Supporting Information

PolyRL: Reinforcement Learning-Guided Polymer Generation for Multi-Objective Polymer Discovery

Wentao Li,[$, a] Yijun Li,[$, b] Qi Lei,[$, a] ,Zemeng Wang,[a] Xiaonan Wang*[, a]

[a] Department of Chemical Engineering, Tsinghua University, Beijing 100084, P. R. China

[b] Tanwei College, Tsinghua University, Beijing 100084, P. R. China

[$] These authors contributed equally to this paper

Corresponding Authors: Xiaonan Wang, wangxiaonan@tsinghua.edu.cn

Summary: 37 Pages, 17 Tables, 19 Figures

List of Supporting Information

Catalog

1 . Experimental Workflow and System Overview

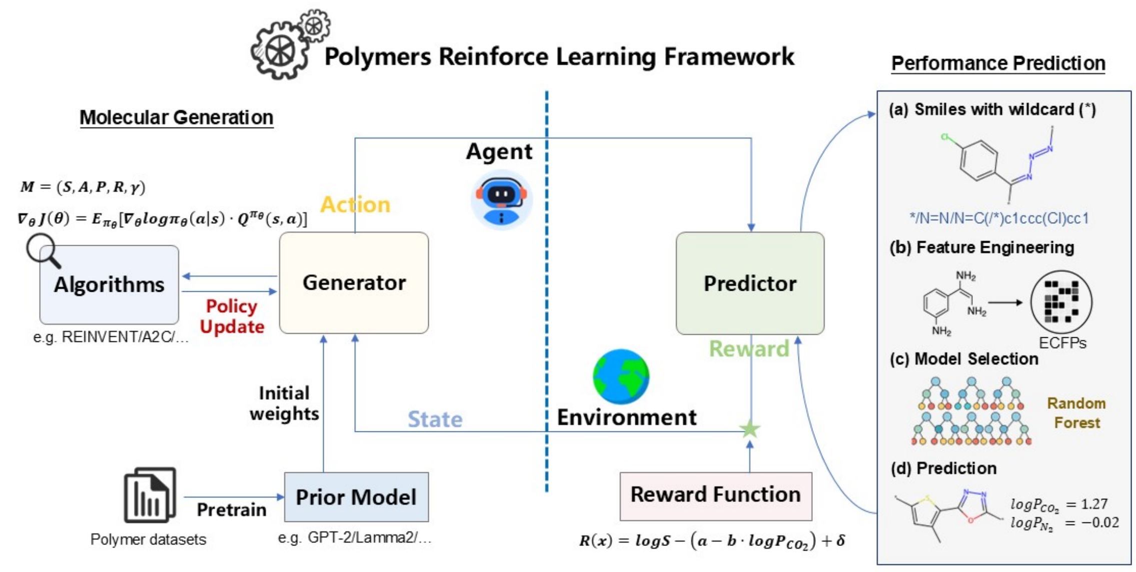

Figure S1. PolyRL framework

This study aims to achieve multi-objective performance-directed optimization of polymer materials for gas separation, specifically targeting the simultaneous enhancement of CO_2 permeability and CO_2/N_2 selectivity. To this end, we developed a closed-loop material design framework that incorporates reinforcement learning, enabling the iterative generation of high-performance molecular structures.

The PolyRL framework consists of the following three core modules:

Property prediction models: Machine learning-based models are used to predict gas separation properties (e.g., CO_2 and N_2 permeability) from polymer structures. These models serve as surrogate reward functions during reinforcement learning to guide the optimization of molecular structures.

Polymer generative models: Deep generative models—including GRU,LSTM, GPT-2, and LLaMA2—are pretrained on polymer SMILES sequences to learn their syntactic patterns and are used to generate new candidate molecules.

Reinforcement learning algorithms and optimization: The pretrained generative models are fine-tuned as policy networks under the guidance of reinforcement learning algorithms (e.g., REINVENT, REINFORCE, AHC), to improve the target performance of the generated polymers.

After training, molecular dynamics (MD) simulations are used to validate the structural stability and separation performance of the generated molecules.

Together, these modules form a closed-loop system encompassing “prediction–generation–optimization–validation”, offering a scalable design paradigm for the efficient discovery of gas separation polymers.

2 Methods and Modules

2.1 Datasets and Preprocessing

To construct a robust reinforcement learning–driven framework for polymer design, we prepared two main datasets: one for training the property prediction model, and another for pretraining the generative model.

(1) Property Prediction Dataset

The dataset used for property prediction is primarily derived from the work of Yang et al. ,[1] which compiles experimentally measured gas permeabilities of various polymer membranes, particularly for CO_2/N_2 separation. The original data undergo manual curation to remove duplicate entries and records with incomplete information. After cleaning, the dataset contains a total of 353 unique polymer samples, each with recorded CO_2 and N_2 permeability values.

All polymer structures are represented using P-SMILES (Polymer SMILES), a linear string format specifically designed to encode repeating units of polymers. Unlike conventional SMILES, P-SMILES uses one or more asterisks (*) to denote the boundaries of repeating units, enabling explicit representation of linear polymers (e.g., [*]...[*]) and ladder-like structures. This representation is particularly suitable for language model–based molecular generation and downstream property prediction tasks. The dataset is randomly split into a training set (80%, 282 samples) and a test set (20%, 71 samples) for model development.

Table S1. Count of each atom type present in prediction dataset.

C	O	*	F	N	Si

Count	8086	991	762	425	335	62
	S	**Cl**	**Br**	**P**		
Count	44	26	12	11		

To further understand the chemical diversity and feature composition of the experimental dataset, we performed an analysis of functional group occurrences and molecular representation patterns.

Functional groups were identified using SMARTS pattern matching implemented via the RDKit library. A curated list of SMARTS patterns covering common organic moieties—such as hydroxyl (–OH), carbonyl (C=O), carboxyl (–COOH), amino (–NH_2), and aromatic rings—was applied to all polymer fragments represented by P-SMILES.

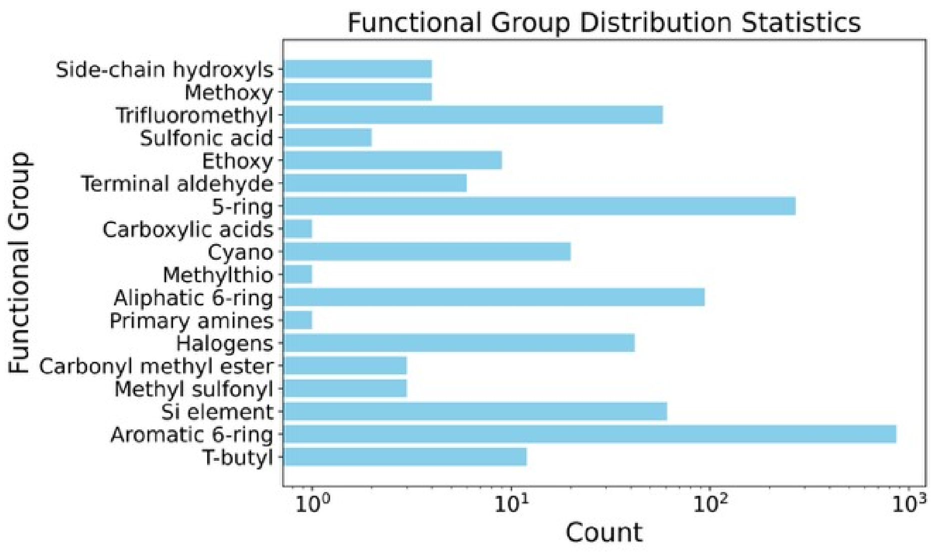

Figure S2. Functional groups distribution of the experimental dataset

From the Figure S2, it can be observed that aromatic rings (Aromatic 6-ring), five-membered rings (5-ring), and silicon-containing groups (Si element) appear most frequently, far exceeding other functional groups. This indicates that the polymers in this dataset are predominantly composed of rigid backbones, silicon-containing segments, and aromatic structures, which aligns well with the typical design features of gas separation membrane materials—high rigidity and steric hindrance contribute to enhanced size selectivity and anti-swelling performance. In addition, carboxylic acids, carbonyl methyl esters, ethers (Ethoxy/Methoxy), and primary amines are also present to a noticeable extent. These polar functional groups are often employed to tune the interactions between polymer matrices and gas molecules, thereby influencing both permeability and selectivity of the membranes.

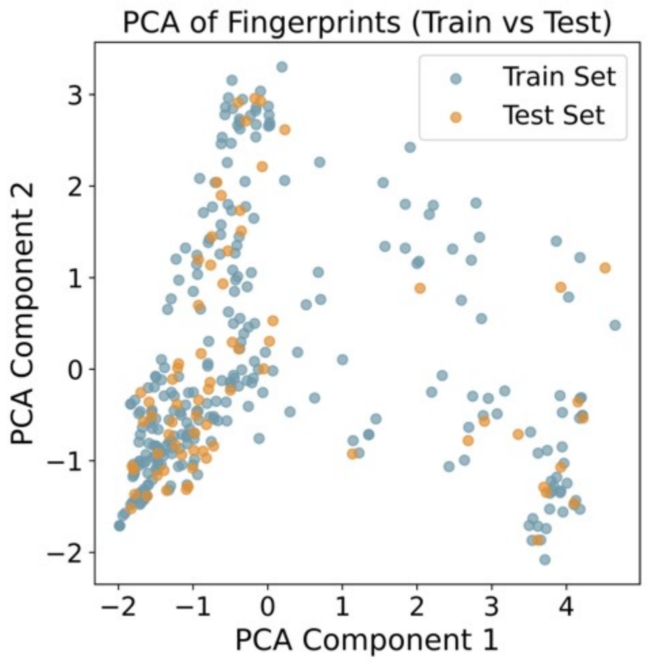

Figure S3. PCA of P-SMILES fingerprints in the experimental dataset

To assess structural diversity and feature clustering, the P-SMILES fingerprints were embedded into a low-dimensional space using Principal Component Analysis (PCA). As shown in Figure S3, the training and test data exhibit a partially uniform distribution in the projection space, indicating that the experimental dataset covers a diverse and well-balanced chemical space.

