Disentangled Graph LLM for Molecule Graph Editing under Distribution Shifts

Yang Yao
DCST, Tsinghua University
Beijing, China
yaoyang21@mails.tsinghua.edu.cn

Xin Wang*
DCST, BNRist, Tsinghua University
Beijing, China
xin_wang@tsinghua.edu.cn

Yuan Meng
DCST, Tsinghua University
Beijing, China
yuanmeng@tsinghua.edu.cn

Zeyang Zhang
DCST, Tsinghua University
Beijing, China
zy-zhang20@tsinghua.org.cn

Hong Mei
Peking University
Beijing, China
meih@pku.edu.cn

Wenwu Zhu*
DCST, BNRist, Tsinghua University
Beijing, China
wwzhu@tsinghua.edu.cn

Abstract

Molecule graph editing has become a powerful paradigm for optimizing chemical compounds in drug discovery. Existing methods overlook the *invariant* structure-property relationships, and rely on *variable* correlations that shift across different instructions, thereby failing to generalize to out-of-distribution (O.O.D.) scenarios. To overcome the weakness of existing work, in this paper we propose to capture and utilize the invariant factors in order to achieve generalizable molecule graph editing under distribution shifts. However, this problem remains challenging, given that the invariant and variant factors are deeply entangled within the editing models. To tackle this challenge, we propose **MoFE**, a disentangled graph large language model for molecule graph editing that handles editing instructions under distribution shifts via disentangling invariant factors that govern editing-relevant properties. Specifically, we propose a *disentangled graph projector with invariance loss* that encodes molecular graphs into disentangled latent factors, with an invariance loss that ensures consistency across paraphrased prompts with the same objective. Then, we enhance the LLM with a *factor-aware LoRA mixture-of-experts*, where each expert is associated with a distinct latent factor. Additionally, we introduce a *factor disentanglement loss weighting* strategy that adaptively assigns higher weights to expert-factor pairs that perform well on relevant editing tasks. The proposed **MoFE** model promotes joint disentanglement between experts and latent factors, reinforcing their alignment and preventing collapse. Experiments on a representative benchmark demonstrate that MoFE is able to achieve superior O.O.D. generalization performance in molecule graph editing.

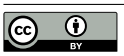
CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Large language model; Molecule edit; Molecule graph

*Corresponding Authors.



WWW '26, Dubai, United Arab Emirates.

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792464

ACM Reference Format:
Yang Yao, Xin Wang, Yuan Meng, Zeyang Zhang, Hong Mei, and Wenwu Zhu. 2026. Disentangled Graph LLM for Molecule Graph Editing under Distribution Shifts. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3774904.3792464

1 Introduction

Molecule graphs, where atoms and bonds form structured, heterogeneous relations, represent a critical class of web-related graphs that are increasingly analyzed, queried, and optimized through web-based platforms. With the rapid growth of open molecular repositories such as PubChem, ChEMBL, and MoleculeNet, molecule graph reasoning has become a core component of web-based drug discovery pipelines, enabling large-scale querying, representation learning, and property prediction. In this emerging landscape, molecule graph editing [13–15], the task of modifying molecular structures according to natural language instructions, offers a powerful paradigm for controllable compound optimization and interactive molecular design [21, 23].

Recent studies [7, 15] explored the integration of large language models (LLMs) [1, 6, 9, 24] into molecule editing workflows. Owing to their capacity to interpret and execute complex natural-language instructions, LLMs provide a promising interface for controllable and versatile molecule editing. Current LLM-based molecule editing methods [7, 15] heavily rely on the in-distribution (I.D.) assumption, i.e., the training and testing data are independently drawn from an identical distribution. However, distribution shifts of molecule-instruction data frequently occur in real world scenarios. For example, instructions may cover unseen molecule graph properties, and instruction syntax differences can create new molecule-instruction combinations. Existing approaches overlook the *invariant* structure-property-semantic patterns and tend to rely on correlations that are *variant* across different instructions, resulting in the failure of generalization to unseen molecule graph editing scenarios.

To tackle this issue, we study the problem of molecule graph editing in out-of-distribution (O.O.D.) scenarios by discovering and utilizing *invariant patterns*, namely structural and property-based factors that remain stable across distribution shifts. This problem poses several key challenges: (1) How can we extract editing-relevant, instruction-invariant latent factors from molecular graphs that reflect underlying structure–property relationships? (2) How can these factors be coupled with language representations so that

instructions influence molecular edits through stable structural semantics rather than surface patterns? (3) How can we ensure these factors interact in a complementary yet non-interfering manner during model training, enabling modular specialization without collapse?

