal{L}_{WCEL}), KAE, without any checking for molecular grammar or chemical rules, outperforms both string and graphical-based models in generation tasks while exhibiting nearly flawless reconstruction, as demonstrated on the ZINC250k testing sets. The freedom to adjust KAE’s behavior through the \mathcal{L}_{WCEL} gives the "Elastic" term in the naming.

When implemented to solve optimization problems, KAE outperforms the state-of-the-art by a substantial 28% [24]. Additionally, KAE tackles the problem of molecular docking by finding suitable binding ligands with conditional generation, as demonstrated using the dataset from GFlowNet [5]. Superior candidates from the baseline and the training data are independently verified by both Autodock Vina [25] and Glide [26, 27], demonstrating its efficacy and practicality.

2 Result

2.1 KAE Performance

The overall performance of the KAE (Figure 1) 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 (\mathcal{L}_{WCEL}), with hyperparameters λ and δ , 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 (R) 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 $\mathcal{L}_{WCEL}(\lambda, \delta)$ defined by Eq. (4) where the hyperparameter δ controls the AE-like objective (see S.I. for a discussion of the effect of changing λ and δ). The best results for the NUVR metric were obtained by using a combination of $\lambda = 3.5$ and $\delta = 1$.

With the same Transformer architecture, KAE is compared to approaches using different loss functions (SI, Figure 2) 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.

2.2 Learning Behavior

We have analyzed the KAE behavior by comparing under the same architecture but with various loss functions (Figure 2). The reconstruction was evaluated from 1000 molecules from the validation set at every epoch. Figure 2 shows the improvement in validity, uniqueness, novelty, and reconstruction along the training process for models based on a loss that combines the weighted cross-entropy

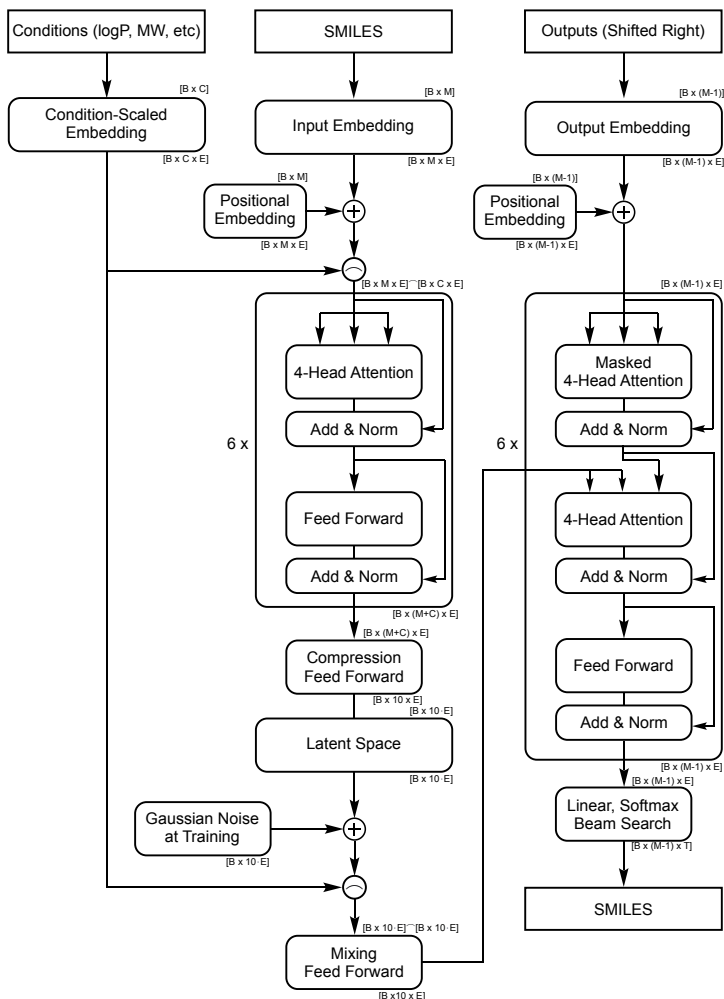


Fig. 1: Conditional KAE transformer architecture. KAE consists of 6 encoder layers, 6 decoder layers, and a latent space for conditional generations. During training, the condition is concatenated after positional embedding and provided as input to the 4-head attention encoder. The condition is also concatenated with the latent vector before a mixing layer. During training, Gaussian noise is added to the latent vectors. 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 .

$\mathcal{L}_{WCEL}(\lambda, \delta)$, defined by Eq. (4), with m-MMD (m-MMD(λ), Eq. 11), s-MMD, 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 (\mathcal{L}_{WCEL} , Eq. 4) reduces to the standard cross-entropy loss (\mathcal{L}_{CEL}). Additionally, we examine the effect of noise while training with the m-MMD loss. The results (Figure 2) indicate that the KAE model using m-MMD loss with Gaussian noise added to the latent space exhibits the best

performance. The models exhibit significant differences in their ability to generate valid SMILES strings and reconstruct input molecules. The m-MMD 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 a 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

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 \mathcal{L}_{WCEL} and is equivalent to not using any AE objectives. The validity check selects alternative candidates from beam search.

Method	N	U	V w/o	V	NUV	R	NUVR
CVAE [9] ^a	0.980	0.021	0.007	N/A	0.0001	0.446	5e-6
GVAE [10] ^a	1.000	1.000	0.072	N/A	0.072	0.537	0.039
JTVAE [11] ^a	1.000	1.000	0.935	1.000	0.935	0.767	0.717
MoFlow [28]	1.000	0.999	0.818	1.000	0.817	1.000 ^b	0.817
Rebalanced [29]	1.000	1.000	0.907	0.938	0.907	0.927	0.841
GraphDF [30]	1.000	0.992	0.890	1.000	0.883	1.000 ^b	0.883
ALL SMILES [31] ^a	1.000	1.000	N/A	0.985	N/A	0.874	N/A
β -VAE [24]	0.998	0.983	0.983	0.988	0.964	N/A	N/A
KAE ($\lambda = 1, \delta = -1$)	0.998	0.994	0.863	N/A	0.856	0.992	0.849
KAE ($\lambda = 3.5, \delta = 1$)	0.996	0.973	0.997	1.000	0.966	0.997	0.963

^aResults obtained from sampling 1,000 vectors from latent space.

^bReconstruction rates were obtained on training datasets.

space (noisy models) performed much better than the corresponding models without noise. The reason for adding noise is to prevent the model from remembering exactly the locations of the latent vectors and the decoder has to see the multitude of possible outcomes related to the region of the decoding latent vector. Further, the decision to add a Gaussian noise on top of confining the latent vector to the same Gaussian through m-MMD is to maximize the overlap of the distribution of all latent vectors with respect to the distribution of any individual latent vector. This approach is different from VAE as the VAE has the option to output small variances for some latent vectors which could reduce the probability of sampling corresponding instances from its prior distribution.

2.3 Conditional KAE

In this section, the performance of the Conditional-KAE (CKAE) (Figure 1) on the constraint optimization task is investigated. CKAE generates new candidates conditioned on properties such as PLogP or docking scores. Here, we demonstrate the capabilities of CKAE as applied to the PLogP

values defined, as follows: [9, 11]:

$$PLogP(m) = LogP(m) - SA(m) - ring(m), \quad (1)$$

where $LogP$ 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 $ring(m)$ 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 coefficient 0.9997). 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. We have also trained a separate model using the dataset from Lim et al [39] who developed a Conditional-VAE (CVAE) with Recurrent Neural Network (RNN) architectures to

