SMI-EDITOR: EDIT-BASED SMILES LANGUAGE MODEL WITH FRAGMENT-LEVEL SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

SMILES, as a crucial textual representation of molecular information, is increasingly drawing interest for its pre-trained language models. However, most existing pre-trained SMILES language models (LMs) only provide supervision at the single-token level during pre-training and fail to fully leverage substructural information of molecules. This limitation results in the pre-training task being overly simplistic and further preventing the models from capturing richer molecular semantic information. Additionally, during pre-training, these SMILES LMs only process corrupted SMILES inputs, never encountering any valid SMILES as input, leading to a train-inference mismatch. To address these challenges, we propose SMI-EDITOR, a novel edit-based pre-trained SMILES language model. SMI-EDITOR randomly disrupts substructures within a molecule and feeds the resulting SMILES back into the model, which then attempts to restore the original SMILES through an editing process. This training method not only introduces a fragment-level training signal but also allows the use of valid SMILES as inputs, enabling the model to learn how to edit these incomplete structures back to complete molecules. This significantly enhances the model's scalability and capability to learn fragment-level molecular information. Experimental results show that the SMI-EDITOR performs well across multiple downstream molecular tasks, achieving state-of-the-art results, and even surpasses several 3D molecular representation models in performance.

1 INTRODUCTION

With the widespread application of AI technology in various molecular-related tasks, enhancing the modeling of SMILES data has emerged as a research focal point. Due to the textual nature of SMILES data, we can conveniently apply experiences from text modeling to address challenges in SMILES modeling, and the knowledge extracted from SMILES data often aligns more easily with textual knowledge. A large number of research has attempted to design SMILES language models to explore the knowledge inherent in SMILES sequences (Wang et al., 2019; Chithrananda et al., 2020; Bagal et al., 2021; Ross et al., 2022) , and significant efforts have been made to align the learned knowledge from SMILES with textual knowledge Edwards et al. (2022); Pei et al. (2023); Liu et al. (2023b), aiming to boost the application effectiveness in downstream tasks such as property prediction and molecular design. A core issue in these model designs is how to more efficiently mine important molecular-related knowledge from SMILES data. Therefore, this paper seeks to address this issue by attempting to design a SMILES language model with enhanced modeling capabilities.

Current designs of SMILES language models often follow similar approaches used for natural language models, such as predicting missing tokens in a corrupted SMILES context (e.g., MLM, CLM). However, this can also lead to many problems. **(i)** SMILES data differs from text in that individual tokens in text are independent semantic units (like words, phrases, or subwords), whereas in SMILES, individual tokens often represent single atoms, chemical bonds, or special symbols. However, molecules typically depend more on specific substructures (like functional groups) for functionality, meaning that the functional information usually reflects at the substructure level. This suggests that if a SMILES language model focuses solely on modeling the relationships between individual tokens and their SMILES contexts, it would struggle to learn the semantic information of specific molecular substructures. **(ii)** Moreover, predicting a single missing token in a given SMILES context is very easy. This can cause the model's capacity to reach a saturation point quickly during training, preventing it from acquiring additional and more comprehensive molecular knowledge. As

a result, this affects the model's scalability and its effectiveness in generalizing to a wider range of molecular data. **(iii)** Additionally, as these models are trained on corrupted SMILES contexts which contain the special symbol `[MASK]` that does not exist in real SMILES, their ability to model the semantic content of complete SMILES is compromised.

To address these challenges, we propose an edit-based SMILES language model with fragment-level supervision. **(i)** First, to help the model learn richer substructure-related molecular information, we designed a fragment-level supervision signal. By randomly dropping substructures in molecules and having the model learn to recover this information, the model can acquire more comprehensive fragment-level semantic knowledge. **(ii)** We also devised an edit-based pre-training objective, allowing us to input a valid SMILES sequence and restore missing substructures through edits.

In summary, the contributions of this paper are threefold:

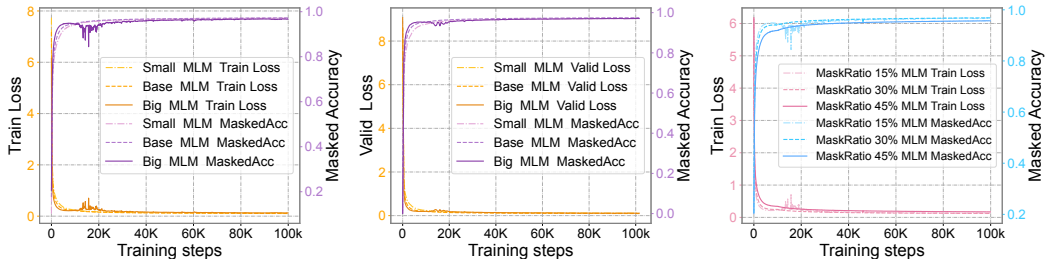
- We analyze the behavior of current SMILES masked language models (MLMs) during the pre-training phase and downstream tasks, and further identify that current SMILES MLMs exhibit rapid saturation problem during pre-training and have a weak ability to model the molecular substructure information. Previous research lacks a systematic analysis of these issues.
- To address the limitations of existing models, we introduce the first edit-based pre-trained language model for SMILES, enabling the transformation of a valid SMILES sequence into a structurally closely related one. This approach resolves the train-inference mismatch issue in current SMILES language models. Additionally, we incorporate fragment-level supervision, enhancing the model's ability to learn richer semantic knowledge from SMILES and improving its overall performance.
- Extensive experiments demonstrate that the SMI-EDITOR model achieves state-of-the-art performance on multiple molecular property prediction tasks, surpassing several 3D molecular models, and the ablation and analysis experiments designed in this study confirm the effectiveness and better scalability of the SMI-EDITOR model.

2 RELATD WORKS

SMILES, as a key sequential representation of molecular information, has become a significant focus in molecular representation learning. Numerous Pre-trained SMILES Language Models (Wang et al., 2019; Chithrananda et al., 2020; Ross et al., 2022) have been proposed to address various challenges in SMILES-based molecular modeling, and their effectiveness has been validated across many downstream tasks (Bagal et al., 2021; Tong et al., 2021) . Edit-based generative models, another important approach to sequence modeling, have been widely applied in tasks such as machine translation, summarization, and grammatical error correction. In this section, we first introduce representative work in Pre-trained SMILES Language Models, followed by a discussion on Edit-based Language Models for sequence modeling.

2.1 PRE-TRAINED SMILES LANGUAGE MODEL

Similar to text, SMILES is a type of sequential information. Early pre-trained SMILES language models adopted methods from text modeling. Wang et al. (2019) introduced SMILES-BERT, inspired by the BERT model (Devlin et al., 2018) and the masked language model (MLM) training objective, demonstrating its effectiveness in molecular property prediction tasks. Likewise, Chithrananda et al. (2020) developed ChemBERTa, based on RoBERTa(Liu et al., 2019b), to capture SMILES semantics using the MLM objective. Ross et al. (2022) proposed Molformer, trained on a larger dataset with MLM training objective, showing that SMILES models can capture molecular properties and structure. As a result, MLM-based models have become dominant in SMILES representation learning. In addition, generative pre-training approaches have also been applied. The MolGPT model (Bagal et al., 2021) uses an autoregressive approach, while Tong et al. (2021) applied generative models to drug design. Liu et al. (2023b) further unified SMILES and textual data through generative pre-training. Overall, pre-trained SMILES language models, particularly those based on the MLM objective, are now essential in molecular modeling research.



(a) Training Curves: Model Scales (b) Validation Curves: Model Scales (c) Training Curves: Mask Ratios

Figure 2: **Rapid Saturation Problem.** We train SMILES MLMs of various sizes and masking ratios using the dataset from Zhou et al. (2023). Figure (a) displays the training loss and masking prediction accuracy of different-sized models, showing a rapid decrease in loss and an increase in accuracy at the start of the training. Figure (b) presents similar trends for the validation set. Figure (c) illustrates the training loss and accuracy for models with different masking ratios, showing similar patterns.

2.2 Edit-based Language Model

Edit-based sequence generation offers a faster, more flexible alternative to traditional autoregressive methods. Malmi et al. (2019) introduced the LASERTAGGER model, which uses tags (keep, delete, add) to edit sequences, while the Felix model (Mallinson et al., 2020) combines a pointer-based mechanism with an MLM model to handle insertions and deletions. Recognizing that edit operations from an input sequence to a target output can be diverse and difficult to compute directly, Gu et al. (2019) developed the Levenshtein Transformer (LevT) model. This model calculates the minimum levenshtein distance between the input and target sequences to create an optimal sequence of edit operations, using this as the training objective. This approach significantly improves performance on tasks such as machine translation and post-editing. LevT was further applied to lexically constrained translation tasks with notable success (Susanto et al., 2020). To resolve inconsistencies between training and inference, Zheng et al. (2023) introduced a dual training objective, improving performance in tasks such as summarization and grammatical error correction. Overall, edit-based models have proven highly efficient across many tasks and are a key research area in sequence modeling.

3 Understanding the Behavior of MLM

Masked Language Model (MLM) is a widely used approach for modeling textual information and has been extensively applied in SMILES modeling (Wang et al., 2019; Chithrananda et al., 2020; Ross et al., 2022). During the training process of MLM model, tokens in the SMILES sequence, including single atoms, chemical bonds, or special symbols, are randomly masked with a fixed masked ratio of 15%. The model is then tasked with learning to accurately predict these masked tokens, as shown in Figure 1. To further assess the effectiveness and capabilities of MLMs for SMILES data, we conducted a series of experiments.

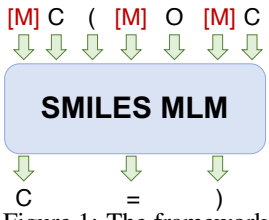
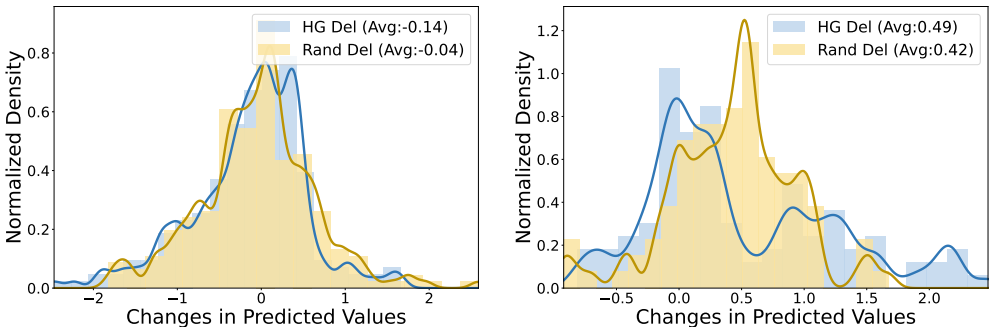


Figure 1: The framework of MLM for SMILES.

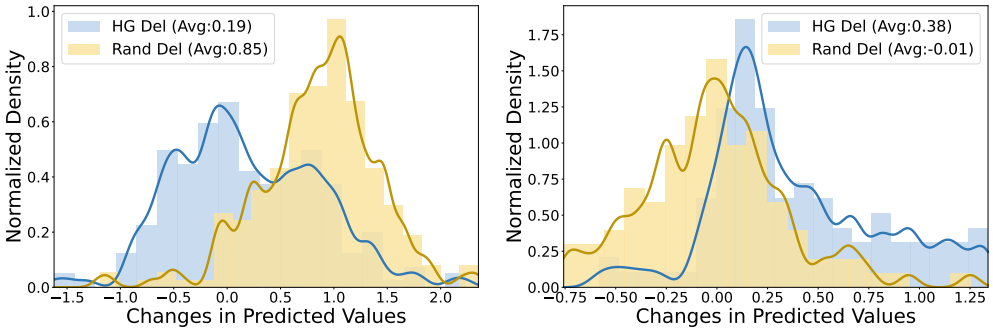
3.1 Rapid Saturation Problem

Rapid Saturation During Pre-training. To investigate whether the MLM model experiences rapid saturation during training and how this issue impacts the model's scalability, we trained MLMs of various scale and compared their training curves (Details of models with different scale can be found in Appendix C). As shown in Figure 2a, while the training loss rapidly decreases, the mask-prediction accuracy on training set of the models quickly rose above 90% within the first 5,000 steps. By around 10,000 steps, the mask-prediction accuracy on training set of all models exceeded 95%. A similar rapid saturation phenomenon is observed on the validation set. As shown in Figure 2b, the validation loss drops quickly after training begins, while mask-prediction accuracy rises sharply. All models of varying scales exhibit the same rapid saturation phenomenon, including the small model with only 6.7M parameters. These results indicate that the MLM pre-training task is overly simplistic, allowing even very small models to converge quickly, which limits the model's capacity and scalability for



(a) ESOL Dataset (b) FreeSolv Dataset

Figure 3: **Substructure Semantics Modeling.** We compared two molecular perturbation methods—removing hydrophilic groups and randomly deleting atoms—and their effects on the model's predictions of hydrophilicity and related properties. Figure (a) presents the impact of these perturbations on model predictions in the ESOL dataset, including the distribution of prediction changes. The average prediction change is similar for both methods (-0.14 vs. -0.04) and shows similar distributions. Figure (b) shows the effects on the FreeSolv dataset, also with similar average prediction change.