To address these challenges, we propose **MoFE**, a molecule graph editing LLM designed for robust generalization across diverse editing instructions. The proposed **MoFE** model extracts latent factors from graph structures and performs modular reasoning through a mixture-of-experts architecture. Specifically, we propose a *disentangled graph projector with invariance loss* that extracts editing-relevant latent factors from molecular graphs and aggregates them into graph tokens. These tokens are incorporated into the LLM input, while an invariance loss ensures consistency across paraphrased prompts describing the same editing objective. Building on this foundation, we enhance the LLM with a *factor-aware LoRA mixture-of-experts*, where each expert is conditioned on a distinct latent factor. Additionally, to promote stable and specialized expert behavior, we introduce a *factor disentanglement loss weighting* strategy that prioritizes training on data most relevant to each expert–factor pair, thereby encouraging targeted optimization and reducing interference across experts. Experiments show that MoFE significantly outperforms baselines in O.O.D. settings, demonstrating its effectiveness in learning generalizable molecule editing behavior under distribution shifts.

Our contributions are summarized as follows:

- We are the first to study the problem of molecule graph editing under distribution shifts, to the best of our knowledge.
- We propose a disentangled graph LLM for O.O.D. molecule graph editing (**MoFE**), which explicitly disentangles molecular structures and aligns them with instruction semantics.
- We develop a modular architecture that combines a disentangled graph projector for extracting stable editing-relevant factors, a factor-aware LoRA mixture-of-experts for factor-specific reasoning, and a loss weighting strategy that promotes expert specialization and training stability.
- We conduct extensive experiments on O.O.D. molecule editing tasks, demonstrating that MoFE achieves significant improvements over state-of-the-art baselines.

2 Problem Formulation and Notations

This section formalizes the problem of molecule graph editing. We first introduce its general definition and objective, and then extend it to the O.O.D. setting, which motivates our approach.

Molecule Graph Editing. Let $\mathcal{G}$ denote the space of valid molecular graphs, and $\mathcal{X}$ the space of editing instructions. Given an input molecule graph $G^{\text{src}} \in \mathcal{G}$ and an instruction $X \in \mathcal{X}$, the molecule graph editing task aims to output a molecule graph $G^{\text{tgt}} \in \mathcal{G}$, that (1) semantically aligns with the intent described by X, (2) satisfies chemical validity, and (3) introduces minimal structural changes to G^{src}. Formally, the editing process is modeled as a parameterized mapping:

$$G^{\text{tgt}} = \mathcal{F}_\theta(G^{\text{src}}, X), \tag{1}$$

where $\mathcal{F}_\theta$ is a parameterized editing model. The alignment between the generated molecule and the instruction can be measured by a semantic matching function $\mathcal{A}(G^{\text{tgt}}, X)$, while the modification degree can be quantified by a molecular distance metric $\Delta(G^{\text{src}}, G^{\text{tgt}})$. The overall objective is thus to maximize instruction alignment while minimizing unnecessary structural deviation:

$$\mathcal{F}_\theta^* = \arg\max_{\mathcal{F}_\theta} \mathbb{E}_{(G^{\text{src}},X)\sim P_{\text{data}}}\left[\mathcal{A}(G^{\text{tgt}}, X) - \lambda\,\Delta(G^{\text{src}}, G^{\text{tgt}})\right], \tag{2}$$

where λ balances semantic fidelity and molecular similarity.

Molecule Graph Editing in O.O.D. Setting. Consider the training and testing graph datasets, $\mathcal{D}_{\text{train}} = \{(G_i^{\text{src}}, X_i)\}_{i=1}^{N^{tr}}$ and $\mathcal{D}_{\text{test}} = \{(G_i^{\text{src}}, X_i)\}_{i=1}^{N^{te}}$ sampled from distributions P_{train} and P_{test} respectively. In realistic molecule editing scenarios, these distributions often differ due to two major types of shifts:

- **Instruction shift:** linguistic variations, paraphrases, or unseen syntactic patterns in textual editing instructions that were not observed during training.
- **Property shift:** novel or unseen optimization objectives that differ from those in $\mathcal{D}_{\text{train}}$.

Formally, the task operates under **out-of-distribution (O.O.D.)** conditions if $P_{\text{train}} \neq P_{\text{test}}$ and P_{test} is unknown during training. The goal is to learn an optimal editing model $\mathcal{F}_\theta^*$ on $\mathcal{D}_{\text{train}}$ that generalizes effectively to the test data $\mathcal{D}_{\text{test}}$. This setting requires $\mathcal{F}_\theta$ to identify *invariant patterns* that capture stable relationships between molecular structures and textual instructions, thereby enabling robust generalization under instruction and property shifts.

3 Method

In this section, we introduce **MoFE**, a molecular graph editing LLM designed to enable robust generalization across unseen editing objectives and instruction descriptions. We first present the overall framework of MoFE (Section 3.1), followed by detailed descriptions of its three key components: the *Disentangled Graph Projector with Invariance Loss* (Section 3.2), the *Factor-aware LoRA Mixture-of-Experts* (Section 3.3), and the *Factor Disentanglement Loss Weighting* (Section 3.4). We conclude with the optimization procedure of our method (Section 3.5).