(2) Generative Model Pretraining Corpus

The pretraining corpus for the polymer generative model is constructed based on known polymer structures collected from the publicly available PoLyInfo database. First, polymer repeating units with well-defined structures and standardized annotations are extracted from PoLyInfo and converted into canonical SMILES representations. A recurrent neural network (RNN) is then trained on these sequences to learn the structural connectivity patterns and syntactic rules inherent to polymer chemistry.[2]

Based on the trained model, a large number of hypothetical polymer structures are generated via autoregressive sampling. All generated sequences are subsequently converted into P-SMILES format for consistency and structural clarity. This process yields a curated dataset of approximately 995,799 chemically valid polymer sequences. The resulting corpus not only preserves the distribution characteristics of the original chemical space but also densifies underrepresented regions, thereby enhancing the generative model's ability to learn diverse polymer structures and improving its generalization performance during downstream tasks.

Table S2. Count of each atom type present in generative model pretraining dataset.

	C	**O**	*****	**N**	**F**	**S**
Count	20755662	3736746	1977534	1158075	506399	285990
	Si	**Cl**	**P**	**Br**	**H**	**Na**
Count	128670	72755	38694	34130	14831	5365
	I	**Se**	**Sn**	**Ge**	**B**	**Fe**
Count	5123	3563	2339	2018	1264	585
	As	**Zn**	**Ca**	**Pb**	**K**	**Ni**
Count	381	205	142	107	90	78
	Cd	**Te**	**Co**			
Count	68	67	49			

2.2 Property Prediction Models

This study develops a property prediction model based on the Random Forest (RF) algorithm to estimate the gas permeabilities of polymers for CO_2 and N_2 (expressed as log-scale values). The resulting model is subsequently embedded within the reinforcement learning framework as a surrogate reward function (Section 2.4.1).

Model Architecture and Hyperparameters

The Random Forest model is implemented using the following hyperparameter settings: number of trees = 200 (n_estimators = 200), maximum depth = 20 (max_depth = 20), maximum number of features per split set to the square root of the total (max_features = "sqrt"), and bootstrap sampling enabled (bootstrap = True). The default Gini impurity is used as the splitting criterion, and mean squared error (MSE) is employed for node evaluation during regression.

Molecular Fingerprints and Input Features

To encode the structural features of polymers, we compute Morgan fingerprints from the P-SMILES representations of repeating units using the RDKit toolkit.[3,4,5] The fingerprint radius is set to 3, capturing atom environments up to three bonds away. The initial fingerprint space includes 3209 unique substructures, from which we retain the 114 most frequently occurring substructures as final input features. This produces a sparse yet information-rich feature matrix. Compared to using full-length fingerprints, this selection strategy significantly reduces model complexity and mitigates overfitting while preserving the most informative structural patterns for gas permeability prediction.

Data Splitting and Normalization

Both input features and target values are standardized prior to training. The full dataset is randomly split into training and test sets in an 80:20 ratio, and model performance is independently evaluated on the test set.

Evaluation Metrics and Formulas

To comprehensively evaluate the model, we adopt the following four standard regression metrics:

Coefficient of Determination (R^2): measures the proportion of variance in the observed data that is explained by the model. It indicates the goodness-of-fit of the regression model, with a value of 1 representing perfect prediction and 0 indicating no predictive power beyond the mean.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Where: y_i:$the\ true\ value$ $\hat{y}_i$:$the\ pricted\ value$

$\bar{y}$:$the\ mean\ of\ all\ true\ values$ n:$the\ number\ of\ samples$

Mean Absolute Error (MAE): is the average of the absolute differences between the predicted values and the true values. It provides a linear score that penalizes all errors equally.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Mean Squared Error (MSE): measures the average of the squared differences between predicted and actual values. It penalizes larger errors more severely than MAE and is sensitive to outliers.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE): RMSE is the square root of MSE, which brings the error units back to the original scale of the output. It is a commonly used metric to assess the average magnitude of prediction errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}$$

Table S3. Evaluation metrics of RF model for gas permeability prediction.

Metric	**Dataset**	**Target1 (N2)**	**Target2(CO2)**
R^2	Train	0.8992	0.8936
R^2	Test	0.7409	0.7535

MAE	Train	0.3559	0.3358
MAE	Test	0.7044	0.6245
MSE	Train	0.3321	0.3095
MSE	Test	0.9761	0.7784
RMSE	Train	0.5762	0.5563
RMSE	Test	0.9880	0.8823

2.3 Generative Model Pretraining

2.3.1 GPT2

(a) Model Architecture

The generative model is based on the classical GPT-2 autoregressive language modeling architecture, with modifications tailored to the structural characteristics of polymer representations in the P-SMILES format. The model adopts a decoder-only Transformer structure consisting of 24 stacked Transformer blocks, each comprising standard subcomponents including multi-head self-attention, feedforward neural networks, residual connections, and layer normalization. Each attention layer contains 16 parallel attention heads, enabling the model to capture long-range dependencies between tokens in the polymer sequences. The dimensionality of the hidden layers in the feedforward subnetwork is set to 128, serving as the embedding space for individual tokens. Compared to the original GPT-2 design, we reduce the embedding size and model depth to better match the relatively lower complexity of polymer datasets and to improve training stability and efficiency.

The model is pretrained in a self-supervised manner on a curated corpus of 100,000 unique P-SMILES sequences. Each sequence is tokenized using a customized SMILES tokenizer and either padded or truncated to a fixed length of 128 tokens. The token sequence is first processed through an embedding layer and a positional encoding module, followed by the main Transformer body. The final hidden states are projected to the vocabulary space through a linear language modeling head to obtain the probability distribution for each token in an autoregressive setting. During training, the model maximizes the likelihood of each token conditioned on its preceding context, and padding tokens are explicitly excluded from the loss computation. The architecture is implemented based on HuggingFace's GPT2Model and is encapsulated using the PyTorch Lightning framework to support scalable fine-tuning and seamless integration into downstream reinforcement learning pipelines.

Table S4. Key hyperparameter settings for GPT-2 model pretraining.

Parameter	**Value**
Max sequence length	128 tokens
Number of layers	24
Number of attention heads	16
Hidden dimension	128
Vocabulary size	63
Tokenizer	SMILESTokenizerEnamine
Loss function	CrossEntropyLoss
Optimizer	AdamW
Learning rate	5×10^{-5}
Batch size	64
Max training epochs	50
Gradient clipping	1.0
Early stopping	Patience=3

(b) Training Process and Loss Evolution

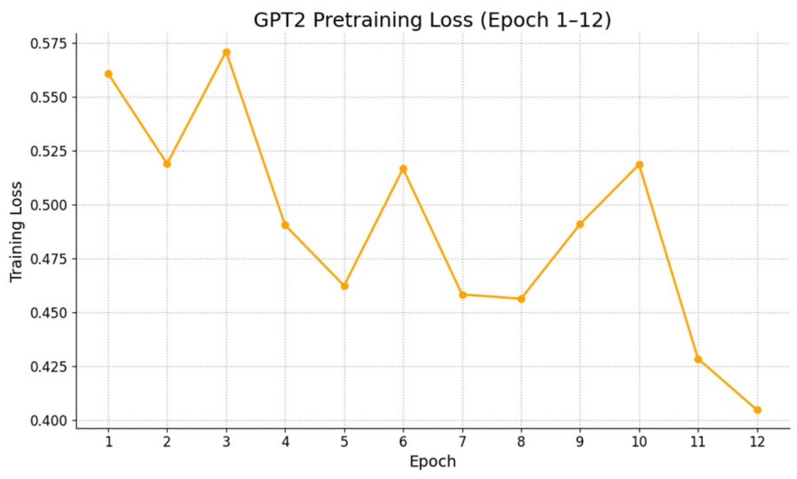

Figure S4. Training loss curve of GPT-2 model during pretraining

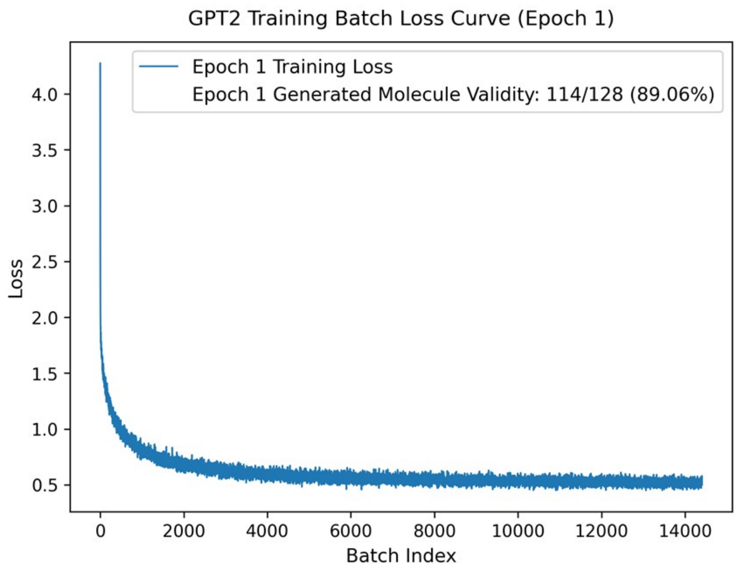

Figure S5. Training batch loss curve of GPT-2 model during the first epoch

Figure S4 illustrates the loss evolution during the training process, which shows an overall decreasing trend, indicating that the model achieves stable convergence. The initial training loss starts at approximately 0.56 and drops to around 0.40 by epoch 12, reflecting a gradual improvement in the model's ability to predict tokens within polymer sequences. Although slight fluctuations are observed at epochs 3 and 6, the loss steadily stabilizes in the later stages. The model reaches its lowest loss values at epochs 11 and 12, suggesting that it has effectively learned the structural patterns embedded in the training corpus. The entire pretraining process takes approximately 240 minutes to complete.