(a) ESOL Dataset (b) FreeSolv Dataset

Figure 4: **Substructure Semantics Modeling from SMI-EDITOR.** We compared the effects of two molecular perturbation methods on the SMI-EDITOR's predictions of hydrophilicity and related properties. Figure (a) and Figure (b) show that the impact of deleting hydrophilic groups (HG Del) and randomly deleting atoms (Rand Del) on the model's predictions differs significantly, both in the average change in prediction values and their distributions.

more complex tasks. We also test the performance of MLM models of different sizes and training steps on downstream tasks, and the detailed results can be found in Appendix D. The results on downstream tasks also suggest that the scalability of MLM models is limited.

Different Mask Ratio Cannot Alleviate Rapid Saturation. One possible reason for the rapid saturation problem in MLM pre-training is that only 15% of tokens are masked during training, providing the model with too little training information and making token prediction too easy, which leads to rapid saturation. To investigate whether this is the cause, we trained large-scale MLM models with different mask ratios (15%, 30%, 45%). The training curves are shown in Figure 2c. The results show that MLM models with different mask ratios all show a rapid decrease in training loss at the beginning of training, quickly converging to a very low level. And even with a mask ratio of 45%, the training loss still drops rapidly, and by 10K steps, the mask-prediction accuracy already exceeds 92%. This indicates that increasing the mask ratio does not prevent the MLM model from converging quickly, limiting its scalability. It further demonstrates that the emergence of rapid saturation is not due to a low mask ratio, but rather because the MLM training task is relatively simple and lacks sufficient information for SMILES data.

3.2 CHALLENGES IN MODELING SUBSTRUCTURE SEMANTICS

To evaluate the ability of MLM to learn molecular substructures semantics, such as functional groups, we design experiments to analyze whether the model can accurately capture functional group information closely related to molecular properties. We use two molecular property prediction

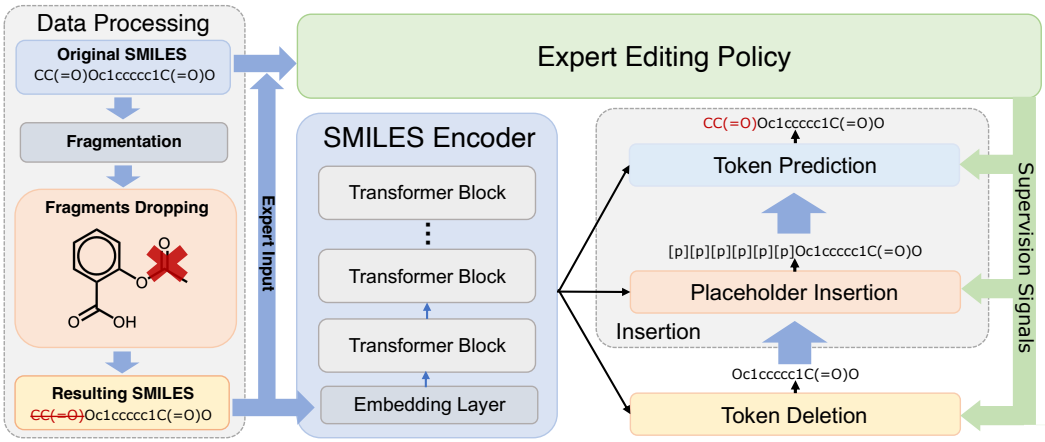


Figure 5: **Overall Framework of SMI-EDITOR**. The framework includes a data processing module, a SMILES encoder, and an edit-based pre-training process. In data processing, some fragments of the input molecule are randomly removed, and the resulting SMILES is fed into the model. The pre-training goal for the model is to edit the corrupted SMILES to recover the original SMILES. To enable this, we add three different heads for token deletion, placeholder insertion, and token prediction to the SMILES encoder (see Appendix A for details). An expert provides training signals for these operations to help the model learn how to recover the original SMILES through editing.

datasets, ESOL and FreeSolv (Wu et al., 2017), both of which are highly relevant to molecular hydrophilicity. Specifically, the ESOL dataset provides information on the water solubility of molecules, while the FreeSolv dataset focuses on hydrogen free energy, both of which are tightly linked to hydrophilic groups within the molecules.

In our approach, we first fine-tune the MLM model on these datasets using linear probing. Then, we traverse the SMILES of all molecules in the datasets and remove the hydrophilic groups (e.g., $-\mathtt{OH}$, $-\mathtt{COOH}$, $-\mathtt{NH_2}$, etc.) identified in each molecule. We compare the predicted values of the model before and after the removal. As a control, we also randomly delete atoms from these molecules and compare the predicted changes in molecular properties.

As shown in Figure 3, the changes in predicted values after deleting hydrophilic groups (HG Del) are similar to those from random deletions (Rand Del) in both the ESOL and FreeSolv datasets. This indicates that the model struggles to differentiate between the effects of removing hydrophilic groups and random atoms on molecular properties. This result further suggests that the MLM model fails to effectively capture the semantic information of important substructures in SMILES.

4 EDIT-BASED PRE-TRAINING FRAMEWORK

To address the limitations of MLM-based SMILES language models, we propose a novel SMILES language model that employs an edit-based training objective. To enhance the model's ability to capture the semantics of substructures within molecules, we introduce fragment-level supervision during pre-training, which includes randomly discarding parts of substructures and requiring the model to predict the missing components. This method enables the model to effectively learn substructure semantics. In contrast, MLM models only operate on corrupted SMILES contexts with unreal `[MASK]` symbols, leading to inconsistencies between training and testing. To mitigate this issue, we input complete and valid SMILES sequences into the model, requiring it to reconstruct the missing substructures through an editing approach. Moreover, the editing framework offers greater flexibility compared to MLM, as it imposes no specific restrictions on input forms. This allows us to create more versatile model inputs by removing certain substructures from a molecule, converting it back to SMILES, and then feeding it into the model. In this section, we will discuss SMI-EDITOR from both model design and pre-training framework perspectives.

4.1 SMILES ENCODER WITH EDITING OPERATIONS

In the edit-based pre-training process, the model should be capable of modeling editing operations. Specifically, when given a SMILES sequence with missing substructures, the model needs to accu-

rately predict the editing operations required to complete these missing substructures. To address this, we have designed a SMILES encoder that supports editing operation modeling.

Model Architecture. The core architecture of the model remains a Transformer Encoder built from multiple stacked Transformer blocks. Each transformer blocks contains a multi-head self-attention layer and a feed-forward layer (Vaswani et al., 2017). The SMILES representations extracted by the Transformer Encoder are then passed to the Editing Operations Head, which is responsible for predicting the required editing operations. Similar to existing Edit-based Language Models (Gu et al., 2019), the model needs to handle two types of editing operations: deletion and insertion. Specifically, the model completes missing parts of the SMILES sequence by removing redundant parts and inserting the missing substructures. This requires an additional prediction head to handle these editing operations effectively. Consequently, the model gains the capability to model editing operations proficiently.

Deletion Operations Head. For a given input token, there are two possible deletion operations: delete or not delete. Therefore, the deletion operation is essentially a token-level binary classification problem. Let $\boldsymbol{x}_i^E$ denote the representation of the i-th input token extracted by the Encoder. The probability of deleting the i-th token, denoted as $\pi_\theta^{\text{del}}(i)$, can be expressed as:

$$\pi_\theta^{\text{del}}(i) = \text{Softmax}(\boldsymbol{W}_d^T \boldsymbol{x}_i^E)$$

Here, $\boldsymbol{W}_d$ is a matrix of size $H \times 2$, H is the hidden size.

Insertion Operations Heads. Modeling the insertion operation is more complex compared to the deletion operation. Similar to LevT, the insertion operation is modeled in two steps. In the first step, the model needs to predict the positions and number of tokens to be inserted into the original input sequence. At these predicted positions, placeholders `[P]` are inserted to represent the tokens that will be added. In the second step, the model predicts the actual tokens for each placeholder `[P]`.

For the first step, given the length of the SMILES, we constrain the model to insert at most 255 tokens at a time. Thus, this step can be seen as a 256-class classification problem for each token position. The probability of inserting tokens at the i-th position, denoted as $\pi_\theta^{\text{ins}}(i)$, can be expressed as:

$$\pi_\theta^{\text{ins}}(i) = \text{Softmax}(\boldsymbol{W}_{in}^T \boldsymbol{x}_i^E)$$

Here, $\boldsymbol{W}_{in}$ is a matrix of size $H \times 256$.

In the second step, the task is conceptually similar to what is done in MLM models. For each `[P]` symbol, the model needs to predict the probability distribution over the vocabulary for the token at that position. Therefore, the probability distribution for the token corresponding to the `[P]` at the i-th position, denoted as $\pi_\theta^{\text{tok}}(i)$, can be expressed as:

$$\pi_\theta^{\text{tok}}(i) = \text{Softmax}(\boldsymbol{W}_{tok}^T \boldsymbol{x}_i^E)$$

Here, $\boldsymbol{W}_{tok}$ is a matrix of size $H \times vob$, where vob represents the size of the vocabulary.

4.2 Edit-based Pre-training with Fragment-level Supervision

After constructing the SMILES encoder with editing operations, the next crucial step is to build an edit-based pre-training framework and provide fragment-level self-supervised training signals. Unlike traditional masked language models, an edit-based model can transform a valid SMILES input into the target SMILES through editing operations. First, we fragment the input molecule using rule-based molecule fragmentation, breaking it into different fragments. A subset of these fragments is then randomly selected and removed from the original molecule. The resulting corrupted molecule is converted back into a SMILES representation and fed into the SMILES Encoder. We train the encoder to predict the correct editing process required to restore the corrupted molecule to its original and complete form.

Molecule Fragmentation and Fragments Dropping. To provide the model with fragment-level training signals, we first need to split the input molecule M into multiple fragments $\{f_1, f_2, ...\}$. The BRICS algorithm (Degen et al., 2008) is a commonly used method for molecular fragmentation, which divides a molecule into fragments based on predefined rules, such as molecular functional groups. However, BRICS often generates relatively large fragments, and removing these fragments can overly disrupt the molecule, leading to the loss of important core structures, like rings. To address this, we adopt a similar fragmentation approach with RMCF (Wang et al., 2022a), where we further split the links between rings and side chains on top of BRICS, resulting in smaller molecular fragments. After cutting the molecule, we randomly select and discard some fragments with a certain probability. The remaining fragments are then reassembled into a corrupted molecule $\hat{M}$. The model is tasked with recovering the original molecule's SMILES from the given SMILES of $\hat{M}$ through an edit-based approach.

Edit-based Training Objective with Dual Loss. The core pre-training task of the SMI-EDITOR model is to take the SMILES of a corrupted molecule $\hat{M}$ as input and attempt to restore it to the SMILES of the original molecule M through an editing process. Specifically, SMI-EDITOR takes the SMILES of a corrupted molecule $\hat{M}$ as input, and generates a valid SMILES output through deletion and insertion operations. However, traditional edit-based models like LevT only provide training signals for deletion by teaching the model to remove incorrect tokens it inserted. This limits the model's ability to learn how to delete the incorrect parts in the input SMILES. To overcome this problem, we introduce the dual deletion loss, which trains the model to correctly delete incorrect tokens from the initial SMILES input.