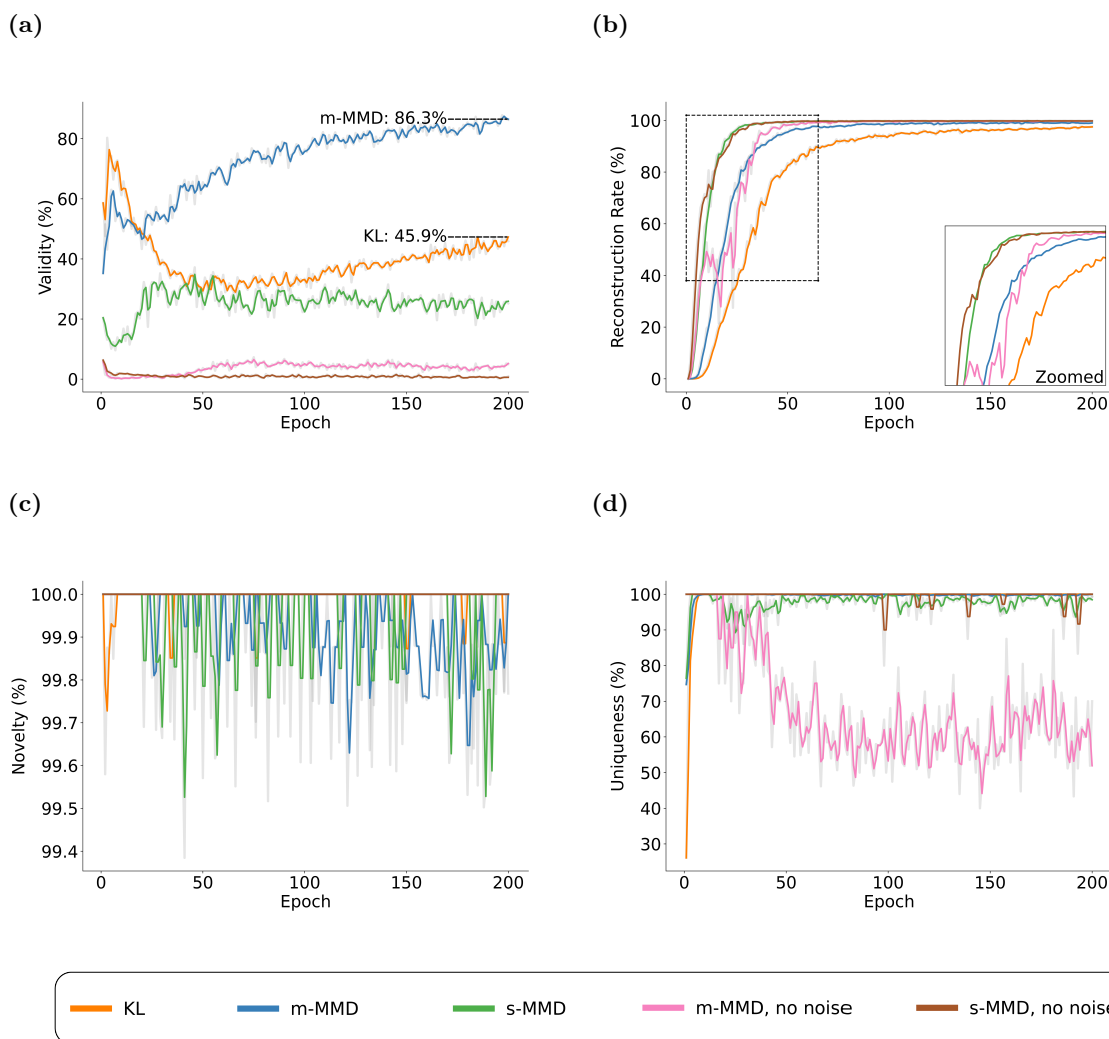


Fig. 2: Comparison of learning rates for models trained with m-MMD loss, s-MMD loss, and KL divergence loss. (a) Validity evaluated at each epoch. (b) Fraction of molecules properly reconstructed as a function of epochs. (c) Novelty evaluated at each epoch. (d) The uniqueness at each epoch. The model 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 $\mathcal{L}_{CEL} + m\text{-MMD}(\lambda = 1)$, s-MMD with $\mathcal{L}_{CEL} + s\text{-MMD}(\lambda = 1)$, and KL with $\mathcal{L}_{VAE} = \mathcal{L}_{CEL} + KL(\lambda = 1)$. "No noise" means no noise is added to the latent vectors during training.

sample molecules given five distinct pharmaceutically relevant properties. The comparison between CKAE and CVAE from Lim et al further shows that CKAE generates candidates correlating to the asked conditions and outperforms the given baseline by a wide margin (Table 3).

Instead of using regressors to navigate in the latent space[11, 24, 29, 40], 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

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 (standard deviations reported after \pm). 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.

Method ^a	PLogP-Improvement	Tanimoto Similarity	Success Rate
JT-VAE [11]	0.84 \pm 1.45	0.51 \pm 0.1	83.6%
MHG-VAE [33]	1.00 \pm 1.87	0.52 \pm 0.11	43.5%
GCPN [34]	2.49 \pm 1.30	0.47 \pm 0.08	100%
Mol-CycleGAN [1]	2.89 \pm 2.08	0.52 \pm 0.10	58.75%
MolDQboot [35]	3.37 \pm 1.62	N/A	100%
ZINC250K (This work)	4.64 \pm 2.33	0.48 \pm 0.16	97.88%
MoFlow [28]	4.71 \pm 4.55	0.61 \pm 0.18	85.75%
Random Sample (This work)	4.78 \pm 2.08	0.43 \pm 0.03	81.75%
MNCE-RL [36]	5.29 \pm 1.58	0.45 \pm 0.05	100%
β -VAE [24]	5.67 \pm 2.05	0.42 \pm 0.05	98.25%
CKAE (This work)	7.67 \pm 1.61	0.42 \pm 0.02	100%

^aTanimoto similarity constraint = 0.4

Table 3: Performance of CKAE compared to CVAE, as applied to conditional molecular generation We impose strict counting criteria for the CKAE statistics so that only valid, novel, and unique molecules are considered attempts. Therefore, the number of valid molecules will be equal to the number of attempts. To further compare to the CVAE method by Lim et al where there is more than one valid molecule per attempt, we have applied beam search with a beam size of 10 (labeled CKAE w. Beam). When beam search is used, the number of valid molecules reports the number of valid, novel, and unique candidates derived from all attempts. The success rate is defined as 100 times the rate of finding a candidate within a 10% error range of each property per attempt. The result shows CKAE, without using a method to derive more candidates per output SMILES strings is better than CVAE with RNN architectures. Further, if beam search is applied, CKAE significantly outperforms the given baseline.

Method	Target	Attempts	Number of Valid Molecules	Success Rate (%)
CVAE Lim et al	Aspirin	28,840	32,567	0.34
CVAE Lim et al	Tamiflu	15,960	34,696	0.62
CKAE	Aspirin	4743	4743	2.11
CKAE	Tamiflu	3715	3715	2.63
CKAE w. Beam	Aspirin	671	4221	14.90
CKAE w. Beam	Tamiflu	436	3927	22.94

(e.g., PLogP) by using the same or slightly perturbed latent vector representations with gradually increasing conditions. Formally, $f(z, c) \approx f(z + \Delta_z, c + \Delta_c)$ for small values of Δ_z and Δ_c where

$f(z, c)$ is the decoding output function of latent vector z subject to the condition c (e.g., PLogP = c). When the generative model has high enough NUVR values, it is able to pinpoint the exact latent vector

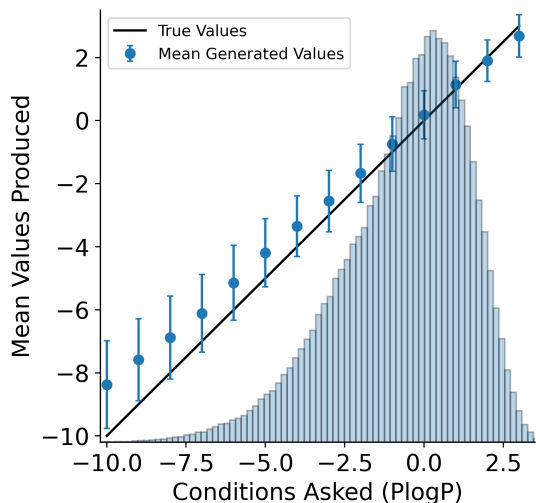


Fig. 3: CKAE correlation performance. The blue dots represent the mean PLogP values of 1,000 molecules generated by CKAE, as a function of the condition PLogP value. The error bars on each dot indicates the associated standard deviation as estimations of errors. The black line shows the ground truth 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.

location and perform an exhaustive search for all possible Δz . 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 S4.

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.

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.

We further compared CKAE to direct search using randomly sampled latent vectors with different conditions (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.

2.4 CKAE for Ligand Docking

2.4.1 Comparison to GFlowNet

Table 4 shows the performance of the CKAE 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, 41].

CKAE was trained using the same dataset of 300,000 molecules used for training GFlowNet [42], each with a binding energy calculated using AutoDock [25] (see Section 4.5). Binding energies were converted to a reward metric, using a custom scaling function. Results in Table 4 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 CKAE 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 4 shows that CKAE 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, CKAE 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 CKAE for generative extrapolation, which allows for applications to generative dataset augmentation including molecules with scoring values beyond the range of the original dataset.

Table 4: 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 are better.

Method	Top 10 reward	Top 100 reward	Top 1000 reward	Top-1000 similarity
GFlowNet	8.36	8.21	7.98	0.44
Training Data	9.62	8.78	7.86	0.58
CKAE (This work)	11.15	10.46	9.63	0.63

2.4.2 Glide Analysis

A comparison of the ligand-receptor interactions established by the top-scoring CKAE, 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 CKAE 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 π - π stacking and π -cation interactions. In addition to occupying the pocket 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 [26, 27]. 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 [25] 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 [25] calculations, as described in Section 4.5.1.

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 [25] and Glide [26, 27], 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.

3 Discussion

KAE allows the integration of the strengths of both VAE and AE frameworks in applications to molecular design. The KAE loss, with hyperparameters λ and δ , controls varying degrees of VAE and AE features as needed for the specific applications.