In addition, Figure S5 presents the per-batch loss of the GPT-2 model. It can be observed that the model converges rapidly within the first epoch, with subsequent epochs showing a further overall decrease in loss. However, this continued reduction has a slightly impact on the validity or chemical plausibility of the generated polymer structures, indicating that the generative model does not suffer from underfitting.

(c) Validity and Examples of Generated Molecules

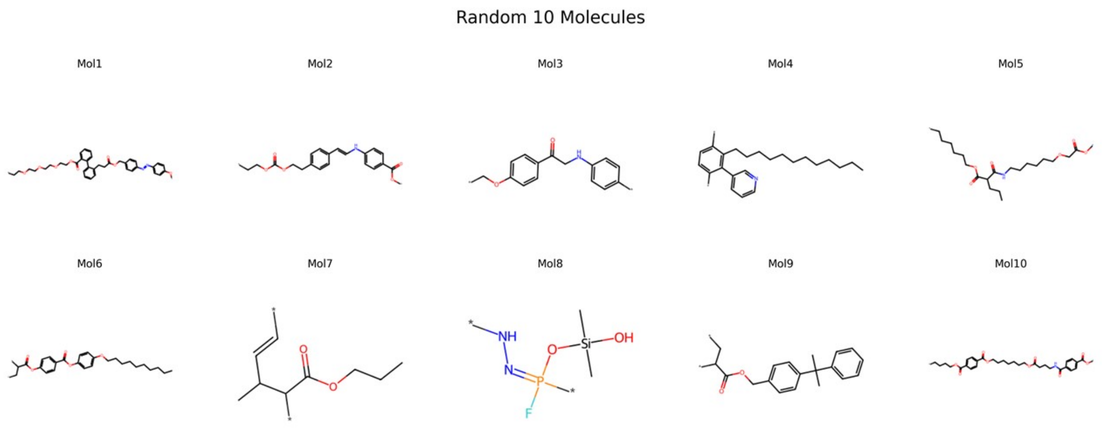

Figure S6. Representative Molecular Structures Generated by GPT-2

The pretrained GPT-2 model randomly generates 128 polymer molecules, of which 121 are valid, resulting in a validity rate of 94.53%.

2.3.2 LLaMA2

(a) Model Architecture

LLaMA2 is a decoder-only Transformer model originally developed for large-scale language modeling tasks. In this work, we adapt the architecture into a lightweight configuration consisting of 4 Transformer decoder layers. Each layer includes multi-head self-attention, a feedforward neural network (FFN), residual connections, and layer normalization. The number of attention heads is set to 16, and both the embedding dimension and hidden state size are configured to 320, enabling each token to be represented in a 320-dimensional semantic space. Compared to the original LLaMA2 design, we reduce the model depth and parameter count to better suit the lower structural complexity of polymer sequence data and to improve training efficiency and stability.

The input to the model is a polymer molecule represented as a P-SMILES string, which is tokenized using a custom SMILES tokenizer and padded or truncated to a fixed length of 128 tokens. The token sequence is first processed through an embedding layer and positional encoding module, and then passed through the LLaMA2 backbone for feature extraction. The final hidden states are projected onto the vocabulary space through a linear output layer to produce token-wise probability distributions. Training is performed in an autoregressive manner by maximizing the conditional likelihood of each token given its preceding context, with padding tokens explicitly excluded from the loss calculation. The architecture is implemented using the HuggingFace LlamaModel and wrapped with PyTorch Lightning to support scalable fine-tuning and integration into downstream reinforcement learning workflows.

Table S5. Key hyperparameter settings for LLaMA2 model pretraining.

Parameter	**Value**
Max sequence length	128 tokens
Number of layers	4
Number of attention heads	16
Hidden dimension	320
Vocabulary size	63
Tokenizer	SMILESTokenizerEnamine
Loss function	CrossEntropyLoss
Optimizer	AdamW
Learning rate	5×10^{-5}
Batch size	64
Max training epochs	50
Gradient clipping	1.0

Early stopping	Patience=3

(b) Training Process and Loss Evolution

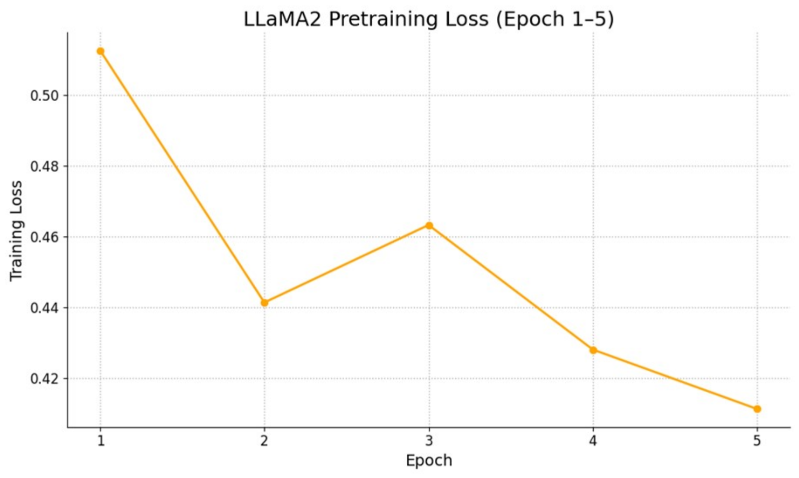

Figure S7. Training loss curve of LLaMA-2 model during pretraining

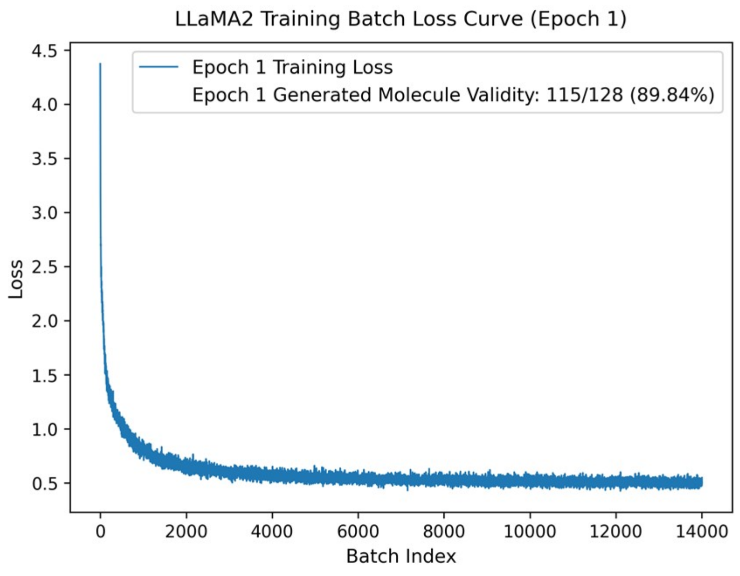

Figure S8. Training batch loss curve of LLaMA-2 model during the first epoch

Figure S7 shows the evolution of training loss during the first five epochs of LLaMA2 pretraining. The model exhibits a consistent downward trend in loss, starting at approximately 0.53 in epoch 1 and steadily decreasing to around 0.41 by epoch 5. This decline reflects the model's progressive improvement in predicting token sequences. A minor increase in loss is observed at epoch 3, likely due to learning instability or parameter adjustments, but the overall trajectory suggests effective convergence.

In addition, Figure S8 presents the per-batch loss of the LLaMA-2 model. It can be observed that the model converges rapidly within the first epoch, with subsequent epochs showing a further overall decrease in loss. However, this continued reduction has a slightly impact on the validity of the generated polymer structures, which is consistent with the behavior observed in GPT-2, indicating that the generative model does not suffer from underfitting.

(c) Validity and Examples of Generated Molecules

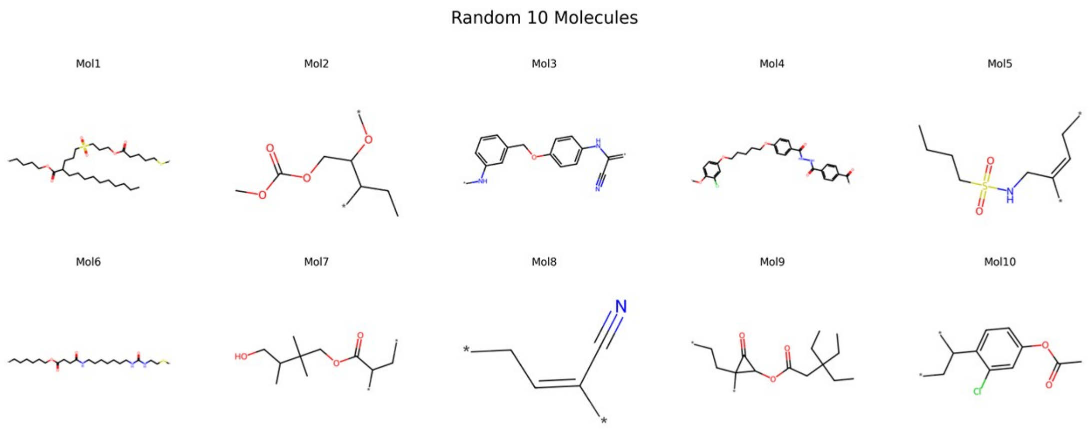

Figure S9. Representative Molecular Structures Generated by LLaMA-2

The pretrained LLaMA-2 model randomly generates 128 polymer molecules, of which 128 are valid, resulting in a validity rate of 100%.

2.3.3 GRU

(a) Model Architecture

The GRU-based generative model consists of three main components: an embedding layer, a stacked GRU module, and an output prediction head. Each token in the input sequence, encoded by the SMILES tokenizer, is first mapped to a 256-dimensional vector through the embedding layer. The resulting sequence is then fed into a three-layer GRU network with a hidden state size of 512. Optional dropout and layer normalization mechanisms are supported to enhance generalization performance.