To provide proper training signals for the model, we adopt the imitation learning method from LevT, which supervises the model by minimizing the Levenshtein distance between the input and target output. The training objective is defined as follows:

$$\mathcal{L}_\theta^{\text{DualDel}} = -\sum_{\substack{y_i \in \hat{M} \\ d_i^* \in d^*}} \log \pi_\theta^{\text{del}}(d_i^* | i, \hat{M})$$

Here, d^* is the optimal deletion action determined by an expert to minimize the Levenshtein distance to the target output y^* which is the SMILES of molecule M. This is formulated as $d^* = \text{argmin}_d \mathcal{D}(y^*, \varepsilon(\hat{M}, d))$, where $\mathcal{D}$ is the Levenshtein distance, π_θ^{del} is the Deletion Classifier, and ε represents the environment in LevT's Markov Decision Process. $\varepsilon(\hat{M}, d)$ applies the deletion action d to the initial input SMILES $\hat{M}$, removing selected tokens.

In addition to the dual deletion loss, we retain the original LevT's training objective $\mathcal{L}_\theta^{\text{LevT}}$ (details of $\mathcal{L}_\theta^{\text{LevT}}$ can be found in Appendix A), which supervises both deletion and insertion actions by minimizing the Levenshtein distance between the input and target output. Thus, the final training objective is:

$$\mathcal{L}_\theta = \mathcal{L}_\theta^{\text{DualDel}} + \mathcal{L}_\theta^{\text{LevT}}$$

Through this edit-based pre-training process, we equip the SMI-EDITOR model with fragment-level training signals, enabling it to learn how to transform the SMILES of a corrupted molecule $\hat{M}$ into the SMILES of the original molecule M via fragment editing.

5 EXPERIMENTS

In this section, we first evaluate the performance of SMI-EDITOR on molecular property prediction tasks and compare it with baseline models (Section 5.2). The results show that SMI-EDITOR outperforms both the MLM and 3D molecular models, achieving state-of-the-art performance. To further validate the model design and pre-training framework, we conduct ablation studies on training signals and editing operations (Section 5.3). Additionally, analytical experiments confirm that SMI-EDITOR has a stronger ability to capture the semantics of molecular substructures compared to MLM models. Analysis of the training curves also demonstrates that SMI-EDITOR mitigates the issue of

Table 1: The overall results on 7 molecule property classification datasets. We report ROC-AUC score (higher is better) under scaffold splitting. The best results are **bold**. The second-best results are underlined. For more detailed information about the dataset, please refer to Table 7.

Datasets # Molecules	BACE↑ 1531	BBBP↑ 2039	Tox21↑ 7831	SIDER↑ 1427	MUV↑ 93087	ClinTox↑ 1478	ToxCast↑ 8575	Mean↑ -
D-MPNN	80.9	71.0	75.9	57.0	78.6	90.6	65.5	74.2
Attentive FP	78.4	64.3	76.1	60.6	76.6	84.7	63.7	72.1
N-Gram$_{\mathrm{RF}}$	77.9	69.7	74.3	**66.8**	76.9	77.5	-	-
GROVER	**82.6**	70.0	74.3	64.8	62.5	81.2	65.4	71.5
GraphMVP	81.2	72.4	75.9	63.9	77.7	79.1	63.1	73.3
InfoGraph	77.8	67.5	73.2	59.9	74.1	76.5	63.7	70.4
MolCLR	82.4	72.2	75.0	58.9	79.4	91.2	**69.2**	75.5
Mole-BERT	80.8	71.9	76.8	62.8	78.6	78.9	64.3	73.4
3D InfoMax	79.7	69.1	74.5	60.6	74.4	79.9	64.4	71.8
MoleculeSDE	80.4	73.2	76.5	59.6	79.9	86.6	65.2	74.5
SMI-MLM	77.8	68.6	75.1	61.2	75.1	89.8	64.9	73.2
SMI-EDITOR	80.3	**77.4**	**77.1**	63.0	**80.2**	**98.9**	67.4	**77.8**

rapid saturation and enhances the training stability (Section 5.4). Additionally, we provide details on the training data, baseline models, and implementation used in the experiments (Section 5.1).

5.1 EXPERIMENT SETTINGS

Datasets. For pre-training, we use the large-scale molecular dataset provided by Zhou et al. (2023), which includes SMILES information for 19 million molecules. For fine-tuning, we employ the widely recognized MoleculeNet benchmark (Wu et al., 2018). For more details about this dataset, please refer to Appendix H. We follow the same data split as used by Zhou et al. (2023) and tokenize SMILES sequences with the regular expression from Schwaller et al. (2018).

Baselines. We evaluate our approach against various supervised learning and pre-training baselines, including both SMILES-based and 3D molecular pre-training models. The supervised methods include D-MPNN (Yang et al., 2019) and AttentiveFP (Xiong et al., 2019), both of which are based on graph neural networks (GNNs). For 2D and 3D molecular pre-training, we consider several methods: N-gram (Liu et al., 2019a), GROVER (Rong et al., 2020), GraphMVP (Liu et al., 2021), MolCLR (Wang et al., 2022b), InfoGraph (Sun et al., 2019), Mole-BERT (Xia et al.), 3D InfoMax (Stärk et al., 2022), and MoleculeSDE (Liu et al., 2023a). For a fair comparison, we train a SMILES model based on MLM pre-training, referred to as SMI-MLM, using the same training data, model architecture, and training hyperparameters as SMI-EDITOR.

Implementation Details. We use a Transformer block with a hidden size of 768 and 12 attention heads, comprising 12 layers in the SMILES encoder, which contains a total of 86.3 million trainable parameters. During pre-training, the fragment drop ratio is set to 0.15. For downstream tasks, we use the same fine-tuning dataset established by Uni-Mol. (cf. Appendix F for more details about hyper-parameter configuration.)

5.2 RESULTS ON MOLECULAR PROPERTY CLASSIFICATION TASKS

Tasks Details. We evaluate SMI-EDITOR on the MoleculeNet (Wu et al., 2017) benchmark and compare its performance with baseline models. We evaluate SMI-EDITOR on 7 widely used molecular property prediction tasks. For detailed hyperparameters used in different tasks, please refer to Appendix G. For all seven molecular property prediction tasks, we input the normalized SMILES information into the model and perform further fine-tuning for each task. The hyperparameters for each task are detailed in the supplementary materials. ROC-AUC is used as the evaluation metric, and the results are summarized in Table 1.

Results. SMI-EDITOR achieves state-of-the-art (SOTA) performance on 4 tasks and closely matches the SOTA models on the remaining tasks. Compared to the MLM model SMI-MLM, which uses the same training settings, training data, and evaluation processes for downstream tasks,

SMI-EDITOR demonstrates superior performance across all seven tasks, validating the effectiveness of its pre-training framework. Additionally, SMI-EDITOR outperforms several molecular representation learning models that utilize 3D information, indicating that SMILES contains important and rich semantic information related to molecular properties and that SMI-EDITOR effectively captures this information. SMI-EDITOR also demonstrated the strongest average performance across all 7 tasks, indicating that it outperforms other baseline models in these prediction tasks.

5.3 ABLATION STUDIES

5.3.1 ABLATION STUDIES ON FRAGMENT-LEVEL SUPERVISION

Table 2: **Ablation Studies on Fragment-level Supervision.** Fragment-level supervision provide more informative and useful training signals than atom-level supervision and are crucial for helping the model learn multi-level semantic information in molecules.

Method	BACE↑	BBBP↑	Tox21↑	SIDER↑	ToxCast↑	Mean↑
SMI-EDITOR-AtomsDropping	80.0	<u>73.4</u>	<u>76.5</u>	59.2	<u>66.6</u>	<u>71.1</u>
SMI-EDITOR-AtomsMasking	**80.4**	73.2	75.0	58.3	64.6	70.3
SMI-MLM	77.8	68.6	75.1	<u>61.2</u>	64.9	69.5
SMI-EDITOR	<u>80.3</u>	**77.4**	**77.1**	**63.0**	**67.4**	**73.0**

Experimental Settings. To explore the impact of fragment-level supervision signals on model performance, we train SMI-EDITOR models using two different pre-training strategies. The first model, SMI-EDITOR-AtomsDropping, replaces the fragment dropping process in pre-training with random atom dropping. After discarding certain atoms, we input the modified SMILES into the model, asking it to restore the original SMILES through an editing approach. The second model, SMI-EDITOR-AtomsMasking, uses random token masking similar to MLM, where selected tokens are replaced with `[MASK]`, and the model is tasked with restoring the original SMILES via editing. The performance of these models is presented in Table 2.

Results Analysis. The results show a significant decline in performance when fragment dropping is replaced with random atom dropping (SMI-EDITOR-AtomsDropping vs. SMI-EDITOR), indicating that the fragment-level supervision signal enables the model to learn more important and nuanced semantic information. Furthermore, when random atom dropping is replaced with random token masking, performance decreases again (SMI-EDITOR-AtomsMasking vs. SMI-EDITOR-AtomsDropping). This suggests that while both random token masking and random atom dropping introduce atom-level training signals, the introduction of the unrealistic special symbol `[MASA]` through token masking adversely affects model performance. Compared to these two models, SMI-MLM exhibits even poorer performance, demonstrating that this editing training approach effectively helps the model learn richer semantic knowledge.

5.3.2 ABLATION STUDIES ON EDITING OPERATIONS

Table 3: **Ablation Studies on Editing Operations.** The placeholder insertion process, which is absent in MLM models, enables the model to learn richer and more diverse semantic information.

Method	BACE↑	BBBP↑	Tox21↑	SIDER↑	ToxCast↑	Mean↑
w/o PlhIns	76.1	69.7	76.9	55.5	<u>66.2</u>	68.9
w/o TokPred	<u>79.8</u>	69.2	75.4	57.4	65.9	69.5
w/o TokDel	79.0	<u>73.5</u>	**77.3**	<u>61.9</u>	64.9	<u>71.3</u>
w/o DualDel	78.4	70.1	76.4	59.5	64.4	69.8
SMI-EDITOR	**80.3**	**77.4**	<u>77.1</u>	**63.0**	**67.4**	**73.0**

Experimental Settings. To investigate the impact of different training signals from the editing operations in the SMI-EDITOR model on its performance, we train four variations of the SMI-EDITOR model. These models are obtained by removing the training signals for placeholder insertion, token prediction, token deletion, and dual token deletion (setting the training loss weight to zero), corresponding to the three editing operations in the original LevT model and the dual deletion loss. The detailed results are presented in Table 3.

Results Analysis: Why SMI-EDITOR is Better than SMI-MLM. The results indicate that the ablation of any of these four editing operations leads to a significant decline in model performance. Notably, removing the placeholder insertion operation results in the largest performance loss. This operation primarily models the position of missing fragments within the SMILES, highlighting the importance of teaching the model to predict the locations of these fragments for capturing critical semantic information and improving performance. In contrast, the MLM model attempts to predict masked tokens based on their given positions, which simplifies the pre-training task and limits the model's exposure to important semantic information, ultimately affecting its performance. Moreover, SMI-EDITOR provides supervision signals for each token in the sequence, but the MLM model only provides supervision signals for `[MASK]` tokens, which limits the semantic richness of the model.

Results Analysis: Dual Deletion Loss is More Useful. Additionally, the ablation of the dual deletion operation also causes a significant decline in model performance, with a more pronounced drop than when token deletion is removed. This indicates that the dual deletion loss incorporated into our model provides more useful and richer training signals than token deletion loss in LevT.

5.4 ANALYTICAL EXPERIMENTS

SMI-EDITOR Understands Substructure Semantics. Similar to the analysis in Section 3.2, we tested SMI-EDITOR's response to two different molecular perturbation methods on the ESOL and FreeSolv datasets. As shown in Figure 4, compared to the results in Figure 3, SMI-EDITOR exhibits distinct prediction changes for the two perturbation methods on both the ESOL and FreeSolv datasets. This indicates that SMI-EDITOR can clearly differentiate between the impact of removing hydrophilic groups and randomly deleting atoms on molecular properties, demonstrating that it models the semantics of key molecular substructures more effectively than the MLM model.

SMI-EDITOR Enhances Training Stability and Model Scalability. We train SMI-EDITOR of different sizes and compare their training curve variations. As shown in Figure 6, the losses of the SMI-EDITOR models consistently exhibit a more pronounced downward trend throughout the training process compared to the MLM models (Figure 2a), further alleviating the rapid saturation problem. Additionally, unlike the MLM, the training loss of the SMI-EDITOR shows more distinct differences across sizes. As the model size increases, the loss steadily decreases, with the larger model (Big Model) converging more stably than the MLM, indicating better scalability for SMI-EDITOR. We also analyze the training and validation loss curves for the three types of editing operations in SMI-EDITOR, confirming the model's scalability during pre-training; detailed results can be found in Appendix B. Additionally, we evaluate the performance of SMI-EDITOR models of different sizes on downstream tasks, demonstrating that SMI-EDITOR exhibits better scalability and stability compared to the MLM model (SMI-MLM). Detailed results can be found in Appendix E.