In the context of molecule generation, KAE outperformed VAE approaches in terms of generation validity without the need for additional chemical

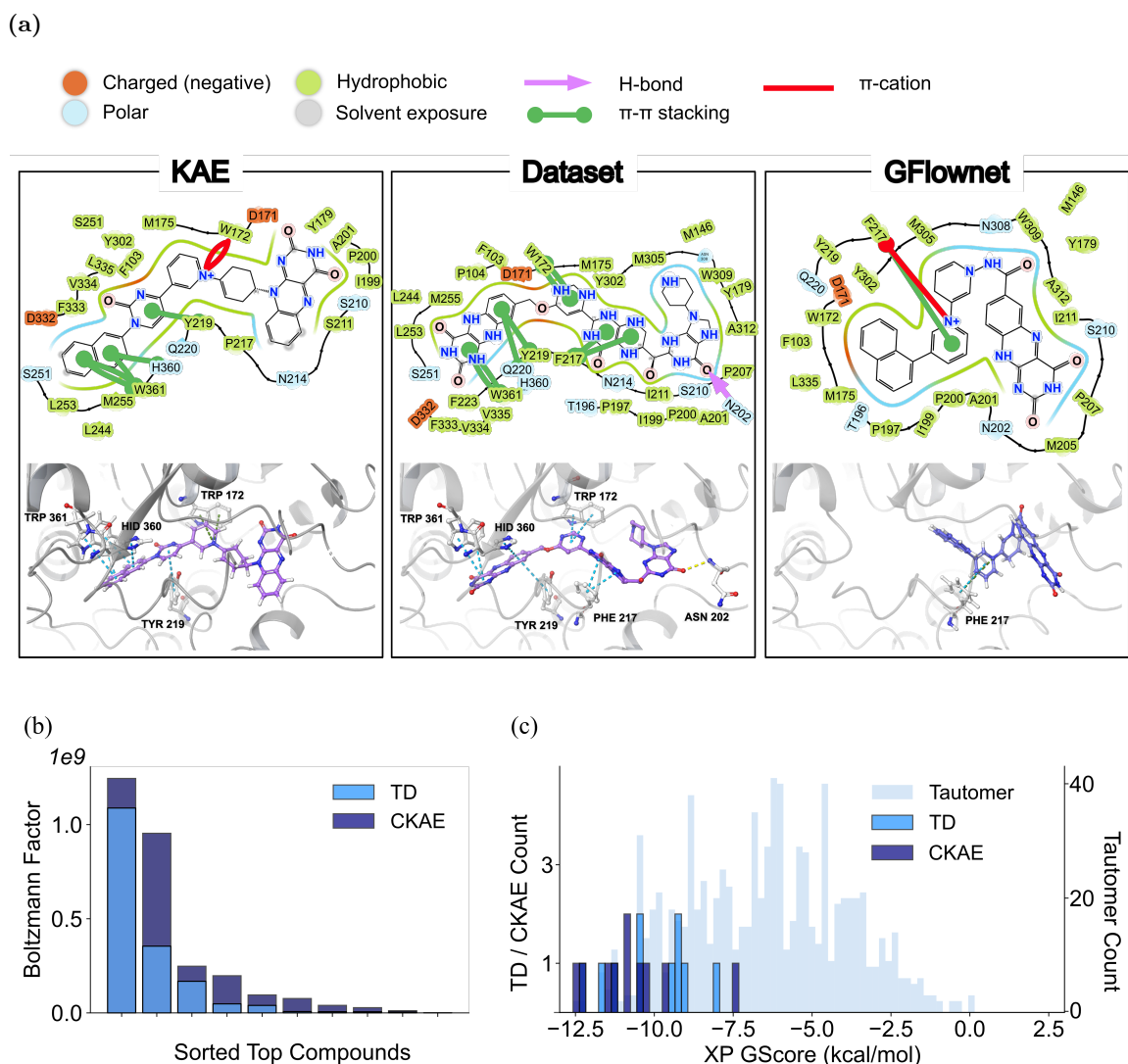


Fig. 4: Glide analysis of molecular inhibitors docked at the active site of sEH. (a) Binding interactions of top scoring molecules generated by CKAE (left), searched from the training dataset (middle), and generated by GFlowNet (right). (b) Extra Precision (XP) Glide score Boltzmann factors for the top 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.

knowledge-based checks, while achieving reconstruction performance near 100% accuracy akin to the AE. With beam search decoding [43–45] multiple candidates per latent vector can be derived. This enriched KAE’s generation diversity and validity. In the context of conditional generation, 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).

In the constrained optimization task, 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. This improvement surpasses directly searching from the training dataset by over 65%. The comparison to "Random Search" shows the strength of KAE’s accurate reconstruction which makes searching around the molecules much more efficient.

Using Glide [26, 27], the validation for CKAE’s generated high binding affinity candidates reveals that they consistently outperform those from the training dataset as well as all structurally similar tautomers, demonstrating CKAE’s ability of extrapolation and the quality of the generated molecules.

As demonstrated in this article, KAE can be used to tackle docking problems with binding affinity and constrained optimization with PlogP. Similar but unlimited to the context of molecule designs, KAE can be effectively employed to address a wide spectrum of problems, especially for those that are labeled as sample-property pairs.

4 Methods

4.1 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. Major components for KAE include the encoder, compression layer, mixing layer, and decoder.

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 separately passed through embedding layers of the encoder and decoder to become vectors of size 128. They are then added to the corresponding position embeddings of the same dimensions. Different from the original Transformer implementation that uses fixed sinusoidal functions in the representation, in this work, each positional token’s embedding is learned and updated during training.

The input is encoded by the Transformer encoder and compressed into latent space. The compression layer is a single linear layer that applies to the sequence length dimensions. This layer takes in a dimension M , the maximum sequence length in the relevant dataset and outputs a dimension of 10. In the case of ZINC250k without using conditions, M is 113 dimensional. The resulting latent tensor therefore has dimensions of $10 \times E$ where E is the embedding size of 128. The latent vectors are then added with noise from a standard Gaussian distribution. In the CKAE variant, the conditions (i.e., molecule properties) are attached with additional embeddings. Condition-multiplied 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 mixing layer is a linear layer that takes in the compressed tensors with the size $(10 + \text{number of conditions}) \times E$ and maps them back to $10 \times E$ dimensions. These tensors are treated as the new encoder output which the decoder attends to without 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, where T is the dictionary size. This T -dimensional vector is then softmaxed, resulting in a probability distribution (P) for each possible character (c).

4.2 KAE Loss

The KAE loss function is defined, as follows:

$$\mathcal{L}(\lambda, \delta) = \mathcal{L}_{WCEL}(\lambda, \delta) + m\text{-MMD}(\lambda), \quad (2)$$

where $m\text{-MMD}(\lambda)$ is a modified version of the regularizing MMD loss, discussed in Sec. 4.3, and \mathcal{L}_{WCEL} is a weighted cross-entropy loss (\mathcal{L}_{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):

$$\mathcal{L}_{CEL} = - \sum_s \sum_c Y_{s,c} \log(P_{s,c}), \quad (3)$$

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:

$$\mathcal{L}_{WCEL}(\lambda, \delta) = \frac{-1}{\lambda + \delta + 1} \left[\sum_s \sum_c Y_{s,c} \log(P_{s,c}) + (\lambda + \delta) \sum_s \sum_c Y_{s,c} \log(P_{s,c}^*) \right]. \quad (4)$$

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 λ and δ 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). The function of λ is analogous to the β parameter in β -VAE [46]. By adjusting λ and δ , 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 λ in the second term of Eq. (4) allows larger λ 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 λ in Section S5. The normalization factor of $\frac{1}{\lambda + \delta + 1}$ is derived on the basis to make a linear interpolation between the \mathcal{L}_{CEL} with and without noise.

During training, both the latent vector and the decoder outputs with and without noise are necessary for the calculation of the KAE loss. The latent vectors are penalized based on their differences from 1000 randomly sampled Gaussian vectors (\vec{G}_i) using kernel-based metrics[21]. 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 1 captures the training procedure, where information from the latent space 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 reconstructions are both penalized by \mathcal{L}_{WCEL} . The parameter λ controls the shape of the latent vector distribution and the relative weights between the MMD term and the cross-entropy loss. The parameter δ controls the relative weights between the AE and VAE objectives.