During the self-supervised pretraining phase, the GRU operates in a time-unfolded mode, recursively updating hidden states at each step to model sequential dependencies. At every time step, the extracted hidden state is passed through a fully connected MLP head to produce a probability distribution over the vocabulary, predicting the next token in an autoregressive manner. The model is trained by maximizing the likelihood of each token conditioned on its preceding context, using cross-entropy loss as the objective function. Padding tokens (<pad>) are explicitly excluded from the loss calculation.

This GRU-based architecture is implemented within the PolyRL framework, which is built on PyTorch and TorchRL. It is designed for modular extensibility and is compatible with reinforcement learning policy interfaces for subsequent fine-tuning and task-specific optimization.

Table S6. Key hyperparameter settings for GRU model pretraining

Parameter	**Value**
Vocabulary size	63
Tokenizer	SMILESTokenizerEnamine
Sequence length	128 tokens
Embedding dimension	256
GRU hidden size	512
GRU layers	3
Dropout	0.0
Layer normalization	Disabled
Optimizer	Adam
Learning rate	0.001
LR scheduler	StepLR (step=500, γ=0.97)
Batch size	64
Epochs	50

(b) Validity and Examples of Generated Molecules

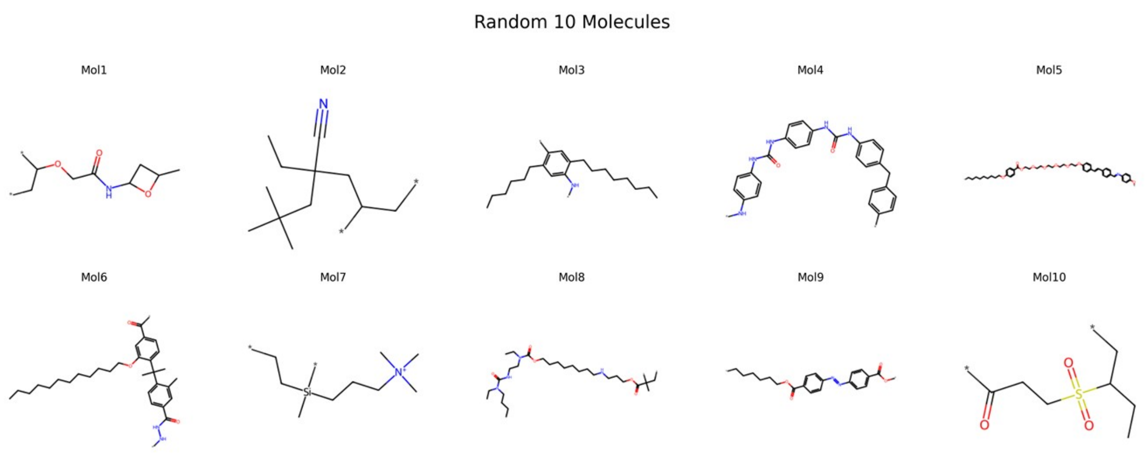

Figure S10. Representative Molecular Structures Generated by GRU

The pretrained GRU model randomly generates 128 polymer molecules, of which 128 are valid, resulting in a validity rate of 100%.

2.3.4 LSTM

(a) Model Architecture

The generative model is built upon a stacked LSTM (Long Short-Term Memorsy) architecture, consisting of three primary components: an embedding layer, a multi-layer LSTM network, and an output projection head. The input polymer structure sequences are first tokenized using a customized SMILES tokenizer. Each token is then mapped to a 256-dimensional continuous vector through the embedding layer to capture fundamental semantic representations. The resulting embedded sequence is passed into a three-layer stacked LSTM module with a hidden state size of 512, enabling the model to capture temporal dependencies across sequence steps and enhancing its capacity to model long-range interactions. Optional dropout and layer normalization mechanisms are supported to mitigate overfitting and improve generalization.

During self-supervised pretraining, the LSTM operates in an unfolded time-step mode, recursively updating hidden states and memory cells to predict each token conditioned on its preceding context. At each time step, the hidden state output is passed through a fully connected MLP head, which projects it onto the vocabulary space to generate a probability distribution over the next token. The model is trained in an autoregressive fashion by maximizing the likelihood of each token given its prior context. Cross-entropy loss is used as the training objective, with padding tokens (<pad>) explicitly excluded from the loss computation.

This LSTM model is implemented within the PolyRL framework, built upon PyTorch and TorchRL. It is designed for high modularity and extensibility, and can be seamlessly integrated into reinforcement learning pipelines as a policy network for further optimization in downstream tasks.

Table S7. Key hyperparameter settings for LSTM model pretraining

Parameter	Value
Vocabulary size	63
Tokenizer	SMILESTokenizerEnamine
Sequence length	128 tokens
Embedding dimension	256
LSTM hidden size	512
LSTM layers	3
Dropout	0.0
Layer normalization	Disabled
Optimizer	Adam
Learning rate	0.001
LR scheduler	StepLR (step=500, γ=0.97)
Batch size	64
Epochs	50

(b) Validity and Examples of Generated Molecules

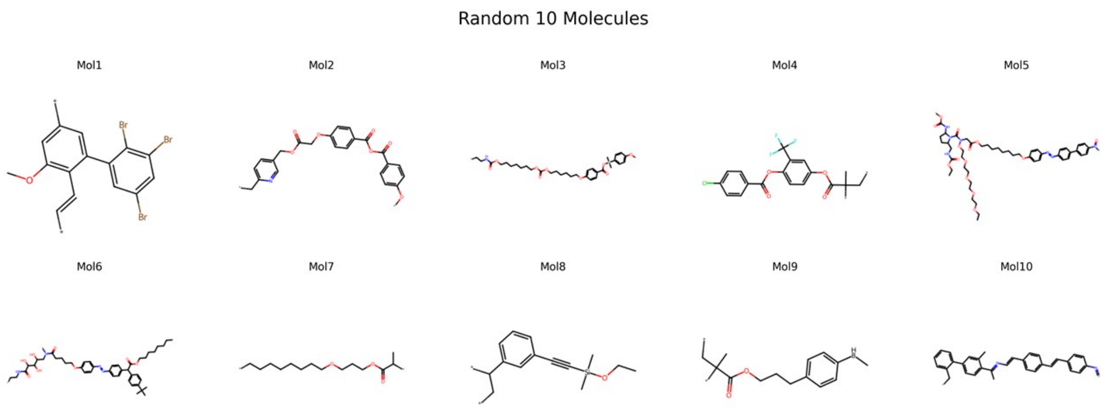

Figure S11. Representative Molecular Structures Generated by LSTM

The pretrained LSTM model randomly generates 128 polymer molecules, of which 126 are valid, resulting in a validity rate of 98.44%.

2.4 Reinforcement Learning Framework

2.4.1 Reward Function Construction

To effectively guide the generative model in producing high-performance polymer structures during reinforcement learning, we designed a reward function based on a multi-objective prediction model. This reward function evaluates the trade-off between CO_2 permeability and CO_2/N_2 selectivity of candidate molecules. The core idea is to encourage the generation of structures that surpass the Robeson upper bound on the permeability–selectivity landscape.

The Robeson upper bound is a widely accepted empirical boundary in the field of polymer membrane separation, characterizing the trade-off relationship between permeability (P) and selectivity (S). In most cases, the influence of molecular structure on these two properties is inversely correlated—higher permeability often comes at the cost of lower selectivity, and vice versa. The Robeson upper bound captures this intrinsic trade-off in logarithmic scale and can be formulated as follows:

$$log_{10}\left(S_{CO_2/N_2}\right) = a - b * log_{10}\left(P_{CO_2}\right)$$

Where $a = 2.595$, $b = 0.3464$

To construct a reward function suitable for policy learning, we reformulated the above equation and defined the extent to which a candidate molecule surpasses the Robeson upper bound as the reward value:

$$Score = log_{10}\left(S_{CO_2/N_2}\right) - \left(a - b * log_{10}\left(P_{CO_2}\right)\right) + \delta$$

Where $\delta = 2$ is an empirical offset introduced to prevent the reward from becoming negative or too small, which would hinder gradient propagation in policy optimization.

For invalid SMILES strings that cannot be parsed, a reward of 0 is uniformly assigned to ensure the stability and differentiability of the reward function.

Through this reward design, the reinforcement learning model is able to learn structural patterns that maximize gas separation performance during molecular generation, enabling performance-driven exploration of the polymer design space.

2.4.2 Reinforcement Learning Algorithms[6]

(a) REINFORCE

REINFORCE (REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility) is a classical policy gradient method. Its core idea is to directly optimize the probability distribution of high-performance molecular generation by leveraging the reward signals obtained from sampled trajectories.In each training iteration, the agent generates a batch of candidate molecular structures based on its current policy, and their performance is evaluated by a reward function R(x). The policy is then updated using the following loss function:

$$L_{reinforce} =- log\pi_{Agent}(x) * R(x)$$

$$L_{reinforce} =- \sum_{t=1}^{T} log\pi_{\theta}\left(a_t \middle| s_t\right) * \ R(x)$$

Where:

$\pi_{Agent}(x)$: probability of generating molecule x under the current agent policy.

$R(x)$: the performance score of molecule x computed by the reward function;

$Action\ a_t$: the token selected by the generative model at time step t.

$State\ s_t$: the partial SMILES sequence generated up to time step t.

This loss function aims to increase the likelihood of generating high-reward molecules, thereby achieving task-directed generative optimization. The gradient is estimated following the standard REINFORCE update rule:

$$\nabla_\theta E_{x\sim\pi_\theta}[R(x)] \approx \nabla_\theta log\pi_\theta(x) * R(x)$$

To improve training stability and sample efficiency, we incorporate batch-wise reward smoothing and optionally enable experience replay, which helps mitigate gradient variance caused by noisy reward signals.