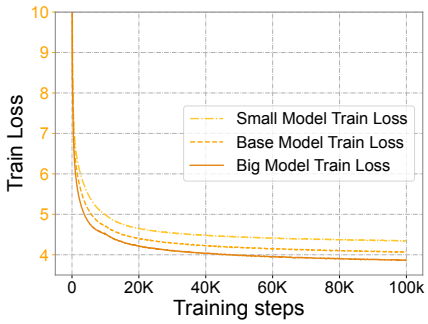


Figure 6: The training loss curves of different-sized SMI-EDITOR models. The loss curves consistently show a stable downward trend throughout the training process, and the model loss gradually decreases as the model size increases.

6 CONCLUSIONS

In this paper, we analyze the behavior and shortcomings of masked language models (MLMs) on SMILES data. Through the examination of training curves, we demonstrate that training MLMs on SMILES data encounters rapid saturation issues. Further analytical experiments reveal that MLMs struggle to effectively capture the semantics of important molecular substructures. To address these issues, we propose the edit-based pre-training molecular representation learning model SMI-EDITOR, which enhances the model's ability to capture substructure semantics by learning how to recover the missing fragments through edit operations. Extensive experiments on molecular property prediction tasks validate the effectiveness of SMI-EDITOR, and ablation studies confirm the advantages of its design over traditional MLMs in modeling molecular substructure semantics and training stability.

REFERENCES

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. Molgpt: molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling*, 62(9): 2064–2076, 2021.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Jorg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503, 2008.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language. *arXiv preprint arXiv:2204.11817*, 2022.

Zhiyi Fu, Wangchunshu Zhou, Jingjing Xu, Hao Zhou, and Lei Li. Contextual representation learning beyond masked language modeling. *arXiv preprint arXiv:2204.04163*, 2022.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in neural information processing systems*, 32, 2019.

Yuqiang Han, Xiaoyang Xu, Chang-Yu Hsieh, Keyan Ding, Hongxia Xu, Renjun Xu, Tingjun Hou, Qiang Zhang, and Huajun Chen. Retrosynthesis prediction with an iterative string editing model. *Nature Communications*, 15(1):6404, 2024.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Shengchao Liu, Mehmet F Demirel, and Yingyu Liang. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. *Advances in neural information processing systems*, 32, 2019a.

Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

Shengchao Liu, Weitao Du, Zhi-Ming Ma, Hongyu Guo, and Jian Tang. A group symmetric stochastic differential equation model for molecule multi-modal pretraining. In *International Conference on Machine Learning*, pp. 21497–21526. PMLR, 2023a.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b.

Zequn Liu, Wei Zhang, Yingce Xia, Lijun Wu, Shufang Xie, Tao Qin, Ming Zhang, and Tie-Yan Liu. Molxpt: Wrapping molecules with text for generative pre-training. *arXiv preprint arXiv:2305.10688*, 2023b.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. Felix: Flexible text editing through tagging and insertion. *arXiv preprint arXiv:2003.10687*, 2020.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. Encode, tag, realize: High-precision text editing. *arXiv preprint arXiv:1909.01187*, 2019.

Qizhi Pei, Wei Zhang, Jinhua Zhu, Kehan Wu, Kaiyuan Gao, Lijun Wu, Yingce Xia, and Rui Yan. Biot5: Enriching cross-modal integration in biology with chemical knowledge and natural language associations. *arXiv preprint arXiv:2310.07276*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571, 2020.

Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12):1256–1264, 2022.

Philippe Schwaller, Theophile Gaudin, David Lanyi, Costas Bekas, and Teodoro Laino. "found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical science*, 9(28):6091–6098, 2018.

Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*, pp. 20479–20502. PMLR, 2022.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. Lexically constrained neural machine translation with levenshtein transformer. *arXiv preprint arXiv:2004.12681*, 2020.

Xiaochu Tong, Xiaohong Liu, Xiaoqin Tan, Xutong Li, Jiaxin Jiang, Zhaoping Xiong, Tingyang Xu, Hualiang Jiang, Nan Qiao, and Mingyue Zheng. Generative models for de novo drug design. *Journal of Medicinal Chemistry*, 64(19):14011–14027, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Lihao Wang, Yi Zhou, Yiqun Wang, Xiaoqing Zheng, Xuanjing Huang, and Hao Zhou. Regularized molecular conformation fields. *Advances in Neural Information Processing Systems*, 35:18929–18941, 2022a.

Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. Smiles-bert: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pp. 429–436, 2019.

Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3):279–287, 2022b.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. *CoRR*, abs/1703.00564, 2017. URL `http://arxiv.org/abs/1703.00564`.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z Li. Mole-bert: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*.

Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8): 3370–3388, 2019.

Kangjie Zheng, Longyue Wang, Zhihao Wang, Binqi Chen, Ming Zhang, and Zhaopeng Tu. Towards a unified training for levenshtein transformer. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: a universal 3d molecular representation learning framework. 2023.

A Details of Levenshtein Transformer

The Levenshtein Transformer (LevT) is a non-autoregressive editing generation model that employs a three-step editing process: token deletion, placeholder insertion, and token prediction. LevT is trained using imitation learning with a dual policy: (i) learning to insert tokens by predicting those that have been randomly deleted from the target output, and (ii) learning to delete tokens by identifying incorrect tokens generated by the insertion module. Below are more details about the training objective of LevT, denoted as $\mathcal{L}_\theta^{\text{LevT}}$.

Placeholder Insertion Loss. In this step, the model needs to determine how many placeholders `[P]` should be inserted at specific positions in the original input, which will later be replaced by concrete words in subsequent steps. Therefore, the core operation here is a classification task that predicts how many words need to be inserted after each token in the input sequence. For practical implementation, LevT limits the maximum number of words that can be inserted after each token to 255. Thus, this step essentially becomes a 256-class classification task at each token, predicting the number of words (0-255) to insert after each token. This process can be represented as follows:

$$\mathcal{L}_\theta^{\text{ins}} = -\sum_{\substack{y_i \in y_0 \\ p_i^* \in p^*}} \log \pi_\theta^{\text{ins}}(p_i^* | i, y_0)$$

where p_i^* is the optimal placeholder insertion action found by the expert that minimizes the Levenshtein distance to the target output y^* which is the SMILES of molecule M, and can be formalized as $p_i^* = \text{argmin}_p \mathcal{D}(y^*, \varepsilon(y_0, p))$, y_0 is the initial input of the model which is the SMILES of molecule $\hat{M}$, $\mathcal{D}$ is the Levenshtein distance measurement, π_θ^{del} is LevT's Deletion Classifier, and ε is the environment in the Markov Decision Process of LevT which receives editing actions and returns the modified sequence, and $\varepsilon(y_0, p)$ means applies the insertion action p to the initial input sequence y_0 (e.g. insert some placeholders in y_0). Details of ε can be found in LevT's framework (Gu et al., 2019).

Token Prediction Loss. In this step, the task is to predict a real word for each placeholder `[P]` in the sequence $y_1 = \varepsilon(y_0, p^*)$ that has had placeholders inserted. This process is very similar to that of MLM, as it essentially involves a classification problem where the number of classes is equal to the size of the vocabulary.

$$\mathcal{L}_\theta^{\text{tok}} = -\sum_{\substack{y_i \in y_1, t_i^* \in t^* \\ y_i = <\text{[P]}>}} \log \pi_\theta^{\text{tok}}(t_i^* | i, y')$$

where t_i^* is the optimal insertion action found by the expert that minimizes the Levenshtein distance to the target output y^*, y_1 is the modified sequence by applying the optimal placeholder action p^* to the input sequence y_0, and these terms can be formalized as: $t_i^* = \text{argmin}_t \mathcal{D}(y^*, \varepsilon(y_1, t))$, $y_1 = \varepsilon(y_0, p^*)$, d^* or $p^* = \text{argmin}_{d,p} \mathcal{D}(y^*, \varepsilon(y_0, \{d, p\}))$. π_θ^{tok} is token classifier.

Token Deletion Loss. In the insertion step, the model may have inserted incorrect words, so in this step, it needs to predict which of the previously inserted words are incorrect and should be deleted. Essentially, this step involves learning how to "correct" the errors made during the insertion phase. Specifically, the input to this step is the output from the insertion module, $y_2 = \varepsilon(y_1, t)$, where t

represents the actions predicted by the model in the token prediction step. Since the task is to decide whether each token in y_2 should be deleted, this step is essentially a binary classification task for each token, which can be represented as follows:

$$\mathcal{L}_\theta^{\text{del}} = -\sum_{\substack{y_i \in y_2 \\ d_i^* \in d^*}} \log \pi_\theta^{\text{del}}(d_i^* | i, y_2)$$

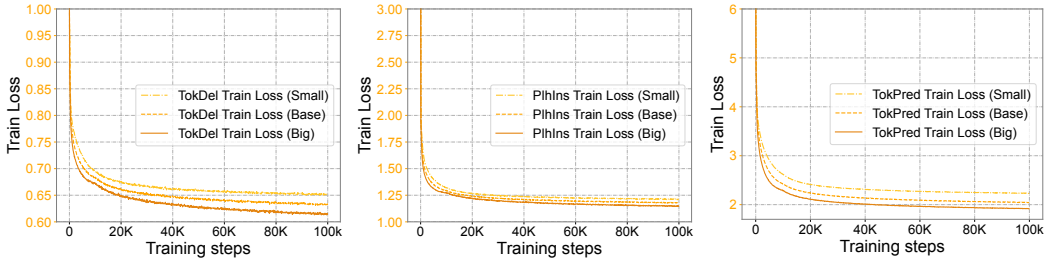
where d_i^* is the optimal delete action found by the expert that minimizes the Levenshtein distance to the target output y^* which is the SMILES of molecule M, and can be formalized as $d_i^* = \text{argmin}_d \mathcal{D}(y^*, \varepsilon(y_2, d))$, π_θ^{del} is LevT's deletion classifier.

Total Loss. Since the editing process of LevT consists of three steps—token deletion, placeholder insertion, and token prediction—the overall training objective of LevT is the sum of the training objectives for these three processes:

$$\mathcal{L}_\theta^{\text{LevT}} = \mathcal{L}_\theta^{\text{ins}} + \mathcal{L}_\theta^{\text{tok}} + \mathcal{L}_\theta^{\text{del}}$$

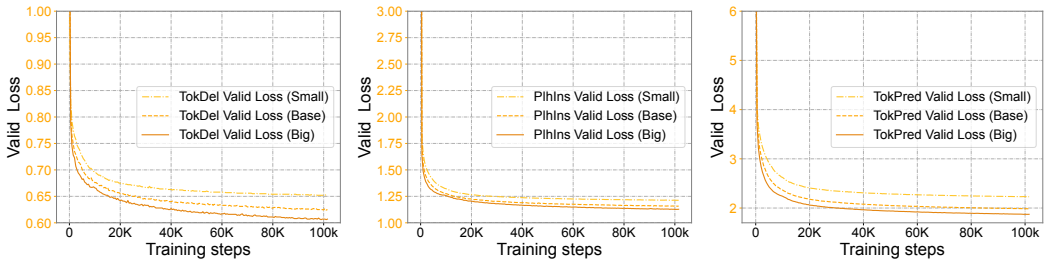
In summary, unlike MLM models, which provide training signals only for each `[MASK]` symbol in the input sequence, the LevT model offers training signals for every token in both the Placeholder Insertion and Token Deletion steps. This requires the model to determine whether each token in the input sequence should be deleted and whether new tokens should be inserted after each existing token, thus providing richer semantic information to the model.

B MORE TRAINING CURVES OF SMI-EDITOR



(a) Token Deletion (b) Placeholder Insertion (c) Token Prediction

Figure 7: **Training Loss Curves of Editing Operations.** We train SMI-EDITOR models of varying sizes and compare their loss curves during training for three different editing operations. As shown in the results, the loss for the token prediction process represented in Figure (c) is consistently the highest among the three type of losses, while the loss for token deletion is the lowest. Furthermore, as the model size increases, all three types of loss exhibit a stable downward trend.