4.3 KAE m-MMD Loss

The MMD loss [47], between two distributions having N_x and N_y samples, is defined as their squared distance calculated in a space \mathcal{F} through the transformer ϕ :

$$\begin{aligned} \text{MMD}(\vec{x}, \vec{y}) &= \|\vec{\mu}_x - \vec{\mu}_y\|_{\mathcal{F}}^2, \\ &= \vec{\mu}_x^T \cdot \vec{\mu}_x + \vec{\mu}_y^T \cdot \vec{\mu}_y \\ &\quad - \vec{\mu}_x^T \cdot \vec{\mu}_y - \vec{\mu}_y^T \cdot \vec{\mu}_x, \end{aligned} \quad (5)$$

where $\vec{\mu}_x = \frac{1}{N_x} \sum_i^{N_x} \phi(\vec{x}_i)$. The space \mathcal{F} is defined by its dot product which can be calculated using a kernel function \mathcal{K} . Introducing the kernel

$$\mathcal{K}(\vec{x}_i, \vec{y}_j) = \phi(\vec{x}_i)^T \cdot \phi(\vec{y}_j), \quad (6)$$

we can write the inner products, as follows:

$$\vec{\mu}_x^T \cdot \vec{\mu}_y = \frac{1}{N_x N_y} \sum_i^{N_x} \sum_j^{N_y} \mathcal{K}(\vec{x}_i, \vec{y}_j), \quad (7)$$

so

$$\begin{aligned} \text{MMD}(\vec{x}, \vec{y}) &= \frac{1}{N_y^2} \sum_i^{N_y} \sum_j^{N_y} \mathcal{K}(\vec{y}_i, \vec{y}_j) \\ &\quad + \frac{1}{N_x^2} \sum_i^{N_x} \sum_j^{N_x} \mathcal{K}(\vec{x}_i, \vec{x}_j) \\ &\quad - \frac{2}{N_x N_y} \sum_{i'}^{N_x} \sum_{j'}^{N_y} \mathcal{K}(\vec{x}_{i'}, \vec{y}_{j'}), \end{aligned} \quad (8)$$

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

$$\mathcal{K}(\vec{\alpha}, \vec{\beta}) = \exp\left(\frac{-\frac{1}{D} \sum_{d=0}^D (\alpha_d - \beta_d)^2}{2\sigma^2}\right), \quad (9)$$

where $D = 10 \times E$ is the size of the latent dimension and $\sigma = \sqrt{0.32}$ has been empirically chosen (see comparison in S8).

The first term in the r.h.s. of Eq. (8) corresponds to $\vec{\mu}_y^T \cdot \vec{\mu}_x$. 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:

$$s\text{-MMD}(\lambda) = \lambda \left[\frac{1}{N_x^2} \sum_i^{N_x} \sum_j^{N_x} \mathcal{K}(\vec{x}_i, \vec{x}_j) - \frac{2}{N_x N_y} \sum_{i'}^{N_x} \sum_{j'}^{N_y} \mathcal{K}(\vec{x}_{i'}, \vec{y}_{j'}) \right]. \quad (10)$$

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

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

$$m\text{-MMD}(\lambda) = \lambda \left[1 - \frac{1}{N_x N_y} \sum_i^{N_x} \sum_j^{N_y} \mathcal{K}(\vec{x}_i, \vec{y}_j) \right] \quad (11)$$

The constant 1 is added to make m-MMD range from 0 to 1 before the λ scaling. A comparative analysis of the effect of using m-MMD versus s-MMD is provided in Supplementary Information (Section S6).

4.4 Decoding Methods

KAE’s generation process involves sampling a vector, $\vec{v} \in \mathcal{R}^{10 \times E}$ 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 mingled by a fully connected layer, yielding \vec{L} again in $\mathcal{R}^{10 \times E}$. 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. One of the approaches 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 end-of-sequence token (?) is produced or the sequence length limit is achieved. Besides 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 $s, s-1, s-2, \dots$ to 0 can be represented, as follows:

$$\begin{aligned} P(s, s-1, s-2, \dots, 0) \\ = P(s|s-1, s-2, \dots, 0) \times P(s-1, s-2, \dots, 0) \end{aligned} \quad (12)$$

This can be interpreted as the product of individual probabilities,

$$\begin{aligned} P(s, s-1, s-2, \dots, 0) = P(s|s-1, s-2, \dots, 0) \\ \times P(s-1|s-2, s-3, \dots, 0) \times \dots P(0) \end{aligned} \quad (13)$$

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:

$$P_i = \frac{1}{\sqrt{N_i}} \sum_{s \neq \text{pad}}^S \text{Log}(P_{i,s}) \quad (14)$$

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 $\frac{1}{\sqrt{N_i}}$ 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 end-of-sequence token, signaling the cessation of decoding.

4.5 Docking Methods

The generated molecular structures were evaluated using Autodock Vina [25], 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. [26, 27] 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.

4.5.1 Glide calculations

The model protein receptor (PDB ID: 4jnc) was prepared by using the adept Protein Preparation Wizard tool in Maestro [48]. The protonation states were defined at a neutral pH = 7.0. The protein was subsequently refined via energy minimization using the OPLS4 force field [49].

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 [49]. The XP (extra precision) [26, 27] 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 [50] to the docking scores [48], 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 [25] calculations.

Acknowledgments

VSB acknowledges support from the NSF CCI grant (Award No. 2124511) and computational resources from the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award BES-ERCAP0024372.

Data Availability

Pretrained KAE can be accessed through API calls at <https://demo.ischemist.com/login>. The key to viewing KAE example results is [publicdemo](#). To use the API, please contact victor.batista@yale.edu.

References

- [1] Lukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):1–18, 2020.
- [2] Michael Moret, Lukas Friedrich, Francesca Grisoni, Daniel Merk, and Gisbert Schneider. Generative molecular design in low data regimes. *Nature Machine Intelligence*, 2(3):171–180, 2020.
- [3] Miha Skalic, José Jiménez, Davide Sabbadin, and Gianni De Fabritiis. Shape-based generative modeling for de novo drug design. *Journal of chemical information and modeling*, 59(3):1205–1214, 2019.
- [4] Jike Wang, Chang-Yu Hsieh, Mingyang Wang, Xiaorui Wang, Zhenxing Wu, Dejun Jiang, Benben Liao, Xujun Zhang, Bo Yang, Qiaojun He, et al. Multi-constraint molecular generation based on conditional transformer, knowledge distillation and reinforcement learning. *Nature Machine Intelligence*, 3(10):914–922, 2021.
- [5] Yoshua Bengio, Tristan Deleu, Edward J Hu, Salem Lahlou, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *arXiv preprint arXiv:2111.09266*, 2021.
- [6] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.
- [7] Oleksii Prykhodko, Simon Viet Johansson, Panagiotis-Christos Kotsias, Josep Arús-Pous, Esben Jannik Bjerrum, Ola Engkvist, and Hongming Chen. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics*, 11(1):1–13, 2019.
- [8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [9] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.
- [10] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pages 1945–1954. PMLR, 2017.
- [11] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.
- [12] Dana H Ballard. Modular learning in neural networks. In *Aaai*, volume 647, pages 279–284, 1987.
- [13] Samuel C Hoffman, Vijil Chenthamarakshan, Kahini Wadhawan, Pin-Yu Chen, and Payel Das. Optimizing molecules using efficient queries from property evaluations. *Nature Machine Intelligence*, 4(1):21–31, 2022.
- [14] Han Van De Waterbeemd, Dennis A Smith, Kevin Beaumont, and Don K Walker.

- Property-based design: optimization of drug absorption and pharmacokinetics. *Journal of medicinal chemistry*, 44(9):1313–1333, 2001.
- [15] Jiazhen He, Huifang You, Emil Sandström, Eva Nittinger, Esben Jannik Bjerrum, Christian Tyrchan, Werngard Czechtizky, and Ola Engkvist. Molecular optimization by capturing chemist’s intuition using deep neural networks. *Journal of cheminformatics*, 13(1):1–17, 2021.
- [16] Ziqi Chen, Martin Renqiang Min, Srinivasan Parthasarathy, and Xia Ning. A deep generative model for molecule optimization via one fragment modification. *Nature machine intelligence*, 3(12):1040–1049, 2021.
- [17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [18] Orion Dollar, Nisarg Joshi, David AC Beck, and Jim Pfandtner. Attention-based generative models for de novo molecular design. *Chemical Science*, 12(24):8362–8372, 2021.
- [19] Junyan Jiang, Gus G Xia, Dave B Carlton, Chris N Anderson, and Ryan H Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 516–520. IEEE, 2020.
- [20] Tianming Wang and Xiaojun Wan. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *IJCAI*, pages 5233–5239, 2019.
- [21] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infvae: Balancing learning and inference in variational autoencoders. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 5885–5892, 2019.
- [22] Talip Ucar. Bridging the elbo and mmd. *arXiv preprint arXiv:1910.13181*, 2019.
- [23] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015.
- [24] Ryan J Richards and Austen M Groener. Conditional β -vae for de novo molecular generation. *arXiv preprint arXiv:2205.01592*, 2022.
- [25] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.
- [26] Thomas A. Halgren, Robert B. Murphy, Richard A. Friesner, Hege S. Beard, Leah L. Frye, W. Thomas Pollard, and Jay L. Banks. Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening. *Journal of Medicinal Chemistry*, 47(7):1750–1759, 2004.
- [27] Richard A. Friesner, Robert B. Murphy, Matthew P. Repasky, Leah L. Frye, Jeremy R. Greenwood, Thomas A. Halgren, Paul C. Sanschagrin, and Daniel T. Mainz. Extra precision glide: Docking and scoring incorporating a model of hydrophobic enclosure for protein-ligand complexes. *Journal of Medicinal Chemistry*, 49(21):6177–6196, 2006.
- [28] Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626, 2020.
- [29] Chaochao Yan, Jinyu Yang, Hehuan Ma, Sheng Wang, and Junzhou Huang. Molecule sequence generation with rebalanced variational autoencoder loss. *Journal of Computational Biology*, 2022.
- [30] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*, pages 7192–7203. PMLR, 2021.