Table S8. Key hyperparameter settings for REINFORCE reinforcement learning

Parameter	Value
Random seed	101
Num environments	128
Total SMILES	20000
Learning rate	0.0001
Epsilon	1e-08
Weight decay	0.0
Experience replay	True
Replay buffer size	100
Replay batch size	10

(b) REINVENT

REINVENT (REINforcement-learned Virtual screening of molecular ENTities) is a reinforcement learning method based on the policy gradient framework. Its core idea is to use an unsupervised pretrained language model as a prior policy for molecular structure generation and to fine-tune the agent policy toward target properties using reward signals. This approach allows the model to retain the structural validity encoded in the prior distribution while optimizing for desired performance characteristics.Rather than directly maximizing the molecular reward, REINVENT constructs a reward-guided log-likelihood objective to update the agent's policy. Specifically, in each training iteration, the agent generates a batch of candidate molecules, which are scored by a reward function R(x). The agent is then updated using the following loss function:

$$L_{reinvent} = (\sigma * R(x) + log\pi_{Prior}(x) - log\pi_{Agent}(x))$$

Where:

$\pi_{Prior}(x)$: probability of generating molecule x under the prior model.

$\pi_{Agent}(x)$: probability of generating molecule x under the current agent model.

$R(x)$: the performance score of molecule computed by the reward function.

σ: a hyperparameter that controls the influence strength of the reward signal during training.

The essence of this loss function is to encourage the agent to generate molecules that not only receive high rewards but also lie within high-probability regions of the prior distribution. In this way, goal-directed optimization is embedded within the structure-aware generative framework, achieving a balance between performance and synthetic feasibility. Compared to directly maximizing the reward or strengthening the generator's distribution, REINVENT's reward-augmented log-likelihood minimization strategy helps mitigate issues such as policy degradation and mode collapse, leading to improved training stability and generalization.

Table S9. Key hyperparameter settings for REINVENT reinforcement learning

Parameter	Value
Random seed	101
Num environments	128
Total SMILES	20000
Learning rate	0.0001
Epsilon	1e-08

Weight decay	0.0
Sigma	120
Experience replay	True
Replay buffer size	100
Replay batch size	10

(c) AHC

AHC (Augmented Hill-Climb) is a molecular reinforcement learning method that combines reward-augmented likelihood optimization with a top-k hill-climbing strategy. It builds upon the REINVENT framework by using a pretrained generative model as the prior policy and guiding the agent policy toward desired molecular properties through a reward function. Unlike REINVENT, which updates the policy using all generated samples, AHC selects only the top k% highest-scoring molecules in each training batch for policy updates. This enhances learning from high-quality samples and reduces gradient noise introduced by low-quality candidates.

Specifically, the agent generates a batch of candidate molecules, each evaluated by a reward function R(x). The molecules are then ranked by reward, and only the top-k samples are used to compute the loss. The loss function is formulated as:

$$L_{AHC} = (\sigma * R(x) + log\pi_{Prior}(x) - log\pi_{Agent}(x))$$

Where:

$\pi_{Prior}(x)$: probability of generating molecule x under the prior model.

$\pi_{Agent}(x)$: probability of generating molecule x under the current agent model.

$R(x)$: the performance score of molecule computed by the reward function.

σ: a hyperparameter that controls the influence strength of the reward signal during training.

The objective is to minimize the distance between the agent policy and a reward-weighted prior distribution. By introducing the top-k filtering mechanism, AHC significantly increases the impact of high-performance samples during training and prevents the agent from drifting toward low-quality regions of chemical space. In addition, AHC includes a regularization term to penalize overly high sequence likelihoods, helping to avoid mode collapse and improve training stability. It also supports experience replay, allowing the model to revisit high-reward molecules for enhanced learning. Benefiting from these designs, AHC demonstrates faster convergence and stronger reward guidance in polymer optimization tasks in this study.

Table S10. Key hyperparameter settings for AHC reinforcement learning

Parameter	Value
Random seed	101
Num environments	128
Total SMILES	20000
Learning rate	0.0001
Epsilon	1e-08
Weight decay	0.0
Top-k fraction	0.5
Sigma	60
Experience replay	True
Replay buffer size	100
Replay batch size	10

(d) A2C

A2C (Advantage Actor-Critic) is a reinforcement learning algorithm that integrates a policy network (actor) and a value network (critic), enabling stable and efficient optimization by reducing the variance of policy gradient estimates through the use of advantage functions.

In each training iteration, the agent generates a batch of candidate molecules using the current policy network, and each molecule is scored using a task-specific reward function R(x). The generalized advantage estimation (GAE) method is then applied to compute the advantage value A(x) for each molecule trajectory, representing how much better the actual return is compared to the expected baseline.

The A2C loss fucntion consists of four components:

$$L_{A2C} = L_{actor} + \beta * L_{critic} - \alpha * L_{entropy} + \lambda * D_{KL}(\pi_{agent}||\pi_{prior})$$

Where:

1) $L_{actor} =- log\pi_{agent}(x) * A(x)$ is the actor loss that promotes actions with higher advantages
2) $L_{critic} = (V(x) - R(x))^2$ is the critic loss that minimizes the mean squared error between the predicted value V(x) and actual return R(x)
3) $L_{entropy} =- H[\pi_{agent}(x)]$is the entropy regularization term that encourages exploration by maximizing the policy's uncertainty..
4) $D_{KL}(\pi_{agent}||\pi_{prior})$ is the KL divergence between the current agent policy and the pretrained prior policy, used to maintain chemical validity

x: a complete generated molecule (a SMILES string).

$R(x)$: reward of molecule x computed from the scoring function.

$A(x)$: the advantage function, compute using GAE.

$V(x)$: value estimate of the generated molecule given by the critic network.

$\pi_{Prior}(x)$: probability of generating molecule x under the prior model.

$\pi_{Agent}(x)$: probability of generating molecule x under the current agent model.

$H[\cdot]$: Shannon entropy of the probability distribution.

α,β,λ: hyperparameters controlling the contribution of entropy, critic loss, and KL regularization, respectively.

Owing to the coordinated optimization between the actor and critic networks, A2C effectively reduces variance in policy updates while preserving structural validity. The introduction of advantage-based learning enhances the directionality of training signals, and the combined use of entropy regularization and KL constraints encourages sufficient exploration and chemical diversity.

Table S11. Key hyperparameter settings for A2C reinforcement learning

Parameter	Value
Random seed	101
Num environments	128
Total SMILES	20000
Shared actor-critic networks	False
Learning rate	0.0001
Epsilon	1e-06
Weight decay	0.0
Gamma	0.999
GAE lambda	0.99
Critic loss coefficient	0.5
Entropy loss coefficient	0.1
KL divergence coefficient	0.001
Mini-batch size	16
Max gradient norm	0.5

(e) PPO

PPO (Proximal Policy Optimization) is a policy gradient algorithm based on the trust region concept. It aims to improve policy optimization efficiency while restricting the magnitude of updates, thereby preventing policy collapse or instability. In this study, PPO is employed to optimize molecular generation strategies toward target properties, using clipped surrogate objectives and multi-epoch updates to enhance convergence stability in complex molecular spaces.

Similar to A2C, PPO adopts an actor-critic architecture in which the policy network (actor) and value network (critic) are jointly trained. In each training iteration, the actor generates a batch of candidate molecules, which are scored using a reward function R(x). The Generalized Advantage Estimation method is used to compute the advantage function A(x), reflecting the relative value of each trajectory.

The core actor loss in PPO is formulated as a clipped surrogate objective:

$$L_{actor} = -E_x[\min(r(x) * A(x), clip(r(x), 1-\epsilon, 1+\epsilon) * A(x))]$$

Where:

1) $r(x) = \frac{\pi_{agent}(x)}{\pi_{old}(x)}$ the probability ratio between the current and previous policies
2) $A(x)$: the advantage function computed via GAE
3) ϵ: the clipping threshold, used to limit excessive policy updates.

This clipped objective is designed to restrict the magnitude of policy updates. If the probability ratio r(x) deviates too far from 1, the gradient is clipped to ensure conservative updates. Specifically, the minimum operator ensures that the loss does not increase when the update would excessively improve the advantage, effectively bounding the policy improvement within a safe range.

In addition to the surrogate loss, the total PPO objective includes three regularization terms:

$$L_{PPO} = L_{actor} + \beta * L_{critic} - \alpha * L_{entropy} + \lambda * D_{KL}(\pi_{agent}||\pi_{prior})$$

Where:

1) $L_{critic} = (V(x) - R(x))^2$ minimizes the squared error between predicted and actual returns.
2) $L_{entropy} = -H[\pi_{agent}(x)]$ is the entropy regularization term that encourages exploration by maximizing the policy's uncertainty
3) $D_{KL}(\pi_{agent}||\pi_{prior})$ is the KL divergence between the current agent policy and the pretrained prior policy, used to maintain chemical validity

To improve sample efficiency, PPO applies multiple optimization epochs over each batch and optionally incorporates prioritized experience replay to expand the training dataset. The gradients are clipped to prevent numerical instability caused by overly large updates.

Table S12. Key hyperparameter settings for PPO reinforcement learning

Parameter	Value
Random seed	101
Num environments	64
Total SMILES	20000
Shared actor-critic networks	False
Learning rate	0.0005
Epsilon	1e-06
Weight decay	1.0e-06
Discount factor (gamma)	0.999
GAE lambda	1.0
Critic loss coefficient	0.25
Entropy loss coefficient	0.1
KL divergence coefficient	0.001
PPO clipping threshold	0.3
PPO epochs per update	3
Max gradient norm	0.25
Experience replay	False
Replay batch size	24
Replay buffer size	100

(f) DPO

Direct Preference Optimization (DPO) is a preference-based reinforcement learning method that aims to optimize the policy without relying on explicit scalar reward values. Instead, it guides the policy to favor higher-quality samples based on preference comparisons. In this study, DPO is employed to steer the molecular generation policy toward high-performing structures through relative ranking.