(a) Token Deletion (b) Placeholder Insertion (c) Token Prediction

Figure 8: **Validation Loss Curves of Editing Operations.** Similar to the training loss curves, the validation loss for the token prediction process shown in Figure (c) consistently remains the highest among the three types, while the loss for token deletion is the lowest. Additionally, as the model size increases, the validation loss for all three editing operations exhibits a stable downward trend.

We present the changes in training and validation loss curves for SMI-EDITOR models of varying sizes during training. As shown in Figure 7 and Figure 8, both training and validation losses for the

three types of editing operations exhibit a stable downward trend as model scale increases. The loss from the token prediction process consistently constitutes the largest portion of the overall training loss. Interestingly, during the edit-based pre-training, the token prediction task is similar to that of MLM, as it involves predicting the real tokens corresponding to each placeholder token `[P]`, aiming to restore the complete target SMILES. However, unlike the results in Figure 2, the token prediction loss in the SMI-EDITOR pre-training does not show rapid saturation phenomenon in the early training stages. Even in the later training phases, the token prediction loss shows a steady decline. This further emphasizes the benefit of introducing fragment-level training signals; **by removing substructures and then asking the model to predict them instead of randomly masking tokens, we achieve a training task with better scalability.**

C HYPER-PARAMETERS FOR MODELS OF VARYING SCALES

In Table 4, we present the specific training hyperparameters for the models of different sizes (Big, Base, Small) used in this study. Notably, while the training objectives differ for the MLM and SMI-EDITOR models, all other model settings remain consistent, including the listed training hyperparameters and training datasets, to ensure the comparability of results.

Table 4: Hyper-parameters for pre-train models with different scales.

Model	Max Tokens	Layers	Attn Heads	Embed Dim	FFN Dim	Dropout	Num of Paras
Big	64K	9	12	768	2048	0.1	50.5M
Base	64K	6	8	512	2048	0.1	19.4M
Small	64K	3	8	512	1024	0	6.8M

D PERFORMANCE OF MLM MODELS ON DOWNSTREAM TASK

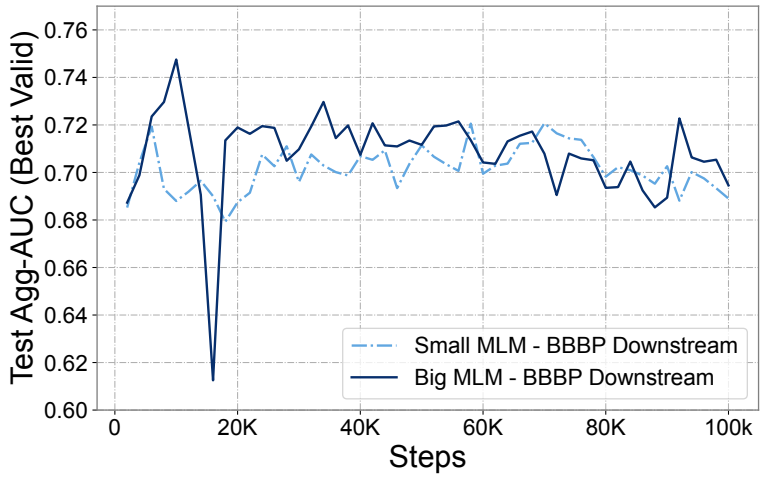


Figure 9: The performance of MLM models of different sizes and training steps on BBBP task.

We test the performance of MLM models (SMI-MLM) of different sizes and training steps on the BBBP task. To reduce variability and ensure accuracy, we evaluate each checkpoint on the downstream tasks five times and take the average of the results. As shown in Figure 9, increasing the model's scale does not consistently improve the model's performance on the downstream task and even in many cases, small model exhibits stronger performance. This indicates that the semantic information learned by the larger MLM model do not translate into better downstream task performance. Instead, the larger models exhibit greater variability in their performance compared to the small model. This results suggest that the scalability and stability of MLM models is very limited.

E PERFORMANCE OF SMI-EDITOR ON DOWNSTREAM TASK

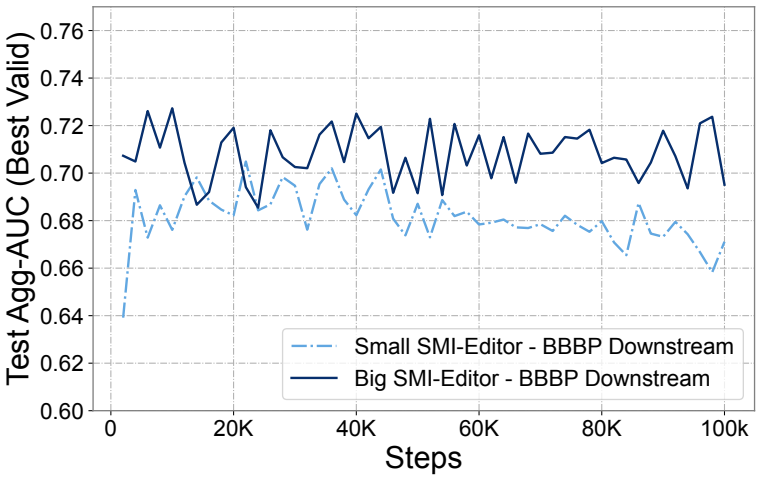


Figure 10: The performance of SMI-EDITOR of different sizes and training steps on BBBP task.

We also test the performance of SMI-EDITOR of different sizes and training steps on the BBBP task, and the results are shown in Figure 10. We also evaluate each checkpoint on the downstream tasks five times and take the average of the results to ensure accuracy. Compared to the performance of the MLM model (Figure 9), the larger SMI-EDITOR model (Big Model) consistently outperforms the smaller models (Small Model). As the number of training steps increases, the performance gap between the large and small models becomes increasingly significant. In contrast, larger MLM models do not show this trend, as different-sized MLM models exhibit similar performance on downstream tasks. Moreover, the larger MLM models exhibit greater performance fluctuations compared to the smaller MLM models. However, the larger SMI-EDITOR model demonstrates greater performance stability than the MLM model, as the larger SMI-EDITOR model does not exhibit increased performance fluctuations compared to the smaller SMI-EDITOR models. These results indicate that **the SMI-EDITOR model offers better training stability and model scalability than the MLM model**.

F HYPER-PARAMETER CONFIGURATION FOR PRE-TRAINING

We implement SMI-EDITOR using 12 stacked Transformer layers, each with 12 attention heads. The model dimension and feedforward dimension of each Transformer layer are 768 and 3072. The total number of SMI-EDITOR's parameters is 86.3M. We use Adam (Kingma & Ba, 2014) and polynomial learning rate scheduler to train SMI-EDITOR and set the learning rate 5e-4 warmup step 10K. The total training step is 120K and each batch has 64k tokens at maximum. We implement the SMI-EDITOR model using the Fairseq library and train SMI-EDITOR on four Tesla A40 GPU for about 2 days.

For more pre-training hyper-parameters, please refer to Table 5.

G HYPER-PARAMETER CONFIGURATION FOR FINE-TUNING

In different downstream task, we use different hyper-parameters. For detailed fine-tuning hyper-parameters, please refer to Table 6.

H DETAILS OF FINE-TUNING DATASETS

We perform a comprehensive set of experiments on the MoleculeNet(Wu et al., 2018) benchmark, focusing on the molecular property prediction task. MoleculeNet has emerged as one of the most

Table 5: SMI-EDITOR hyper-parameters for pre-training.

Hyper-parameters	Value
Learning rate	5e-4
LR scheduler	polynomial_decay
Warmup updates	10K
Max updates	120K
Max tokens	64k
FFN dropout	0.1
Attention dropout	0.1
Activation dropout	0
Num of layers	12
Num of attention heads	12
Encoder embedding dim	768
Encoder FFN dim	3072
Adam (β_1, β_2)	(0.9,0.98)
Fragments Drop ratio	0.15
Vocabulary size	369
Activation function	GELU
Weight Decay	0.0
Clip Norm	1.0

Table 6: SMI-EDITOR hyper-parameters for fine-tuning.

Tasks	Epochs	Batch size	Learning rate	Warmup Ratio	Dropout	Pooler-dropout
BACE	60	64	1e-4	0.06	0.1	0.2
BBBP	40	128	4e-4	0.06	0.1	0.1
TOX21	80	128	1e-4	0.06	0.1	0.1
SIDER	100	32	5e-4	0.4	0.1	0
MUV	40	128	2e-5	0.2	0.1	0.1
ClinTox	100	256	5e-5	0.1	0.1	0.5
ToxCast	80	64	1e-4	0.06	0.1	0.1

widely recognized and utilized benchmarks in the field of molecular property prediction, providing a standardized platform for evaluating machine learning models designed to predict various molecular properties. Its datasets encompass a broad range of molecular tasks, and address diverse scientific problems such as drug discovery, toxicity prediction and so on.

In this section, we provide a detailed summary of the statistics and fundamental characteristics of the MoleculeNet datasets we use in Table 7. This table offers information about the dataset sizes, task types, and compositions, providing readers with essential background information to better understand the experimental setup and subsequent analysis.

I PERFORMANCE OF SMI-EDITOR ON DEEPCHEM DATA

We re-evaluated the performance of SMI-EDITOR on various downstream tasks of MoleculeNet benchmark using the data splits provided by DeepChem[1]. Previously, our experiments were based on a different data split, which made it difficult to compare our model against others built on this dataset. Therefore, we re-tested SMI-EDITOR on DeepChem splits and included comparisons with more baseline models. Detailed results are presented in Table A1. As shown in Table A1, **SMI-EDITOR achieves significant performance gains over baseline models, reaching state-of-the-art levels with noticeable average performance improvements**. Below is a detailed analysis of these results:

- SMI-EDITOR outperforms models trained with various paradigms: On average, SMI-EDITOR surpasses molecular representation learning models like MolCLR and $\mathrm{DMP}_{\mathrm{TF}}$,

[1]https://github.com/deepchem/deepchem

Table 7: Summary information of the MoleculeNet benchmark datasets.

Dataset	Tasks	Task type	Molecules (train/valid/test)	Describe
BACE	1	Classification	1,210/151/151	Binding results of human BACE-1 inhibitors
BBBP	1	Classification	1,631/204/204	Blood-brain barrier penetration
ClinTox	2	Multi-label classification	1,182/148/148	Clinical trial toxicity and FDA approval status
Tox21	12	Multi-label classification	6,264/783/783	Qualitative toxicity measurements
ToxCast	617	Multi-label classification	6,860/858/858	Toxicology data based on in vitro screening
SIDER	27	Multi-label classification	1,141/143/143	Adverse drug reactions to the 27 systemic organs
MUV	17	Multi-label classification	74,469/9,309/9,309	A subset of PubChem BioAssay

which use contrastive pretraining, as well as models like ChemBerta and SMI-MLM, which use masked language modeling. It also outperforms autoregressive language models like Galactica and graph-based models like MolCLR, MGSSL, and MoMu. These results highlight the potential of SMILES language models.

- SMI-EDITOR achieves competitive performance with less training data: SMI-EDITOR outperforms $\mathrm{DMP}_{\mathrm{TF}}$, which is trained on over 100 million compounds, despite using only 19 million compounds for training. This demonstrates SMI-EDITOR's higher data efficiency, enabled by its ability to effectively leverage substructure information from SMILES sequences.