- [31] Zaccary Alperstein, Artem Cherkasov, and Jason Tyler Rolfe. All smiles variational autoencoder. *arXiv preprint arXiv:1905.13343*, 2019.
- [32] Greg Landrum et al. Rdkit: Open-source cheminformatics software. 2016.
- [33] Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. In *International Conference on Machine Learning*, pages 3183–3191. PMLR, 2019.
- [34] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.
- [35] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.
- [36] Chencheng Xu, Qiao Liu, Minlie Huang, and Tao Jiang. Reinforced molecular optimization with neighborhood-controlled grammars. *Advances in Neural Information Processing Systems*, 33:8366–8377, 2020.
- [37] Scott A Wildman and Gordon M Crippen. Prediction of physicochemical parameters by atomic contributions. *Journal of chemical information and computer sciences*, 39(5):868–873, 1999.
- [38] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11, 2009.
- [39] Jaechang Lim, Seongok Ryu, Jin Woo Kim, and Woo Youn Kim. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of cheminformatics*, 10(1):1–9, 2018.
- [40] Changsheng Ma and Xiangliang Zhang. Gf-vae: a flow-based variational autoencoder for molecule generation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1181–1190, 2021.
- [41] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.
- [42] M.; Korablyov M.; Precup D.; Bengio Y. Bengio, E.; Jain. Flow network based generative models for non-iterative diverse candidate generation. <https://github.com/GFNORG/gfnet>, 2022.
- [43] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.
- [44] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8624–8628. IEEE, 2013.
- [45] Jeff Guo and Philippe Schwaller. Beam enumeration: Probabilistic explainability for sample efficient self-conditioned molecular design. *arXiv preprint arXiv:2309.13957*, 2023.
- [46] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.
- [47] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- [48] New York NY Schrödinger, LLC. Schrödinger release 2023-2: Maestro, 2023.
- [49] Chao Lu, Chuanjie Wu, Delaram Ghoreishi, Wei Chen, Lingle Wang, Wolfgang Damm,

Gregory A. Ross, Markus K. Dahlgren, Ellery Russell, Christopher D. Von Bargen, Robert Abel, Richard A. Friesner, and Edward D. Harder. OPLS4: Improving Force Field Accuracy on Challenging Regimes of Chemical Space. *Journal of Chemical Theory and Computation*, 17(7):4291–4300, 2021.

- [50] Jeremy R. Greenwood, David Calkins, Arron P. Sullivan, and John C. Shelley. Towards the comprehensive, rapid, and accurate prediction of the favorable tautomeric states of drug-like molecules in aqueous solution. *Journal of Computer-Aided Molecular Design*, 24(6-7):591–604, 2010.

Supplemental Materials: Kernel-Elastic Autoencoder for Molecular Design

S1 Training Datasets

The KAE model has been trained on 90% (225,000) of the entries of the ZINC250K dataset. 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 GFlowNet, all entries were used for training.

S2 Data Preparation

We used the ZINC250K dataset. During dataset preparation, all SMILES strings were first canonicalized and added to the start of sequence tokens “!” and the end of sequence tokens “?”. The canonicalization gives a unique and unambiguous representation of the molecule. 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. Paddings were added at the end as the 42^{nd} 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 Comparison with Scaled KL Divergence

In Figure 2, we initially set all λ values to 1. However, this choice may not represent the optimal λ value for KL models when conducting a fair comparison with m-MMD models. To explore the impact of varying λ values on KL models, we conducted a similar analysis as depicted in Figure S1. The results indicate that higher λ values tend to enhance validity at the expense of reduced reconstruction rates. Specifically, for the KL model, the optimal λ value is found around 1.5, yielding an NUVR value of 0.51, whereas the model applying m-MMD, at $\lambda = 1$, achieves a NUVR of 0.85. It’s worth noting that although KL models incorporate an additional layer for variance prediction, the increase in parameters is minimal compared to the overall parameter count (1e-3 over the total parameter counts). Thus, this comparison appropriately underscores the advantage of the m-MMD loss in terms of the NUVR metric.

S4 Similarity Exhaustion Search Procedures

The hyperparameters of SES include the beam size (B), the interval (δ_s), the maximum increase in condition (Δ), 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, Δ . In total, $B + \frac{B\Delta}{\delta_s}$ 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

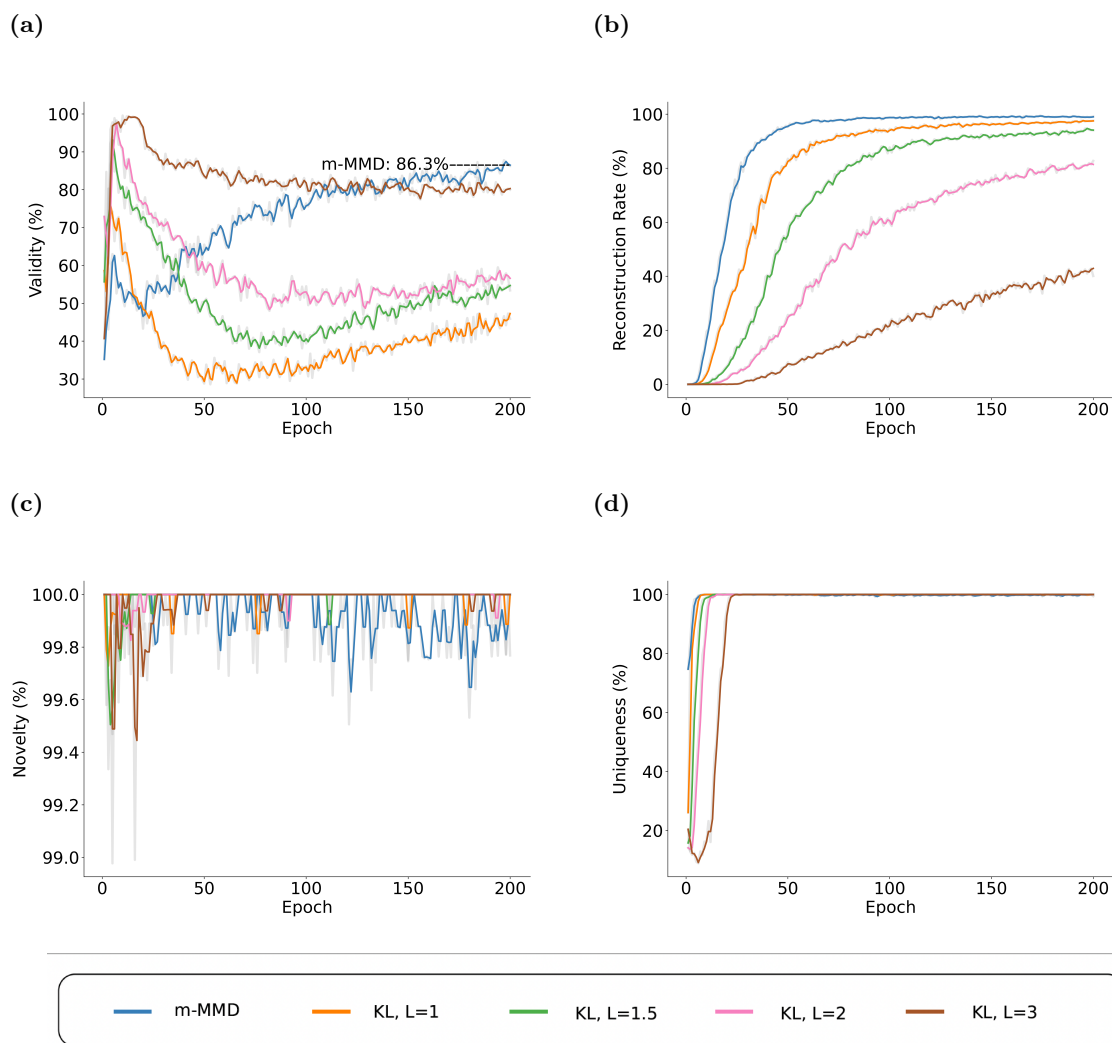


Fig. S1: Comparison of learning rates for models trained with different λ values for KL divergence loss. (a) Validity evaluated at each epoch. (b) Fraction of molecules properly reconstructed as a function of epochs. (c) Novelty evaluated at each epoch. (d) The uniqueness at each epoch. The models are trained with the loss $\mathcal{L}_{VAE} = \mathcal{L}_{CEL} + KL(\lambda)$. The m-MMD model with the loss $\mathcal{L}_{CEL} + m\text{-MMD}(\lambda = 1)$.

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 ϵ 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 S2.

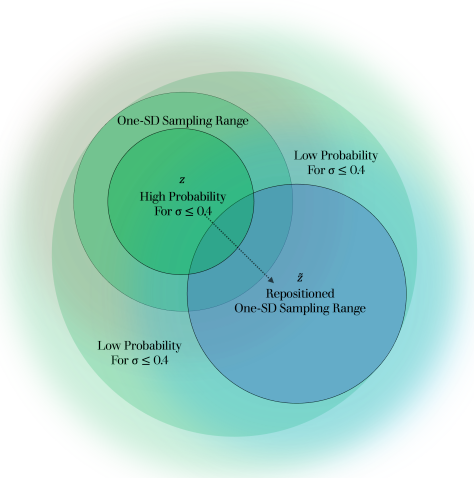


Fig. S2: Repositioning. A \tilde{z}_i is selected around z_i if the generated molecule falls within the similarity threshold (σ) and exhibits an improvement in the optimized property. The subsequent search repetition is then conducted around \tilde{z}_i . Through repositioning, the search space expands for molecules that showed little improvements during the condition search.