During each training iteration, the policy generates a batch of candidate molecules, which are evaluated by the task-specific reward function R(x). The molecules are then ranked based on their reward scores, with the top 50% selected as preferred samples (x^+) and the bottom 50% as dispreferred samples (x^-). These preference pairs are used to compute the DPO loss:

$$L_{DPO} =- log\sigma\ (\beta * [log\frac{\pi_{agent}(x^+)}{\pi_{prior}(x^+)} - log\frac{\pi_{agent}(x^-)}{\pi_{prior}(x^-)}]$$

Where:

1) x^+,x^-: molecules with higher and lower reward scores, respectively.
2) $\pi_{Prior}(x)$: probability of generating molecule x under the pretrained prior model.
3) $\pi_{Agent}(x)$: probability of generating molecule x under the current agent model.
4) β: a temperature scaling parameter controlling the strength of the preference signal
5) $\sigma(\cdot)$: the sigmoid function, mapping preference margins to likelihoods

This objective minimizes the probability that the agent assigns higher relative preference to lower-reward molecules, thus aligning the agent's trajectory distribution with the ranking induced by the reward function. Unlike value-based or advantage-weighted approaches, DPO directly optimizes pairwise preferences without needing scalar advantage estimation, making it more robust in noisy or sparse reward settings. To ensure the chemical validity of generated molecules, DPO includes a prior-policy constraint that regularizes the policy's deviation from the pretrained distribution. This prevents the agent from drifting too far from the chemically meaningful space.

Table S13. Key hyperparameter settings for DPO reinforcement learning

Parameter	Value
Random seed	101

Num environments	128
Total SMILES	20000
Model type	gpt2
Learning rate	0.0001
Epsilon	1e-08
Weight decay	0.0
Beta (preference scaling factor)	1
Number of mini-batches	1
Number of training epochs	1

2.4.3 Reinforcement Learning Performance Comparison

To comprehensively evaluate the performance of different reinforcement learning strategies in molecular generation, we adopt the following metrics based on 20,000 molecules generated per task：

1) Max Score: The highest reward value among all generated molecules, representing the upper bound of model performance for the task-specific objective.
2) Top-100 Mean: The average reward score of the top 100 molecules, measuring how concentrated the generated samples are in the high-performing region.
3) Top-100 SAscore: The mean Synthetic Accessibility (SA) score of the top 100 molecules. A higher SA score generally reflects lower synthetic accessibility, indicating structures that may be more difficult to synthesize.[8]
4) Validity Rate: The proportion of generated SMILES that can be successfully parsed into valid molecular graphs using RDKit. This metric reflects the structural correctness of the generated molecules.
5) Unique Count (molecule): The proportion of structurally unique molecules among all valid molecules, indicating molecular-level diversity.
6) Unique Count (scaffold): The proportion of distinct Bemis–Murcko scaffolds

among valid molecules, reflecting diversity at the scaffold (core structure) level.

7) Novelty Count (molecule): The proportion of valid molecules that are not present in the pretraining dataset, measuring novelty at the molecular level.

8) Novelty Count (scaffold): The proportion of scaffolds in valid molecules that are absent from the pretraining dataset, assessing the model's ability to explore novel chemotypes.

Table S14. Detailed Metric Values for RF + GPT-2 under Different RL Algorithms

Metric	**REINVENT**	**REINFORCE**	**DPO**	**PPO**	**A2C**	**AHC**
Max Score	2.026	2.032	1.715	1.614	1.717	2.02
Top100 Mean	2.024	2.019	1.649	1.568	1.506	2.02
Top100 SAscore	6.1678	6.7343	5.6194	3.6469	5.1681	7.0601
Validity Rate	0.9477	0.9475	0.9665	0.9308	0.9043	0.9107
Unique Count (molecule)	0.9434	0.9199	0.6118	0.8583	0.9973	0.8288
Unique Count (scaffold)	0.0705	0.048	0.0227	0.002	0.2123	0.036
Novelty Count (molecule)	0.9644	0.9605	0.6185	0.9115	0.8907	0.9837
Novelty Count (scaffold)	0.1622	0.074	0.0049	0.0005	0.091	0.0231

Table S15. Detailed Metric Values for RF + REINVENT under Different Generators

Metric	**GPT-2**	**LLaMA-2**	**GRU**	**LSTM**
Max Score	2.026	2.032	1.961	1.972
Top100 Mean	2.024	2.025	1.771	1.764
Top100 SAscore	6.1678	6.0836	5.5043	5.4053
Validity Rate	0.9477	0.9718	0.9835	0.9853
Unique Count	0.9434	0.8288	0.9038	0.9244

(molecule)				
Unique Count (scaffold)	0.0705	0.0984	0.1014	0.1112
Novelty Count (molecule)	0.9644	0.943	0.9345	0.906
Novelty Count (scaffold)	0.1622	0.1448	0.0409	0.0449

Table S16. Metric Comparison of RF + REINVENT + GPT-2 Trained with Varying Pretraining Dataset Sizes

Metric	**100w**	**75w**	**50w**	**25w**	**10w**
Max Score	2.026	2.032	2.026	1.809	1.657
Top100 Mean	2.024	2.025	2.024	1.758	1.636
Top100 SAscore	6.1678	6.2542	5.9454	7.1173	6.7015
Validity Rate	0.9477	0.9447	0.9464	0.9565	0.9577
Unique Count (molecule)	0.9434	0.9217	0.9563	0.8936	0.8898
Unique Count (scaffold)	0.0705	0.1002	0.0654	0.0142	0.0103
Novelty Count (molecule)	0.9644	0.982	0.9717	0.967	0.9654
Novelty Count (scaffold)	0.16s22	0.1405	0.0938	0.0049	0.0068

3 Results Analysis

3.1 Efficiency and Molecular Visualization

Table S17.Efficiency Comparison of Reinforcement Learning Algorithms

	REINVENT	**REINFORCE**	**DPO**	**PPO**	**A2C**	**AHC**

Time(min)	26	26	29	46	27	26

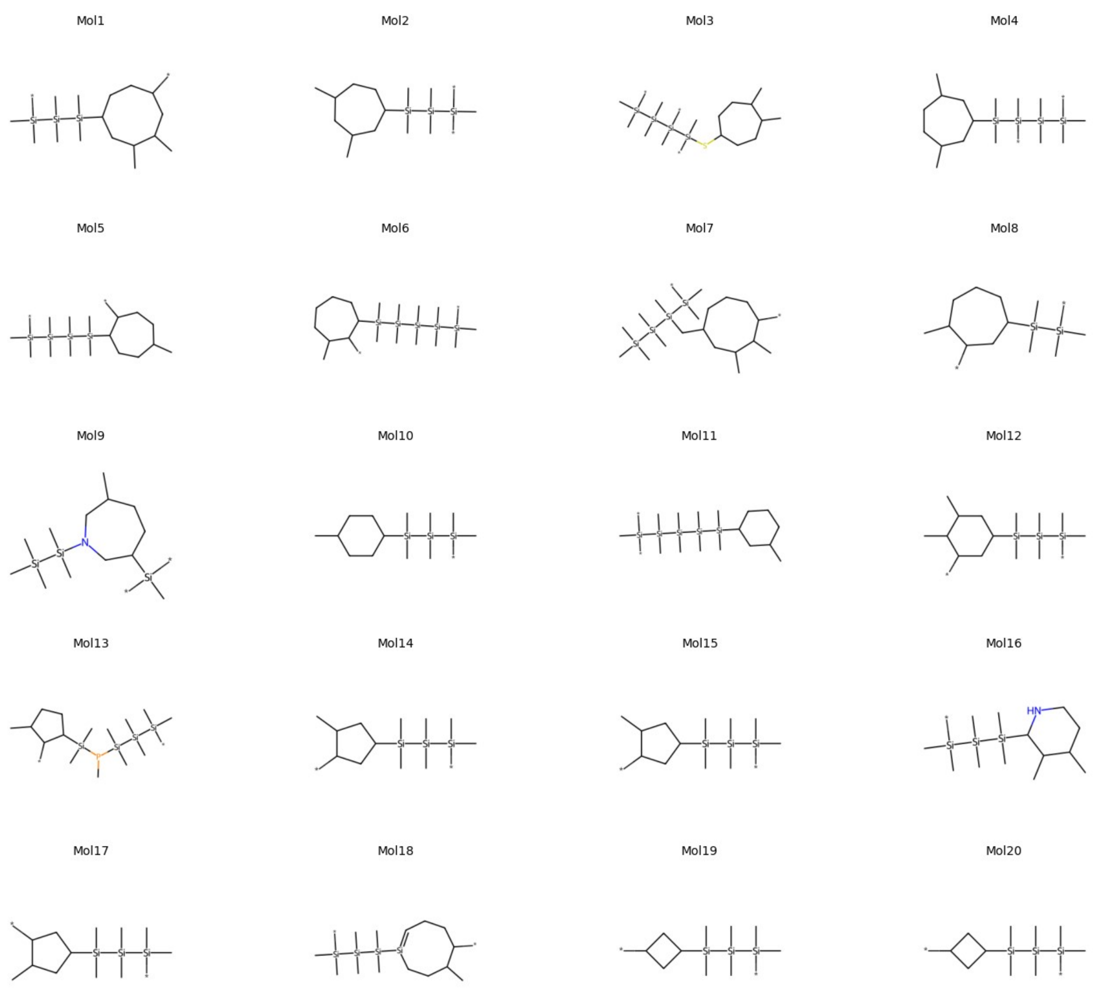

Figure S12. Visualization of Molecules Surpassing the Robeson Upper Bound

3.2 SHAP Analysis

3.2.1 Method

(a) Molecular Feature Selection

To identify structural features that significantly influence model predictions, we employ the Extended Connectivity Fingerprint (ECFP) method to encode all molecules and apply statistical filtering based on the training dataset to select representative substructures.