Table 8: Overall results on MoleculeNet datasets using DeepChem splits. ROC-AUC scores (higher is better) are reported for all tasks. The best results are **bolded**

Method	BBBP↑	Tox21↑	ClinTox↑	HIV↑	BACE↑	SIDER↑	Mean↑
GEM	72.4	78.1	90.1	80.6	85.6	67.2	79.0
ChemBerta	64.3	-	90.6	62.2	-	-	-
MolCLR	73.6	79.8	93.2	80.6	89.0	68.0	80.7
MGSSL	70.5	76.5	80.7	79.5	79.7	61.8	74.8
$\mathrm{DMP}_{\mathrm{TF}}$	78.1	78.8	95.0	81.0	89.3	69.2	81.9
Galactica	66.1	68.9	82.6	74.5	61.7	63.2	69.5
MoMu	70.5	75.6	79.9	76.2	77.1	60.5	73.3
SMI-MLM	89.4	76.2	90.6	79.8	86.6	66.5	81.5
SMI-EDITOR	**93.5**	**81.4**	**95.2**	**81.6**	**89.9**	**69.8**	**85.2**

J PERFORMANCE ADVANTAGES OF SMI-EDITOR OVER AUTO-REGRESSIVE MODELS

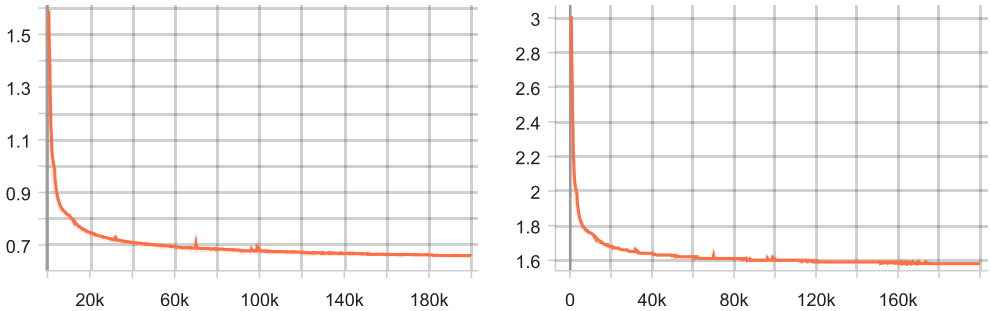
To comprehensively compare SMI-EDITOR with autoregressive models, we trained a decoder-only model with identical architecture and size to SMI-EDITOR using an autoregressive language modeling objective, referred to as SMI-GPT. We evaluated SMI-GPT's performance across several downstream tasks, with results shown in Table 9. The findings indicate that SMI-EDITOR can perform better than SMI-GPT. Below is an analysis of these results:

1. Implementation details for SMI-GPT(NT) and SMI-GPT(Emb):

SMI-GPT(NT): This approach uses next-token prediction for downstream classification tasks by appending a special token (e.g., $Label_0$, $Label_1$) at the end of each SMILES sequence to denote

Table 9: Results of SMI-EDITOR and SMI-GPT on MoleculeNet datasets using DeepChem splits. ROC-AUC scores (higher is better) are reported for all tasks The best results are **bolded**

Method	BBBP↑	Tox21↑	ClinTox↑	HIV↑	BACE↑	SIDER↑	Mean↑
SMI-GPT(NT)	88.5	74.3	88.9	68.8	76.2	63.7	76.7
SMI-GPT(Emb)	91.2	75.1	91.4	79.4	86.2	67.1	81.7
MoMu	70.5	75.6	79.9	76.2	77.1	60.5	73.3
SMI-MLM	89.4	76.2	90.6	79.8	86.6	66.5	81.5
SMI-EDITOR	**93.5**	**81.4**	**95.2**	**81.6**	**89.9**	**69.8**	**85.2**



(a) Train Loss (b) Train PPL

Figure 11: The training loss and perplexity (PPL) curves of the SMI-GPT model.

the classes of sample's label. The model learns to predict the correct label token during fine-tuning. **SMI-GPT(Emb)**: The representations of each token in the SMILES string extracted by the SMI-GPT model are processed using mean pooling. The resulting pooled representation is then fed into a classification head, which predicts the class of the SMILES.

2. Advantages of the encoder-only SMI-EDITOR architecture:

As shown in Table A2, SMI-EDITOR consistently outperforms SMI-GPT(Emb) and SMI-GPT(NT), highlighting its superior semantic learning capabilities. SMI-GPT(Emb) achieves better performance than SMI-GPT(NT), suggesting that pretraining-based feature transfer is preferable for molecular property prediction tasks. Therefore, the encoder-only pre-trained model is highly suitable for molecular property prediction tasks.

3. Rapid convergence in autoregressive LMs: we provide the training curve of the SMI-GPT model in Figure 11, which shows that the loss decreases rapidly during the early stages of training. Similarly, the perplexity also drops quickly, reaching approximately 1.6 at the 40K training step. By the end of training, the model's Perplexity falls below 1.6, which is significantly lower than the perplexity typically observed for GPT models trained on text data.

4. Why does this phenomenon occur?

For auto-regressive language models, each time a new token is generated, it receives all preceding tokens as prefix input. This means that when the model generates tokens at later positions, it has access to more comprehensive contextual information (i.e., a longer prefix and more complete sequence information). As a result, **the prediction difficulty for tokens in later positions is significantly reduced, allowing the model to converge more easily**. A key difference between SMI-EDITOR and SMI-GPT is that in SMI-EDITOR, each discarded token is predicted independently, with equal importance assigned to the prediction of each token. This enables SMI-EDITOR to better capture the complete semantic information encoded in the tokens.

In summary, compared to LLMs on text data, GPT models on SMILES data converge significantly faster and achieve much lower perplexity. This indicates that SMILES data is inherently easier to fit than text. Therefore, it is crucial to design effective methods to extract richer semantic information from SMILES. **SMI-EDITOR represents a meaningful and successful exploration in**

this direction, highlighting the importance of leveraging substructural fragment information within SMILES data.

K PERFORMANCE OF SMI-EDITOR WITH FRAGMENT CORRECTION

Training SMI-EDITOR to correct errors and remove extraneous components did not improve performance: We implemented a version of SMI-EDITOR that learns to correct erroneous functional groups and remove extraneous substructures, referred to as SMI-EDITOR-Cor. However, SMI-EDITOR-Cor did not outperform the original SMI-EDITOR on downstream tasks. Considering the increased complexity and training cost of SMI-EDITOR-Cor (due to longer input sequences), we focused on SMI-EDITOR in the submitted draft. Table 12 below compares the performance of SMI-EDITOR and SMI-EDITOR-Cor, showing that their performance is similar, demonstrating the limited benefit of incorporating these tasks.

Analysis of SMI-EDITOR-Cor's performance: We attribute SMI-EDITOR-Cor's lack of improvement to the following reasons:

- **Correcting errors and removing extraneous components provide limited additional training signals:** SMI-EDITOR's training comprises two major steps: deletion and insertion. During deletion, erroneous functional groups and extraneous substructures are removed, while the insertion step involves learning to recover the correct tokens in the appropriate positions. Thus adding erroneous functional groups or extraneous substructures affects only the deletion step, which is a simpler task providing limited information. Moreover, as shown in Table 3 of the main text, ablating the token deletion (TokDel) step has minimal performance impact.
- **Identifying erroneous functional groups and extraneous structures is too simple for the model:** SMI-EDITOR-Cor constructs erroneous inputs through random substitutions, often resulting in chemically invalid SMILES that are easy for the model to identify. Consequently, the simplicity of the training task limits further performance improvement.

Table 10: Performance comparison between SMI-EDITOR-Cor and SMI-EDITOR.

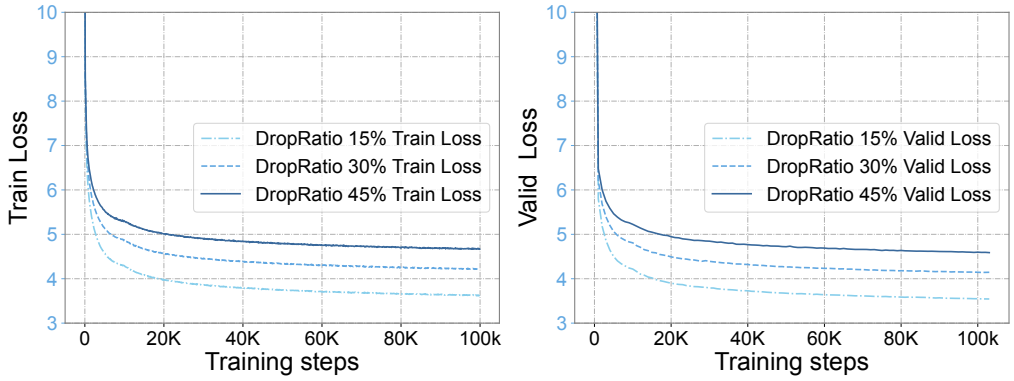
Method	BACE↑	BBBP↑	SIDER↑	Tox21↑	ToxCast↑	Mean↑
SMI-EDITOR-Cor	80.6	77.1	62.2	76.8	68.0	72.9
SMI-EDITOR	80.3	77.4	63.0	77.1	67.4	73.0

L HOW THE FRAGMENT DROP RATIO AFFECT SMI-EDITOR

To investigate the impact of the fragment drop ratio on SMI-EDITOR, we trained SMI-EDITOR models with different drop ratios (15%, 30%, 45%) and analyzed their training curves and downstream task performance. The results indicate that increasing the drop ratio significantly raises training loss for SMI-EDITOR, suggesting that its pretraining task is more challenging than MLM. Below are the detailed findings:

1. Impact on SMI-EDITOR's convergence: We plotted the training and validation loss curves for SMI-EDITOR with varying drop ratios in Figure 12. **The results show that as the drop ratio increases, both training and validation losses rise significantly.** Compared to Figure 2c of the paper, the loss increase for SMI-EDITOR is more pronounced than for MLM, confirming that SMI-EDITOR's task is inherently more challenging.

2. Impact on downstream task performance: We evaluated the performance of SMI-EDITOR and MLM models with varying drop or mask ratios. The results are summarized in Table 11. From Table 11, it can be observed that as the mask ratio increases, the average performance of the SMI-MLMmodel shows no significant change, while the performance of the SMI-EDITOR model declines as the drop ratio increases. This indicates that SMI-EDITOR represents a more challenging training task.



(a) Train Loss (b) Valid Loss

Figure 12: The training loss and valid loss of the SMI-EDITOR with different fragment drop ratios.

Here is a more detailed explanation:

- SMI-EDITOR discards chemically meaningful substructures that often serve as standalone semantic units. This also makes predicting the discarded fragments more difficult than predicting individual masked tokens. Dropping more substructures severely disrupts the molecular structure, making it harder for the model to reconstruct the original molecule.

- MLM, on the other hand, randomly masks tokens in SMILES sequences. Since SMILES tokens often represent individual atoms or bonds, masking does not typically disrupt the molecular semantics significantly. For instance, masking one or two atoms of a functional group like $-$COOH still leaves enough contextual information to reconstruct it. Additionally, the probability of masking an entire functional group is low due to MLM's token-based masking mechanism. This explains why MLM performance is less sensitive to mask ratio increases, as also reflected in Figure 2c of the paper: Different Mask Ratios Cannot Alleviate Rapid Saturation.

Table 11: Performance of SMI-EDITOR and SMI-MLMwith different drop or mask ratios on downstream tasks.

Method	BACE↑	BBBP↑	SIDER↑	Tox21↑	ToxCast↑	Mean↑
SMI-MLM(15%)	77.8	68.6	61.2	75.1	64.9	69.5
SMI-MLM(30%)	78.3	70.2	58.2	76.0	63.7	69.3
SMI-MLM(45%)	78.4	66.1	59.3	76.4	65.5	69.1
SMI-EDITOR(15%)	80.3	**77.4**	**63.0**	77.1	**67.4**	**73.0**
SMI-EDITOR(30%)	**81.6**	73.3	59.6	77.0	66.8	71.7
SMI-EDITOR(45%)	79.3	72.2	61.1	**77.8**	67.1	71.5

M PERFORMANCE OF SMI-EDITOR ON MOLECULAR PROPERTY REGRESSION TASKS

We evaluated the model's performance on three molecular property regression tasks, as shown in Table 12. SMI-EDITOR achieved the best performance compared to baseline models and significantly outperformed the MLM model.

Table 12: Performance of SMI-EDITOR on molecular property regression tasks.