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 $c_i + j\delta_s$, 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.

S5 Impact of the λ Parameter

The reason for increasing λ is similar to that of increasing β in the case for β -VAE by Richards et al. Both λ in KAE and β in β -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 λ and β are set to one for KAE and β -VAE.

For the best model using m-MMD in Figure 2, all validities are lower than 90%. This can be improved by increasing the λ 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. δ values were set to $-\lambda$ throughout the training process to exclude any effects from \mathcal{L}_{WCEL} in the comparison.

Increasing the λ value tightens the placement of latent vectors together in the Gaussian distribution penalized by the m-MMD loss. This is reflected by 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 λ values can be attributed to the decoder more frequently identifying different molecules with overlapping latent representations as the same ones.

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

Table S1: Model performance with varying λ . 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 λ values (loss functions are then $\mathcal{L}(\lambda = \lambda, \delta = -\lambda)$).

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

We seek to find a solution that can increase validity while maintaining other metrics at the same level. Therefore controlling the model via \mathcal{L}_{WCEL} was the key to this problem. Model performance with a range of δ values were compared in S8 and $\delta = 1$ was chosen in \mathcal{L}_{WCEL} ; We then compared different λ values with δ fixed to 1. In Figure S3, models are trained with the loss function $\mathcal{L}(\lambda, \delta = 1)$ for 200 epochs. It can be observed in Figure S3a that higher λ values lead to better final validity. However, uniqueness in Figure S3b breaks down for the case of when $\lambda = 4$ while novelty and reconstruction rates converge to around 100% in (Figure S3c and Figure S3d). Therefore, the $\lambda = 3.5$ model is trained for additional 200 epochs (total 400 epochs) to give final performance metrics in Table 1.

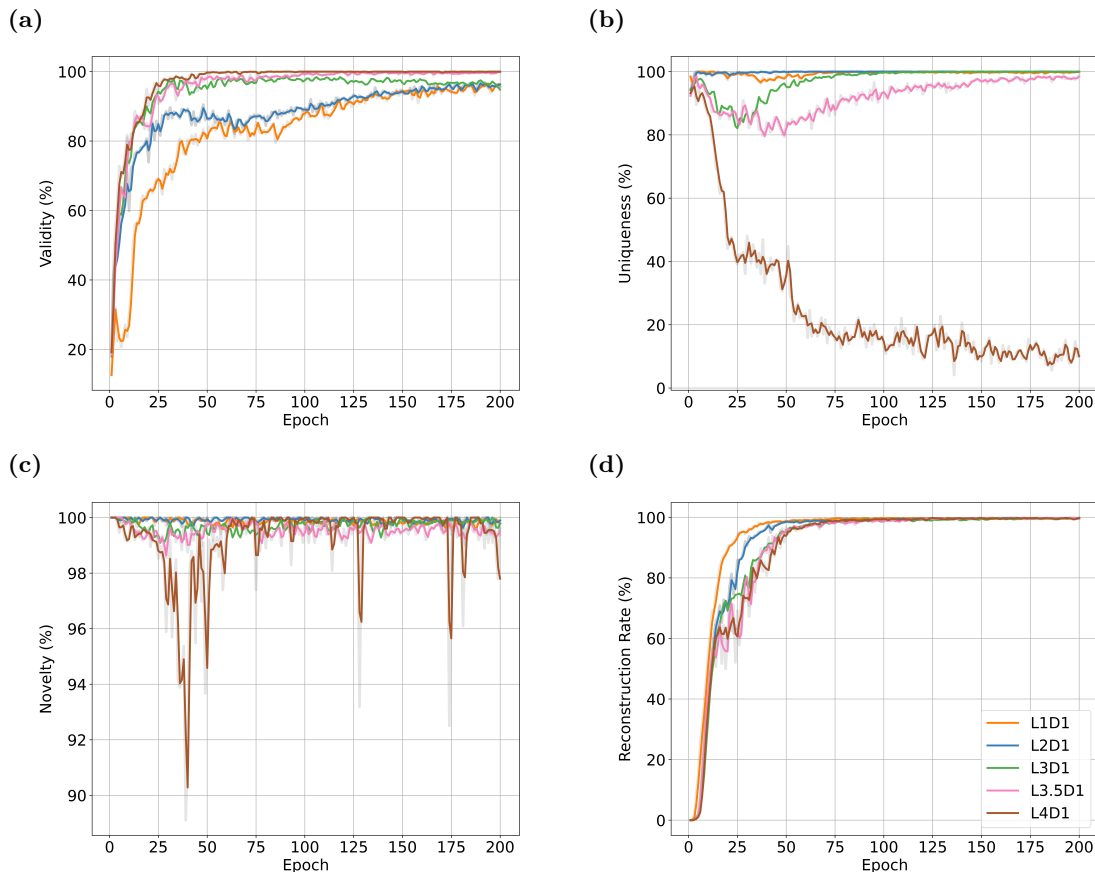


Fig. S3: The performance comparison of models trained with different λ values, while keeping $\delta = 1$, using the m-MMD loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. The 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 LxDy represents a model trained with $\lambda = x$ and $\delta = y$. For instance, the model labeled L3D1 corresponds to $\mathcal{L}(\lambda = 3, \delta = 1)$.

S6 Latent Space And Model Performance

In m-MMD, with the RBF-kernel function, removing the $\vec{\mu}_x^T \vec{\mu}_x$ 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 S4.

It can be seen from Figure 2b 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 $\mathcal{K}(\vec{x}, \vec{x})$ 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 λ as an approach to optimize the model performance in N, U, and V, reducing the uninterpreted regions while still making individual molecules distinct from each other.

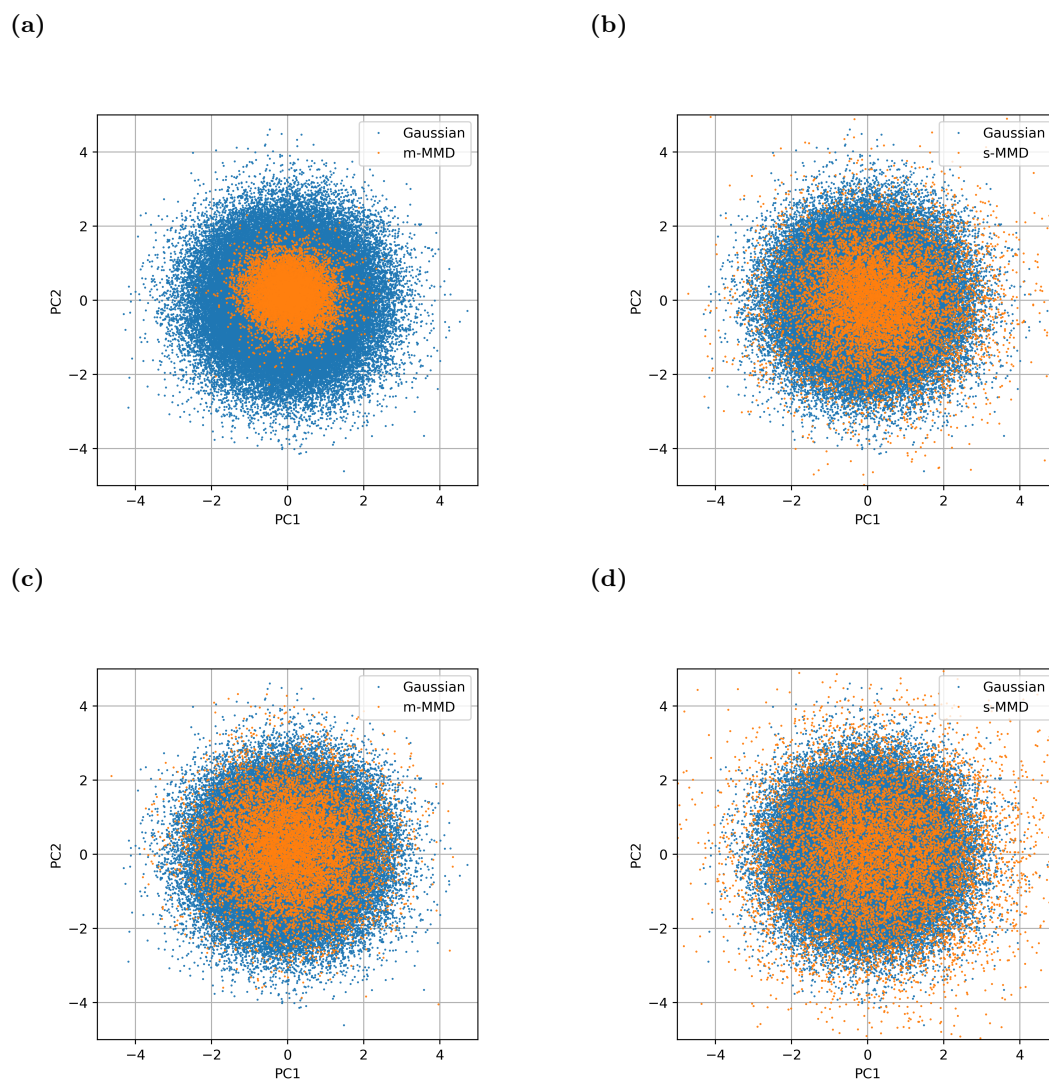


Fig. S4: Latent vectors obtained by passing 10k ZINC250K molecules to the encoder and transforming under the same principal components extracted from the standard Gaussian distribution. (a) m-MMD results showing all latent vectors well-incorporated in the Gaussian. (b) s-MMD loss makes the latent vectors 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.