Specifically, we extract the SMILES representations from the training set and convert them into molecular graphs using RDKit. We then compute Morgan fingerprints with a radius of 3 for each molecule and extract the non-zero bit identifiers (bit IDs) along with their counts. By aggregating all bit IDs across the dataset, we obtain a unique set of substructures and save it as a numbered DataFrame. This file records the mapping between each substructure (bit ID) and its corresponding feature index, serving as a critical reference for reverse-mapping high-contribution features identified during the subsequent SHAP analysis.

Next, we encode each molecule into a fixed-length vector according to the established bit ID order, resulting in a sparse fingerprint matrix. To reduce dimensional redundancy and improve training efficiency, we analyze the proportion of zeros in each bit position and retain only frequently occurring substructures. In this study, we select 114 high-frequency features, forming a compact and informative subset of molecular descriptors used for interpretability analysis.

(b) SHAP

We apply the SHAP (SHapley Additive exPlanations) method to perform feature attribution for the model.[9] Based on the Shapley value theory from cooperative game theory, SHAP quantifies the marginal contribution of each input feature to the model's prediction, thereby enabling interpretability analysis. In this study, we use the feature matrix of high-performance molecules as input samples and employ the TreeExplainer module to interpret a trained random forest model. SHAP values are computed separately for the two prediction targets (CO_2 and N_2). Each structural fragment (bit position) receives a SHAP value vector, indicating its influence distribution across all samples. To assess the overall importance of each feature, we calculate the mean absolute SHAP value for each bit and use it as a quantitative measure of feature contribution. The features are then ranked accordingly to identify the most influential structural fragments.

(c) Structural Visualization

To intuitively illustrate how the top-ranked structural features identified by SHAP analysis influence molecular architecture, we further highlight the specific functional groups or atomic fragments corresponding to these high-contribution features. Before visualization, we first map each feature index back to its original bit_id using the previously saved feature mapping table. Based on this, we select a set of representative molecules from the high-performance subset that contain the target bit_id. Specifically, for each SMILES entry, RDKit is used to generate all active fingerprint bits and identify whether the current molecule includes the target bit_id. If the feature is present, we further extract the corresponding atom indices and visualize the molecular structure using RDKit's MolDraw2D tool, highlighting the relevant atoms.

It is worth noting that the same bit_id in ECFP fingerprints may correspond to different atomic environments across molecules. Therefore, the highlighted regions do not represent a fixed functional group structure, but rather the local fragment patterns that this feature denotes in different molecular contexts.

3.2.2 Result

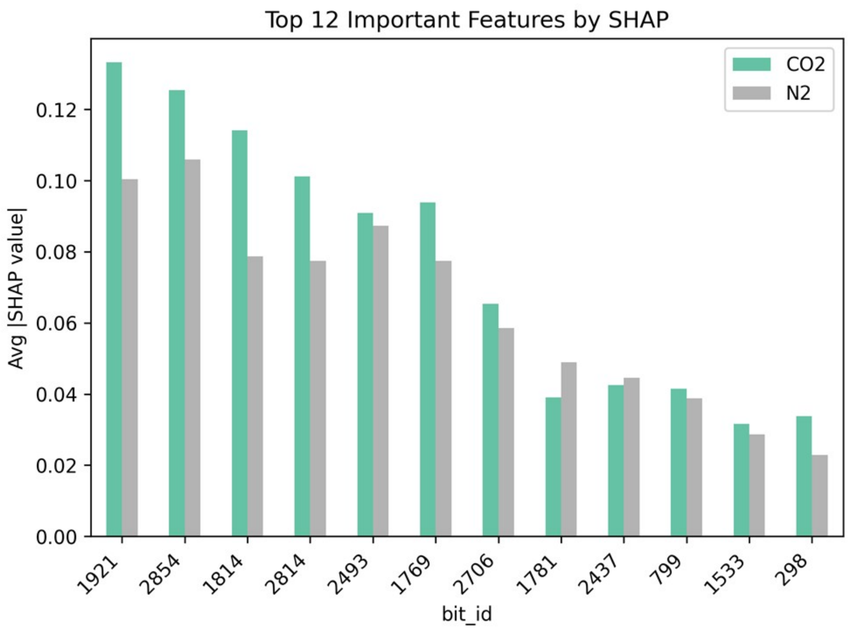

Figure S13. Top 12 Important Molecular Features Identified by SHAP Analysis

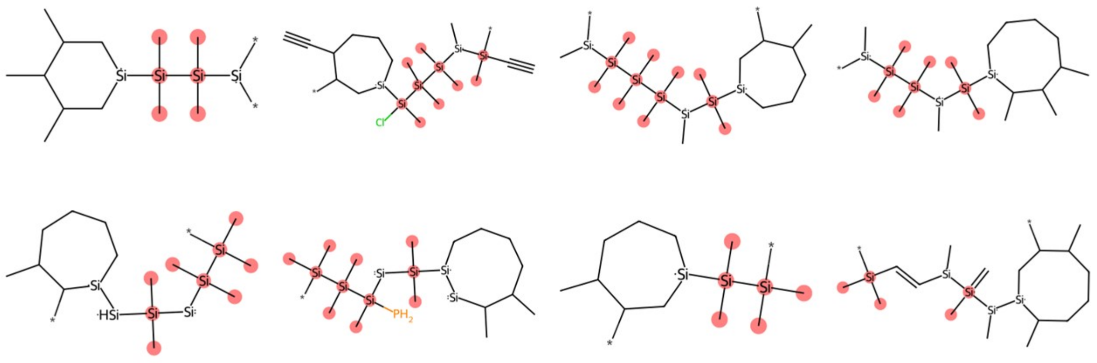

Figure S14. Visualization of the Molecular Substructure Corresponding to Bit ID 1921

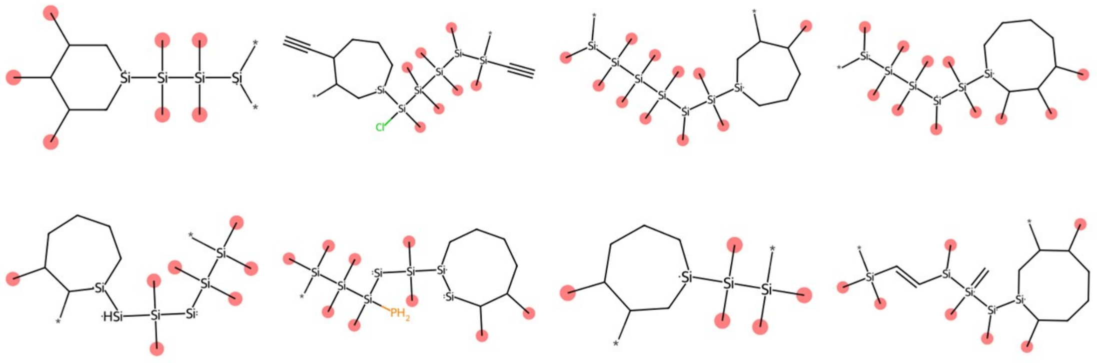

Figure S15. Visualization of the Molecular Substructure Corresponding to Bit ID 2854

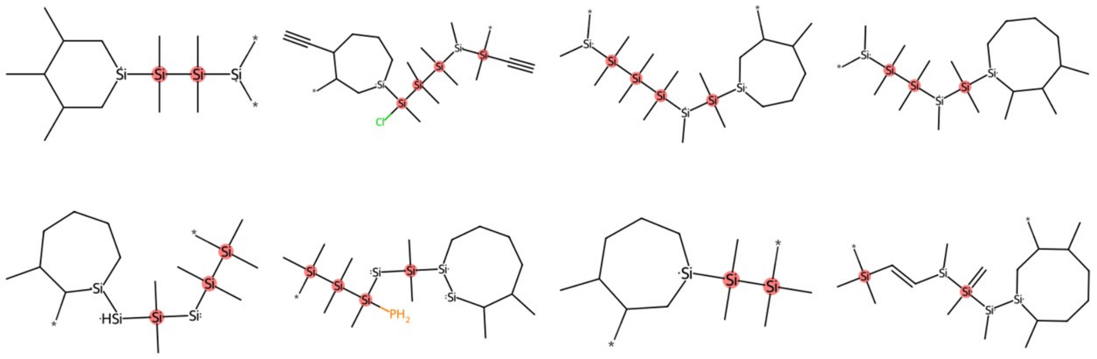

Figure S16. Visualization of the Molecular Substructure Corresponding to Bit ID 1814

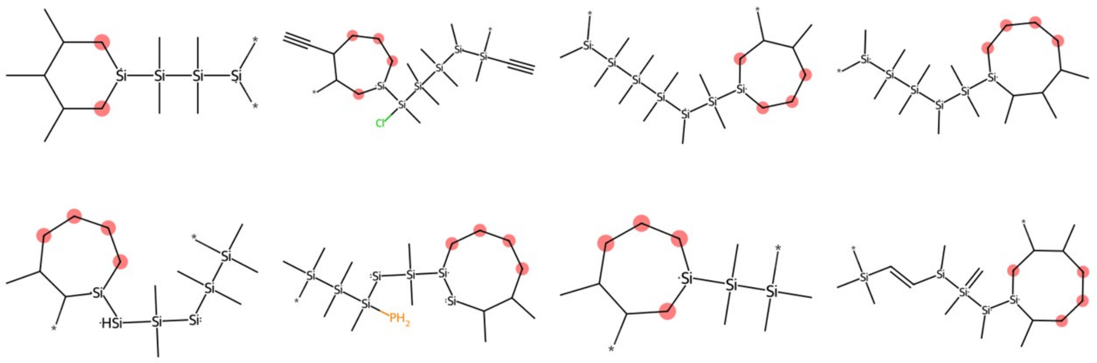

Figure S17. Visualization of the Molecular Substructure Corresponding to Bit ID 2814

3.3 Molecular Dynamics Simulation Setting

To evaluate the gas separation performance of polymer structures recommended by reinforcement learning, this study establishes a systematic molecular dynamics (MD) simulation workflow.[10] All simulations are performed using the GROMACS and LAMMPS platforms. The OPLS-AA all-atom force field is employed to accurately model intermolecular interactions, ensuring the physical reliability of the results.