Method	ESOL↓	FreeSolv↓	Lipo↓
MPNN	0.58	1.150	0.7190
DMP_{TF}	0.700	-	-
A-FP	0.503	0.736	0.578
SMI-MLM	0.576	0.709	0.642
SMI-EDITOR	**0.362**	**0.524**	**0.565**

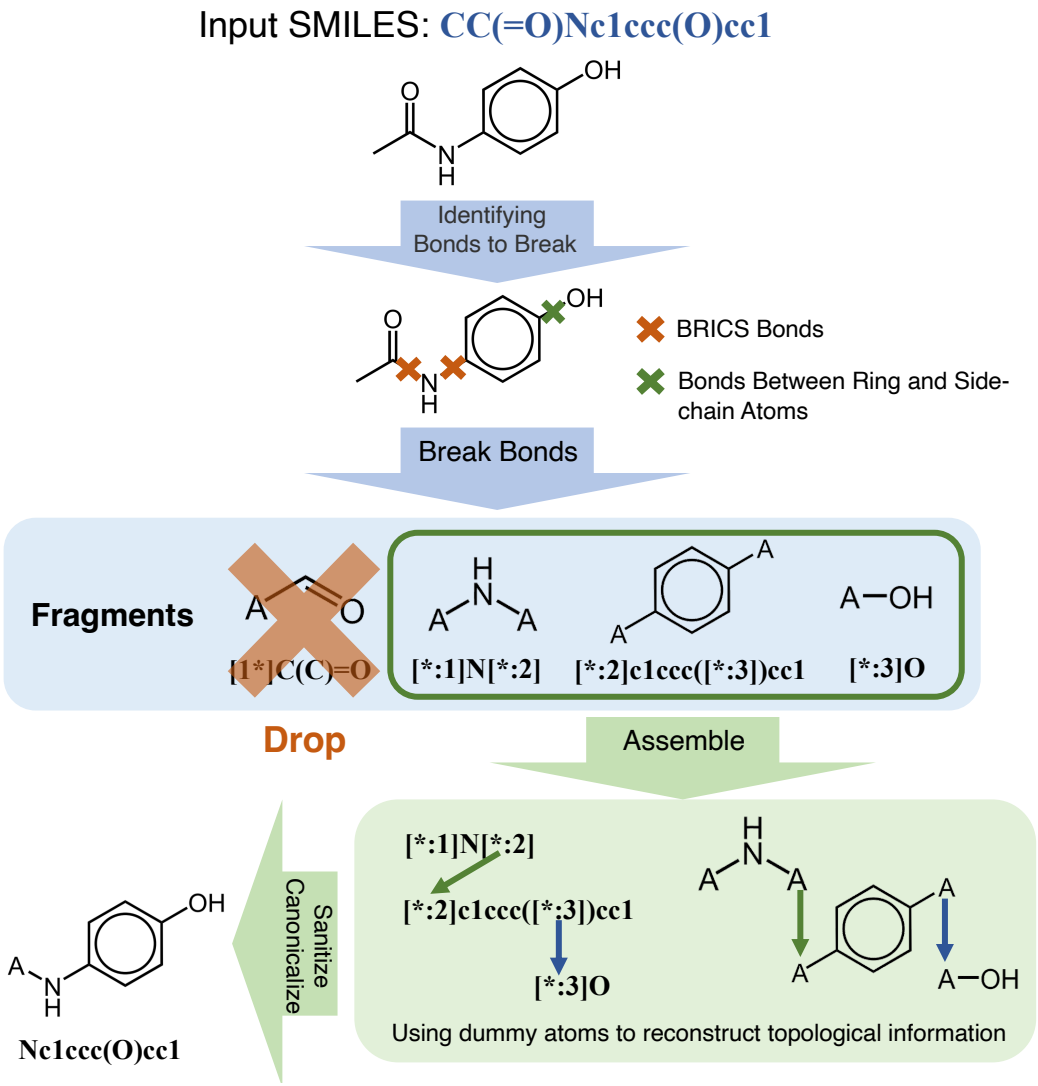


Figure 13: An Example Workflow of Molecule Fragmentation and Assemble with Paracetamol

N A CASE STUDY FOR FRAGMENTS ASSEMBLE

O THE SCALABILITY OF SMI-EDITOR

We added results showing the performance of SMI-EDITOR and SMI-MLMmodels of varying sizes on downstream tasks, which further demonstrate SMI-EDITOR's strong scalability. These results are shown in Table 14. It is evident that while increasing model size has minimal impact on MLM

models, larger SMI-EDITOR models show more consistent performance gains. **This confirms the claim that SMI-EDITOR has better scalability compared to MLM models.**

Table 13: Performance of SMI-EDITOR and SMI-MLMwith different scales on downstream tasks.

Method	BACE↑	BBBP↑	SIDER↑	Tox21↑	ToxCast↑	Mean↑
SMI-MLM(Small)	76.8	69.6	60.5	75.3	64.2	69.2
SMI-MLM(Base)	76.6	69.3	59.9	75.3	64.4	69.1
SMI-MLM(Big)	77.4	68.7	60.8	75.1	65.3	69.4
SMI-EDITOR(Small)	78.3	72.6	59.4	75.6	65.1	70.2
SMI-EDITOR(Base)	79.2	73.2	**61.0**	75.7	65.8	71.0
SMI-EDITOR(Big)	**79.3**	**74.2**	60.9	**76.7**	**66.4**	**71.5**

P THE TRAINING COST OF SMI-EDITOR

We measured that the training cost of SMI-EDITOR is approximately three times that of MLM models (SMI-MLM) for the same model size, training hyperparameters, and data. However, **the training cost of SMI-EDITOR remains acceptabl**e. To better analyze the impact of training cost, we trained an MLM model with equivalent computational cost (SMI-MLM(More)). Results showed that SMI-MLM(More) performed worse than the original SMI-MLMand significantly lagged behind SMI-EDITOR, highlighting that merely increasing MLM training cost does not yield better results. Below is a detailed analysis:

1. Reasons for higher training cost in SMI-EDITOR: SMI-EDITOR requires computing expert actions (using a computationally expensive dynamic programming algorithm) and modeling three different editing operations, which introduces additional overhead.

2. Acceptable training cost: Training SMI-EDITOR on a dataset with 19M compounds using four RTX 3090 GPUs took approximately 24.6 hours. Scaling SMI-EDITOR to larger datasets (e.g., 100M+ compounds) is feasible, demonstrating its potential for broader applications.

3. SMI-EDITOR performs better under the same training cost with MLM: We trained SMI-MLM(More) with the same computational cost as SMI-EDITOR by increasing its training steps from 120K to 360K. Table 14 shows that SMI-MLM(More) performs worse than the SMI-EDITOR and original SMI-MLM. This is due to rapid saturation issues in MLM training on SMILES data. **This also indicates that the speed of model training is not the most important factor; what matters more is whether the model can efficiently extract high-quality semantic representations**. This highlights the importance of designing more powerful training schemes like SMI-EDITOR to effectively extract meaningful information from SMILES.

4. Higher performance ceiling for SMI-EDITOR: Although the inclusion of Experts slows down the training speed of the SMI-EDITOR model, it also enriches the semantic information the model learns. This gives SMI-EDITOR greater scalability and a higher performance ceiling compared to SMI-MLM. As shown in Table D1, SMI-EDITOR benefits more from increased model size and training cost. This makes SMI-EDITOR a better choice when given the same training budget.

Table 14: Performance of SMI-EDITOR and SMI-MLMwith different scales on downstream tasks.

Method	BACE↑	BBBP↑	SIDER↑	Tox21↑	ToxCast↑	Mean↑
SMI-MLM(More)	74.3	66.2	49.5	73.3	62.3	65.1
SMI-MLM	77.8	68.6	61.2	75.1	64.9	69.5
SMI-EDITOR	**80.3**	**77.4**	**63.0**	**77.1**	**67.4**	**73.0**

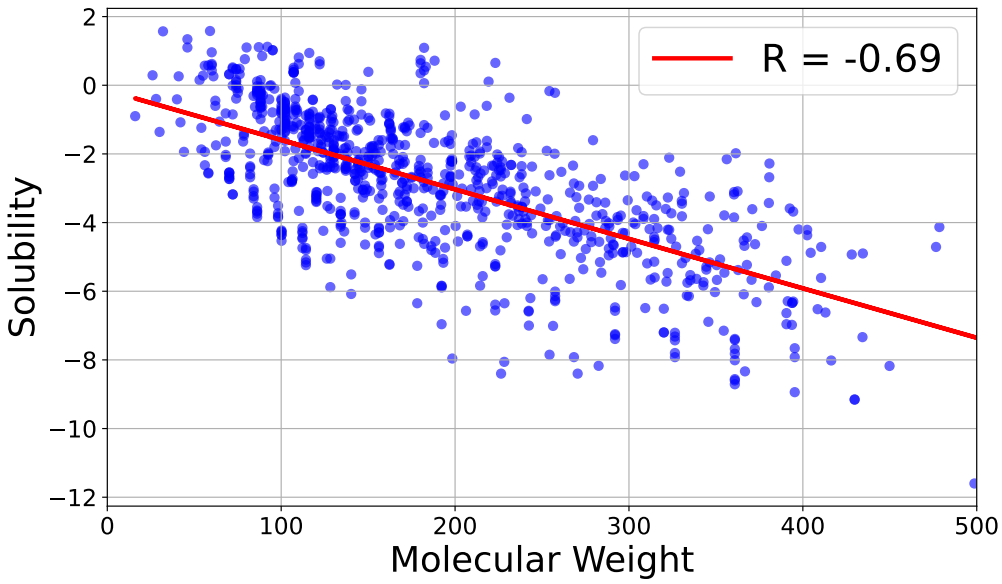


Figure 14: The relationship between molecular weight and solubility in the ESOL training set

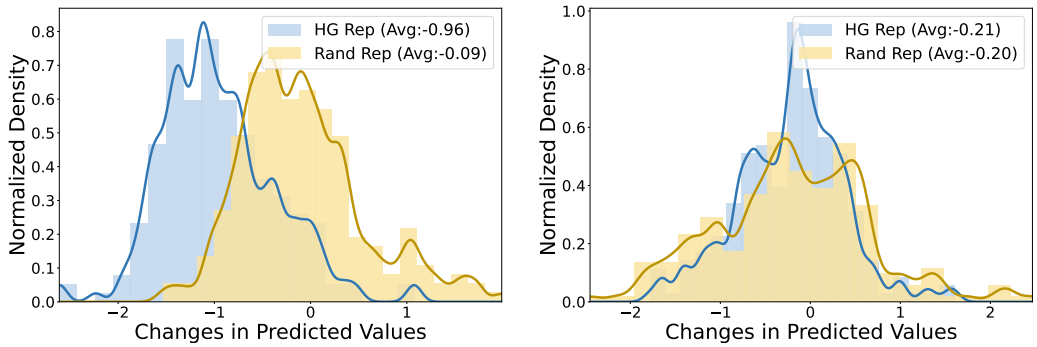
Q A MORE DETAILED ANALYSIS OF THE MODEL'S SUBSTRUCTURE MODELING CAPABILITY.

The observed trends for the FreeSolv dataset are fully consistent with our expectations and align with the definition of its physical properties. On the other hand, the performance on the ESOL dataset is influenced by additional factors such as molecular weight. We also designed more analytical experiments to further investigate the behavior of the SMI-EDITOR model, and the results demonstrate that the model's behavior aligns with expectations. Detailed explanations are as follows.

For the FreeSolv dataset, the observed trends align with its physical property definitions. FreeSolv reflects the hydration free energy of compounds, defined as the free energy change when a compound transitions from a non-dissolved state to a dissolved state. When hydrophilic groups in a molecule are reduced, the change in hydration free energy increases, leading to higher hydration free energy. Thus, when we remove hydrophilic groups from the molecule, the model predicts an increase in hydration free energy, consistent with the trend observed in Figure 5(b), which matches our expectations.

For the ESOL task, the model predictions are significantly influenced by molecular weight. The ESOL dataset reflects compound solubility, which is strongly negatively correlated with molecular weight: the larger the molecular weight, the lower the solubility. We plotted a scatter diagram (Figure 14) showing the relationship between molecular weight and solubility in the ESOL training set. A clear negative correlation with a coefficient of $R = -0.69$ is observed. Consequently, when functional groups or atoms are removed from a molecule, its molecular weight decreases, leading the model to predict an increase in solubility. **This explains why, in Figure 5(a), the model predicts increased solubility regardless of whether hydrophilic groups or random groups are removed**. The increase is more significant with random deletions, demonstrating the model's ability to distinguish between hydrophilic group deletions and random deletions.

To eliminate the influence of molecular weight, we designed a hydrophilic group replacement scheme (HG Rep). We replaced all hydrophilic groups in a molecule with non-hydrophilic groups of similar molecular weight (e.g., methyl, ethyl, propyl) and compared this hydrophilic group replacement scheme (HG Rep) with a random group replacement scheme (Rand Rep), where random groups were replaced with others of similar weight. The results, shown in Figure **??**, reveal that SMI-EDITOR effectively distinguishes between HG Rep and Rand Rep, demonstrating its ability to



(a) SMI-EDITOR (b) SMI-MLM

Figure 15: **Substructure Semantics Modeling on ESOL Dataset.** We compared the effects of two molecular perturbation methods on the SMI-EDITOR's and SMI-MLM's predictions of hydrophilicity. Figure (a) show that the impact of replacing hydrophilic groups (HG Rep) and randomly replacing atoms (Rand Rep) on the model's predictions differs significantly, both in the average change in prediction values and their distributions.