S7 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 4.4).

Similar to previous studies that have employed checking methods to enhance model performance, we propose a new approach with 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 instance,

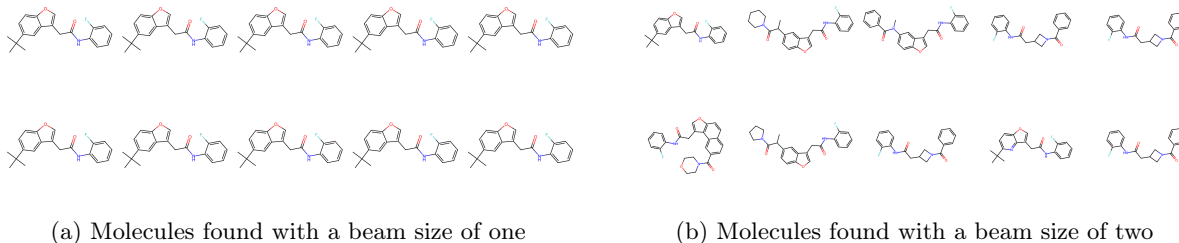


Fig. S5: Molecules obtained by sampling from the 0.1-SD Gaussian distribution centered around a specific latent vector, while varying the beam size. Figure (a) illustrates the molecules found using a beam size of one, where only the original encoded molecule is identified despite the added noise. In 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.

Beam Size	Novelty	Uniqueness	Validity	NUV
1	0.996	0.974	0.998	0.968
2	1.000	0.996	1.000	0.996
3	1.000	0.998	1.000	0.998
4	1.000	1.000	1.000	1.000
5	1.000	1.000	1.000	1.000

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. If any of the generated SMILES strings are novel, unique, and valid, the evaluation process is stopped and the corresponding SMILES string is retained. 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. For a latent vector, if all its beam search results fail to meet the validity criterion, the top-one result is returned. We believe the beam search can help differentiate 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. 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 Fig . 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

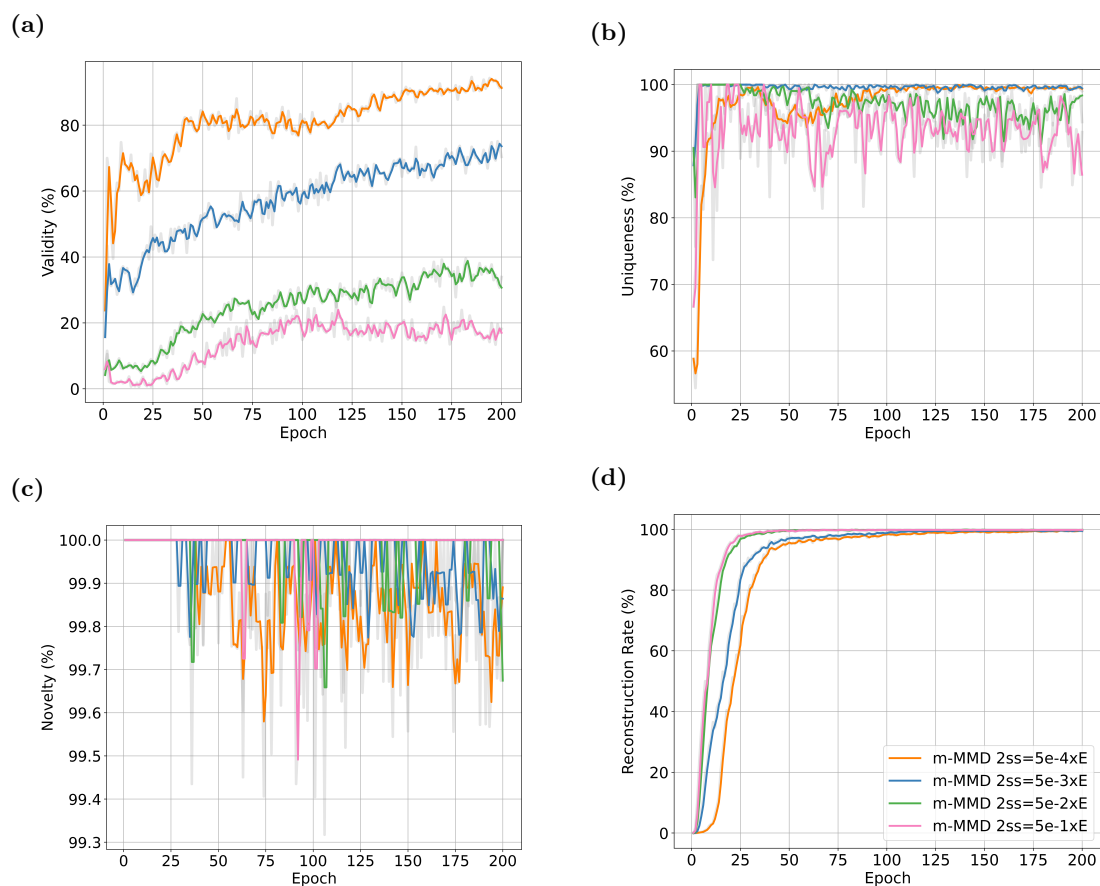


Fig. S6: Performance comparison of the models trained with different sigma values using modified MMD loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. Note that 2ss (2 sigma squared) 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.

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).

S8 σ Comparison

In Figure S6 and Figure S7, 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

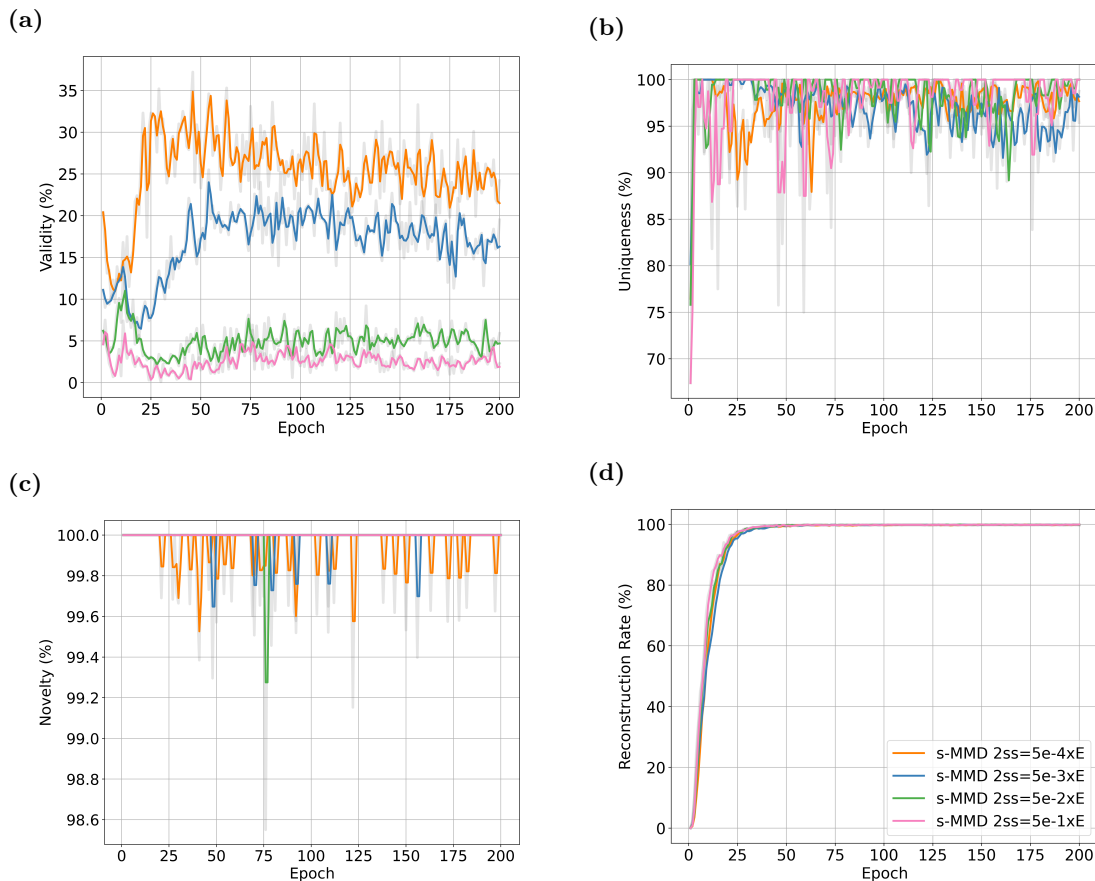


Fig. S7: Performance comparison of the models trained with different sigma values using standard MMD loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. Note that $2\sigma^2$ (2 sigma 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.