The SMILES representations of polymer repeating units generated by reinforcement learning are first converted into three-dimensional structures. Single-chain polymers are constructed using RDKit, and file format conversion as well as stereochemical standardization is conducted using Open Babel. The polymer chains are embedded in a cubic simulation box, with periodic boundary conditions applied along the backbone direction to construct infinitely repeating linear polymer structures.

Before production simulations, the system undergoes energy minimization to remove geometric distortions and relieve initial stress. This is followed by a thermal annealing step to allow the polymer chains to relax into stable conformations. Thermal equilibration is conducted in two stages: a 0.5 ns NVT simulation at 500 K to stabilize the temperature, followed by a 0.5 ns NPT (constant-pressure, constant-temperature) simulation at 500 K and 1 bar to equilibrate system density and volume.

To estimate the glass transition temperature (Tg), the system is cooled from 500 K to 250 K under NPT conditions, with the temperature decreased in 50 K intervals. Equilibrium densities are collected at each temperature point, and Tg is determined by identifying the inflection point in the density–temperature curve.

After thermodynamic equilibration of the polymer matrix, target gas molecules (CO_2 and N_2) are inserted under infinite dilution conditions to assess their solubility and diffusivity within the polymer. The system is further equilibrated under NVT conditions at 300 K for 1 ns and under NPT conditions at 1 atm for 2 ns. A 50 ns production simulation is then performed to record the trajectories of the gas molecules.

Based on the trajectory data, the following key performance metrics are calculated:

Diffusion Coefficient (D):

The diffusion coefficient is calculated from the slope of the mean squared displacement (MSD) curve of gas molecules, using the following equation:

$$D = \lim_{t\to\infty} \frac{1}{6t}\langle|\vec{r}(t) - \vec{r}(0)|^2\rangle$$

Where:

$\vec{r}(t)$ is the position vector of a gas molecule at time t

$\langle\cdot\rangle$ denotes the statistical average over all gas molecules

the coefficient 6 arises from diffusion in three-dimensional space (for two dimensions, the coefficient would be 4).

Solubility (S):

Solubility is calculated via Widom insertion to obtain Henry coefficients, cross-checked on a subset with low-loading GCMC.

The solubility coefficient Squantifies the equilibrium concentration of gas molecules dissolved in a polymer matrix under a given pressure and is inversely proportional to the Henry constant k_H:

$$S = \frac{1}{k_H}$$

The Henry coefficient $k_H$is related to the excess chemical potential by

$$k_H = \frac{1}{RT}exp(\frac{\mu^{ex}}{RT})$$

At infinite dilution, k_H is evaluated using the Widom test particle insertion method, which estimates the excess chemical potential μ^{ex} of a single gas molecule in the polymer phase.

In the canonical (NVT) ensemble, μ^{ex} is expressed as:

$$\mu^{ex} = -k_B T \ln \langle exp(-\frac{\Delta U}{k_B T}) \rangle$$

where ΔU is the potential energy change upon inserting a test gas molecule into the polymer matrix, k_B is the Boltzmann constant, and $\langle \cdots \rangle$ denotes ensemble averaging over equilibrium polymer configurations.

Permeability (P):

Permeability is determined as the product of the diffusion coefficient and solubility, expressed as:

$$P = D * S$$

with the unit of Barrer.

Finite-size analysis:

We simulated box sizes (35 nm; constant density) for representative polymers.

3.4 Evaluation of Generated Polymers via Molecular Dynamics Simulation

To mitigate this issue and provide a more balanced validation, we have adopted a score-based stratified sampling approach to examine the consistency between model predictions and MD-calculated results. Specifically, we sampled 12 polymers across three different predicted score ranges (0.5, 1.0, and 1.5; 10 polymers each, where a score of 2.0 corresponds to the Robeson upper bound) and performed MD simulations for each subset.

As illustrated in Figure S18, the MD-calculated results are in reasonable agreement with the model predictions to a certain extent. This alignment indicates that the model exhibits sufficient accuracy to describe the actual material properties. Notably, the MD-calculated scores for the three polymer clusters show a gradual upward trend—with the cluster corresponding to the highest predicted score also achieving the highest MD-calculated value, while the cluster with the lowest predicted score yields the lowest MD-calculated result.

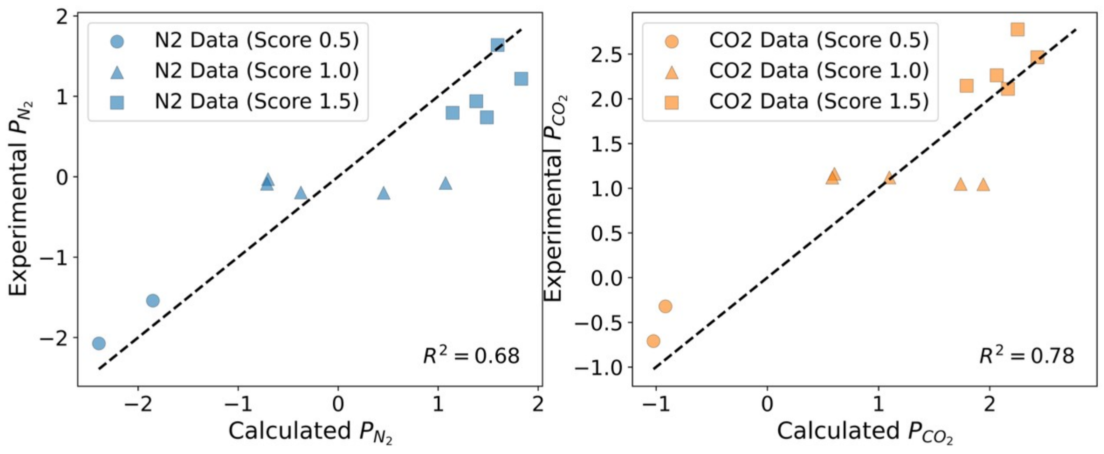

Figure S18. Comparison Between MD-Calculated Results and Model Predictions for Polymers Sampled Across Three Predicted Score Ranges (0.5, 1.0, 1.5)

3.5 Benchmarking the Molecular Dynamics Simulation with Experimental Dataset

To further validate our MD calculations against experimental results, we have performed additional MD simulations for 47 representative polymer systems from the experimental dataset and compared the calculated values with their experimental labels. As shown in Figure S19, the results from the molecular dynamics calculations are in reasonable agreement with the labels of the experimental dataset to a certain extent. This consistency reflects the effectiveness of our molecular dynamics calculation workflow, confirming that the established computational framework can reliably reproduce key properties of the polymer systems.

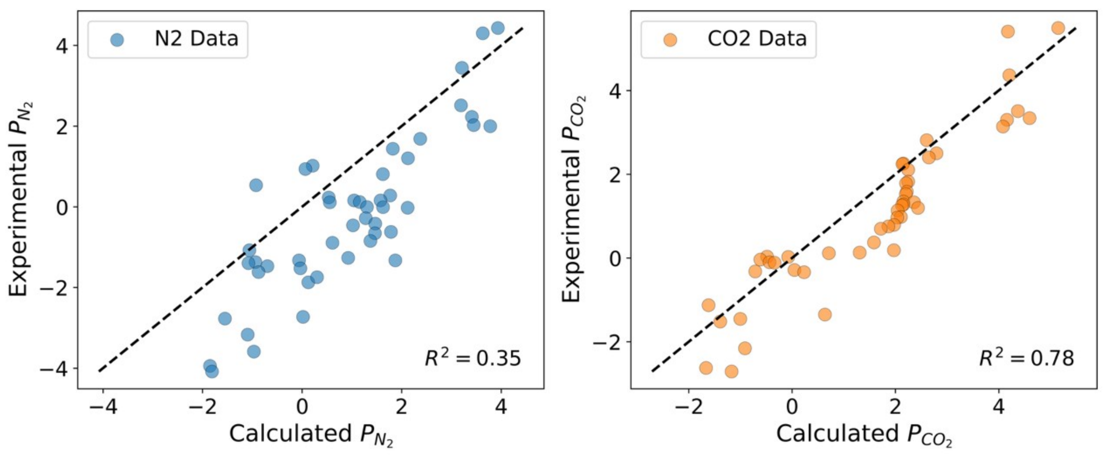

Figure S19. Comparison Between MD-Calculated Values and Experimental Labels for 47 Representative Polymer Systems

Reference

[1] J. Yang, L. Tao, J. He, J. R. McCutcheon and Y. Li, *Science Advances*, **8**, eabn9545.

[2] R. Ma and T. Luo, *Journal of Chemical Information and Modeling*, 2020, **60**, 4684-4690.

[3] V. Varshney, S. S. Patnaik, A. K. Roy and B. L. Farmer, *Macromolecules*, 2008, **41**, 6837-6842.

[4] D. Rogers and M. Hahn, *Journal of Chemical Information and Modeling*, 2010, **50**, 742-754.

[5] L. Tao, V. Varshney and Y. Li, *Journal of Chemical Information and Modeling*, 2021, **61**, 5395-5413.

[6] A. Bou, M. Thomas, S. Dittert, C. Navarro, M. Majewski, Y. Wang, S. Patel, G. Tresadern, M. Ahmad, V. Moens, W. Sherman, S. Sciabola and G. De Fabritiis, *Journal of Chemical Information and Modeling*, 2024, **64**, 5900-5911.

[7] P. Ertl and A. Schuffenhauer, *Journal of Cheminformatics*, 2009, **1**, 8.

[8] C. Rudin, *Nat Mach Intell*, 2019, **1**, 206-215.

[9] R. M. Venable, A. Krämer and R. W. Pastor, *Chem Rev*, 2019, **119**, 5954-5997.