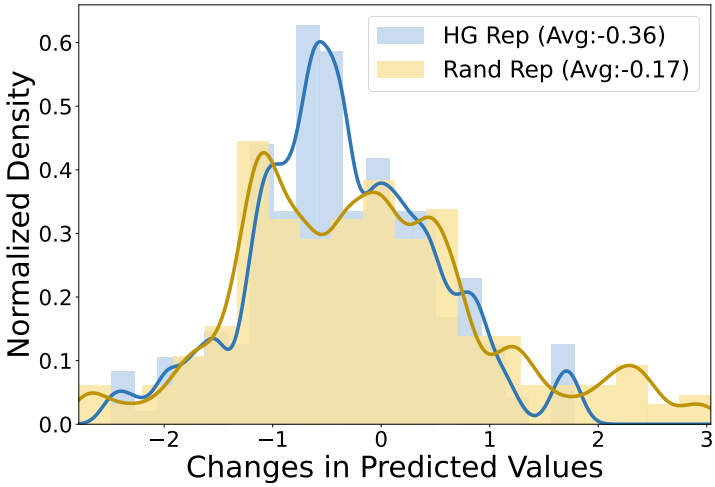


Figure 16: Substructure Semantics Modeling on ESOL Dataset of Auto-regressive LM.

model key molecular group semantics. It also correctly predicts that replacing hydrophilic groups reduces molecular solubility.

Furthermore, we plotted the distribution of predicted changes for MLM models and Auto-regressive language models (Auto-regressive LM) before and after these replacement operations in Figure 16. The results show that these models perform significantly worse than the SMI-EDITOR in distinguishing between random replacements and hydrophilic group replacements. This further highlights the superiority of the SMI-EDITOR in modeling the semantics of molecular substructures.

R SMI-EDITOR'S PERFORMANCE ON RETROSYNTHESIS PREDICTION TASKS

Considering that the original SMI-EDITOR is an encoder-only model and cannot be directly applied to generative tasks, we further pretrained a model based on an encoder-decoder architecture, referred to as **SMI-EDITOR-Gen**. We tested its performance on the retrosynthesis prediction task, where it achieved state-of-the-art results. Below is a detailed discussion:

Model Details of SMI-EDITOR-Gen. SMI-EDITOR-Gen adopts a transformer architecture with a base-sized scale (Vaswani et al., 2017) and the specific model size details are provided in Table

15. During pretraining, the input to the encoder consists of SMILES strings with missing molecular fragments, while the decoder's pretraining task is to reconstruct the original SMILES. Following approaches commonly used in machine translation (Vaswani et al., 2017) , the features extracted by the encoder are passed to the decoder through encoder-decoder attention (Vaswani et al., 2017) . Compared to SMI-EDITOR, the most significant difference is that the encoder-decoder architecture enables SMI-EDITOR-Gen to perform sequence-to-sequence generative tasks**, allowing us to explore the model's capabilities in such tasks.

SMI-EDITOR-Gen Exhibits Strong Performance in Retrosynthesis Prediction Tasks. Following the experimental setup of EditRetro(Han et al., 2024), we evaluated SMI-EDITOR-Gen on the retrosynthesis prediction task. During fine-tuning, we applied the same fine-tuning strategies and data augmentation techniques as EditRetro. The experimental results, shown in Table D4, demonstrate that SMI-EDITOR-Gen achieved strong performance on the USPTO-50K dataset. This validates that the pretraining approach proposed by SMI-EDITOR also exhibits excellent performance and great potential in generative tasks.

Table 15: Top-k exact match accuracy of SMI-EDITOR and baselines on the USPTO-50k dataset.

	Top-1 ↑	Top-3 ↑	Top-5 ↑	Top-10 ↑
RetroPrime	51.4%	70.8%	74.0%	76.1%
Transformer	42.4%	58.6%	63.8%	67.7%
SCROP	43.7%	60.0%	65.2%	68.7%
MEGAN	48.1%	70.7%	78.4%	86.1%
GTA	51.1%	67.6%	74.8%	81.6%
Retroformer	53.2%	71.1%	76.6%	82.1%
Graph2Edits	55.1%	77.3%	83.4%	89.4%
R-SMILE	56.3%	79.2%	86.2%	**91.0%**
EditRetro	60.8%	80.6%	86.0%	90.3%
SMI-EDITOR	**61.2%**	**80.9%**	**86.4%**	89.7%

S SPE TOKENIZER DOES NOT IMPROVE SMILES MLM PERFORMANCE

We trained a SMILES MLM model with SPE tokenizer, **SMI-MLM(SPE)**, using the same architecture and hyperparameters as SMI-EDITOR, and evaluated it on multiple tasks. As shown in Table 16, SMI-MLM(SPE) performs similarly to SMI-MLMand significantly worse than SMI-EDITOR. This demonstrates that introducing SPE cannot replicate the effectiveness of SMI-EDITOR. The reasons are:

- **Limited Fragment Diversity**: SPE relies on a fixed vocabulary, limiting the diversity of fragment-level information it can capture. In contrast, SMI-EDITOR dynamically fragments molecules using the BRICS algorithm, capturing a wider variety of molecular substructures.
- **Topology Information Leakage**: SPE-based models retain token position information, which is tied to molecular topology in SMILES, making the prediction task easier but less effective.
- **Lack of Chemical Context**: SMI-EDITOR fragments molecules based on chemical rules, allowing it to capture substructure information more relevant to molecular properties, unlike SPE, which relies on character pair frequencies.
- **Superior Performance with Fragment-Level Supervision**: A MLM model trained with fragment-level supervision, SMI-MLM(Frag), outperforms SMI-MLM(SPE), as shown in Table 16. This validates the effectiveness of SMI-EDITOR's training approach.

T MASKED SPAN LANGUAGE MODEL DOES NOT IMPROVE SMILES LM PERFORMANCE

1. To highlight the differences between SMI-EDITOR and MSLMs, we trained a SMILES model using MSLM, which randomly masks continuous sequences in SMILES and predicts the missing

Table 16: Performance comparison of MLM models with different pretraining strategies.

	BACE↑	BBBP↑	SIDER↑	Tox21↑	ToxCast↑	Mean↑
SMI-MLM	77.8	68.6	61.2	75.1	64.9	69.5
SMI-MLM(SPE)	76.7	71.1	59.3	74.7	65.3	69.4
SMI-MLM(SPAN)	78.6	67.2	59.4	76.1	62.3	68.7
SMI-MLM(Frag)	79.4	73.3	62.1	74.0	64.8	70.7
SMI-EDITOR	**80.3**	**77.4**	**63.0**	**77.1**	**67.4**	**73.0**

parts (similar to SpanBERT(Joshi et al., 2020)). This model, referred to as **SMI-MLM(SPAN)**, shows performance comparable to SMI-MLMbut significantly worse than SMI-EDITOR (see Table 16). This further demonstrates SMI-EDITOR's advantages over traditional MSLMs. Reasons for Poor Performance of Traditional MSLMs:

- **Differences between Text Data and SMILES Data.** Unlike text, molecular data has complex topological structures. In text, adjacent tokens often have strong semantic relevance, and continuous spans convey related information, making span masking effective for learning local semantics. However, **SMILES lacks such locality**; a single functional group may not appear contiguous, and adjacent tokens may lack strong relevance. For example, aromatic rings with multiple substituents often appear discontinuous in SMILES (we provide a specific case **CASE1**). This limits the effectiveness of applying span masking directly to SMILES data.
- **Traditional MSLM (e.g., T5(Raffel et al., 2020)) and SMI-EDITOR Have Different Implementations; Traditional MSLM is Unsuitable for SMILES Data.** Text data's semantic continuity enables models like T5 to use random span masking, where continuous text segments are masked for prediction. In contrast, SMILES lacks this continuity, so SMI-EDITOR uses a fragmentation algorithm to split molecules into chemically meaningful fragments. The model predicts missing fragments, which may not correspond to continuous SMILES segments. Unlike traditional MSLM, SMI-EDITOR focuses on masking chemically significant fragments, a key difference in its design.
- **Better Performance of SMI-MLM(Frag).** The improved performance of SMI-MLM(Frag) over SMI-MLM(SPAN) highlights the superiority of SMI-EDITOR's fragment-level supervision. While SMI-MLM(SPAN) uses the traditional MSLM approach, SMI-MLM(Frag) incorporates supervision signals similar to SMI-EDITOR, enabling it to better capture molecular substructure information.

CASE1 When does SMILES exhibit discontinuity: SMILES is a linearized representation of graph-structured molecules, which inherently causes discrepancies between molecular topology and sequence-level representation. For example, when a ring contains multiple substituents, its representation in SMILES often becomes discontinuous. Consider Glibenclamide, a drug used for diabetes treatment, with the canonical SMILES: COc1ccc(Cl)cc1C(=O)NCC**c2ccc**(S(=O)(=O)NC(=O)NC3CCCCC3)**cc2**. Here, the bolded atoms originate from the same aromatic ring, but due to the multiple substituents, this ring is represented discontinuously in SMILES. Additionally, the aromatic carbon **cc2** is adjacent to CCC3 atoms from a distant cycloalkane ring. Such discontinuities are common in SMILES and adversely affect Masked Span Language Models.

U COMPARISON BETWEEN SMI-EDITOR AND CONTRASTIVE LEARNING

Similarities: Both contrastive learning and SMI-Editor aim to learn alignment.

- **Contrastive learning aligns representations of different views.** The core idea of contrastive learning is to bring the representations of different views of the same sample (positive pairs) closer while pushing representations of different samples (negative pairs) apart. Essentially, this process learns the correct alignment between views of the same sample.

- **SMI-Editor aligns representations of missing substructures and contexts.** As Fu et al. (2022) noted, MLM models align the representations of contexts and missing words during training. Similarly, SMI-Editor aligns the representations of missing substructures and their contexts. For example, given the input Nc1ccc(O)cc1, the model need to predict the complete molecule CC(=O)Nc1ccc(O)cc1. SMI-Editor can effectively align the representation of the missing fragment CC(=O) with the context Nc1ccc(O)cc‘ through this process.

Differences: The alignment targets differ between the two paradigms.

- **Contrastive learning focuses on global information**: The representations to be aligned often correspond to different augmented views of the same molecule, such as through atom deletion, bond deletion, or subgraph deletion. These views typically preserve the molecule's overall structure and thus contain global information.
- **SMI-Editor emphasizes aligning local substructure information with global context**: In SMI-Editor, the context typically corresponds to the molecule's backbone, representing global information, while the missing substructures contain local information.
- **SMI-Editor is more sensitive to local structure information**: By aligning local substructures with global context, SMI-Editor learns finer-grained semantics from SMILES data, making it better suited to capturing detailed molecular information than contrastive learning.

V K-FOLD CROSS-VALIDATION OF THE SMI-EDITOR MODEL.

Using a 5-fold setup, we evaluated SMI-EDITOR's performance on the training sets of BACE, BBBP, SIDER, Tox21, and ToxCast. The results are shown in Table 17. These results demonstrate that SMI-EDITOR exhibits strong performance and stability across downstream tasks.

Implementation Details: Each dataset was evenly divided into five parts. In each run, one part was selected as the validation set, while the remaining four parts were used as the training set. The model was trained and evaluated on the validation set. This process was repeated five times to complete all runs.

Table 17: 5-fold cross-validation results of the SMI-Editor model.

	BACE↑	BBBP↑	SIDER↑	Tox21↑	ToxCast↑
Run 1	91.92	97.64	62.59	83.69	75.83
Run 2	91.86	96.27	66.89	84.09	73.31
Run 3	90.82	98.53	62.60	84.87	73.52
Run 4	91.13	98.77	63.32	83.95	74.60
Run 5	90.68	97.84	63.50	85.83	75.51
Mean	**91.28**	**97.81**	**63.78**	**84.48**	**74.55**
Std	**0.58**	**0.97**	**1.78**	**0.87**	**1.13**