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).

S9 δ Comparison

We designed the δ such that when λ is 1, and δ is greater than -1, the AE-like term contributes to the model reconstruction performance. When δ is large, the model ignores the regions with added noise and thus is turned into a pure auto-encoder. When δ is equal to -1, the model is VAE-like where each latent vector is treated as a distribution. When δ is in between these two extrema, the model achieves the AE-like reconstruction rate while obtaining better generative performance in NUV metrics (Figure S8).

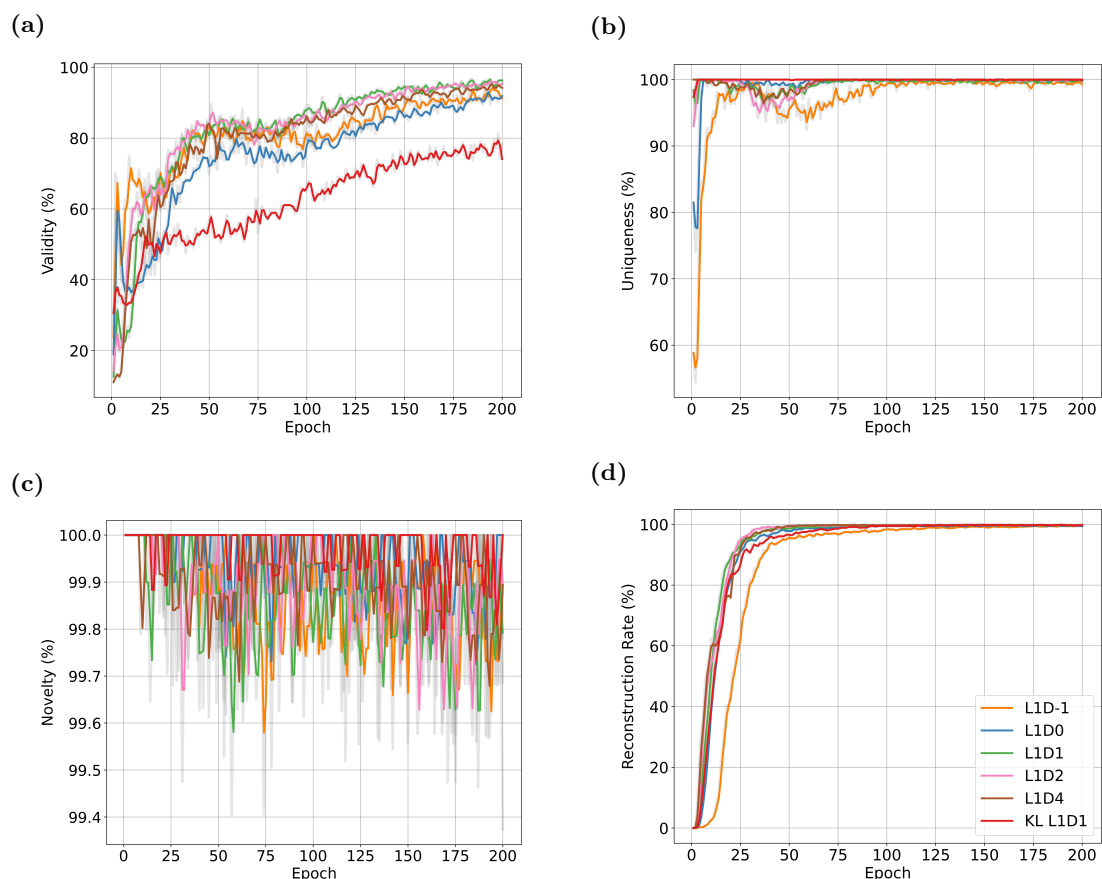


Fig. S8: Performance comparison of the models trained with different δ values (with $\lambda = 1$) using modified MMD loss and KL loss. (a) Validity evaluated at every epoch. (b) Uniqueness evaluated at every epoch. (c) Novelty evaluated at every epoch. (d) Reconstruction rate evaluated at every epoch. 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 $\lambda = x$ and $\delta = y$. For example, the model labeled with L1D-1 is trained with $\mathcal{L}(\lambda = 1, \delta = -1)$

S10 KAE Interpolation

KAE has near 100% reconstruction rate and this gives it an advantage to perform modifications precisely around the input molecules. We show an example by linearly interpolating from Atrazine to Plastoquinone. The two molecules were encoded into their latent representations and the model decoded Atrazine to the Plastoquinone in 100 evenly spaced steps with a beam size of 30. The NUV molecules along this trajectory is plotted in Figure S9. The result shows an effective mixing/transitioning of the two molecules' motifs and exact reconstructions of the starting and ending compounds.

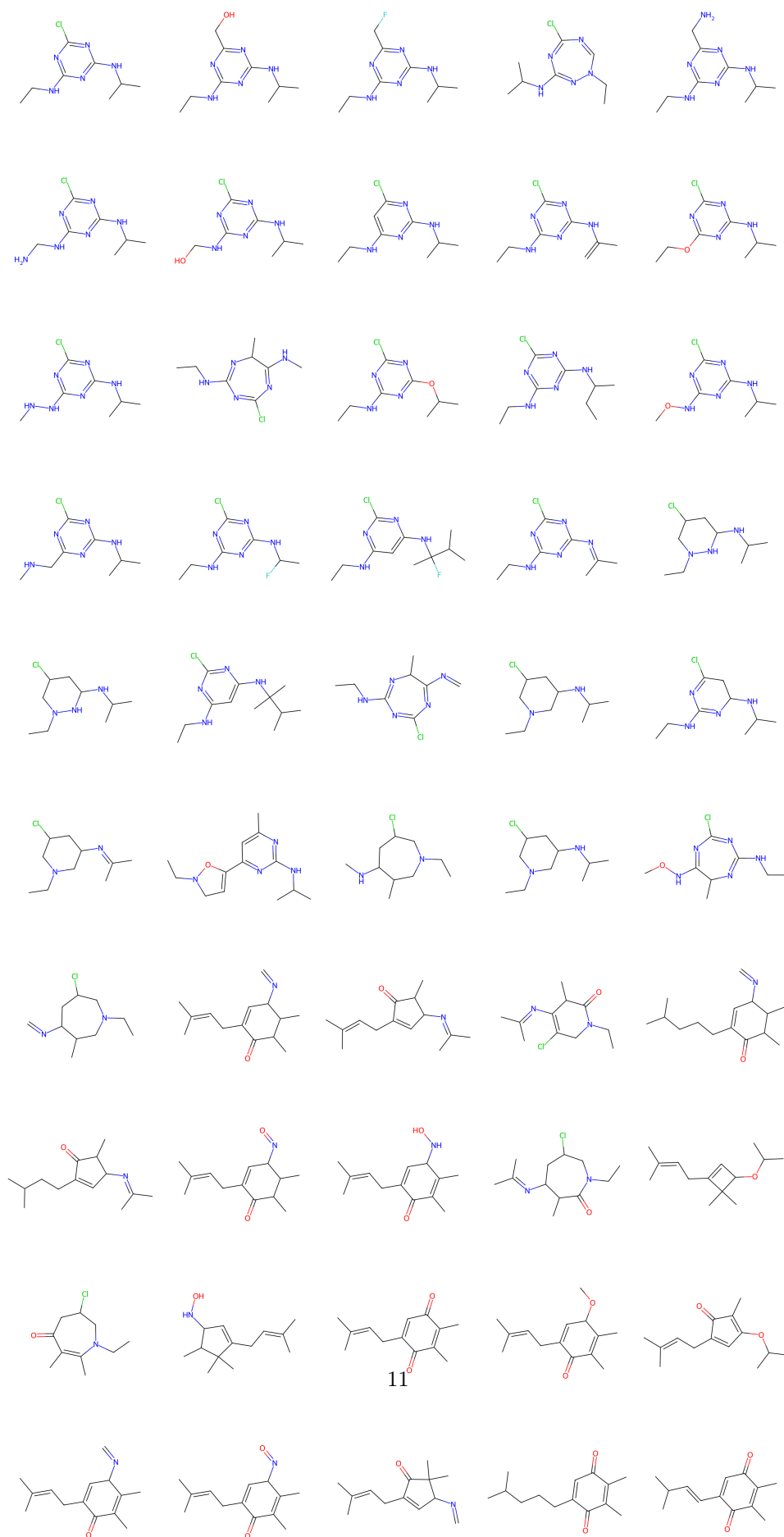


Fig. S9: Interpolation of KAE Latent Space Atrazine and Plastoquinone are encoded into two latent vectors. The model then decodes from Atrazine to Plastoquinone in 100 evenly spaced steps with a beam size of 30.