3.1 Overall Framework

As illustrated in Figure 1, MoFE consists of three interacting components that jointly address the challenge of out-of-distribution molecule graph editing. Specifically, the *Disentangled Graph Projector with Invariance Loss* encodes each molecule graph into multiple latent factors while ensuring consistency across paraphrased instructions. The *Factor-aware LoRA Mixture-of-Experts* enables modular editing by associating each LoRA expert with one latent factor. The *Factor Disentanglement Loss Weighting* adaptively reweights the supervision strength across experts, encouraging expert specialization and improving optimization stability.

These components form a coherent pipeline: the projector provides structured factor-level representations, the MoE module conditions generation on them, and the loss weighting encourages specialization and stability.

During inference, the input molecule graph is first processed by the projector to produce a set of factor-specific graph tokens, which are then inserted into the middle of the editing instruction token sequence. This combined sequence is fed into the LLM, where a

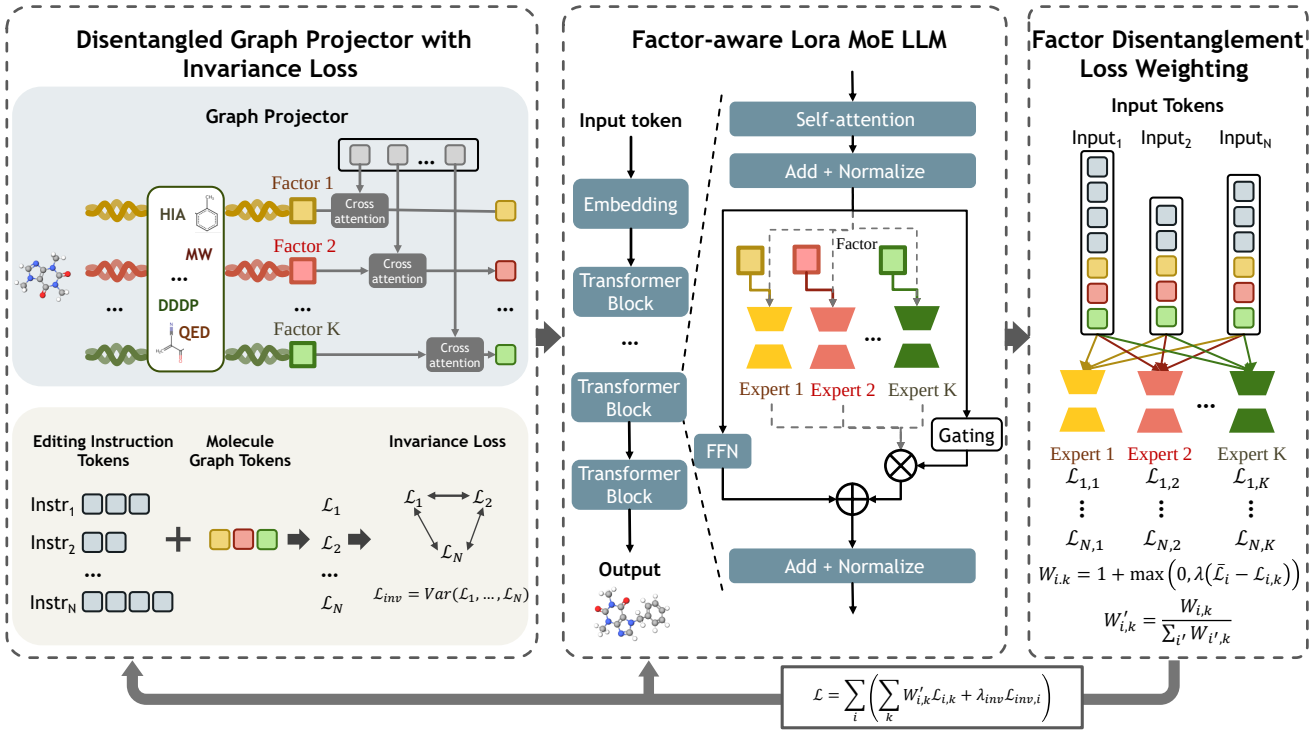


Figure 1: Overview of the proposed MoFE model. The model consists of three main components: (1) the *Disentangled Graph Projector with Invariance Loss* extracts editing-relevant latent factors from molecular graphs using cross-attention and enforces semantic invariance via a variance-based objective across paraphrased instructions; (2) the *Factor-aware LoRA Mixture-of-Experts LLM* dynamically routes input tokens to factor-specific LoRA experts using a learned gating mechanism, enabling modular and interpretable editing under diverse instruction-property combinations. (3) the *Factor Disentanglement Loss Weighting* strategy reinforces expert-factor alignment by assigning higher supervision weights to expert-factor pairs that perform well on relevant tasks, promoting joint disentanglement and stabilizing training.

gating mechanism selects the appropriate expert for each token. Finally, the model outputs the edited molecule that satisfies the specified editing objective.

3.2 Disentangled Graph Projector with Invariance Loss

To uncover the latent factors governing structure-property relationships, we design a *Disentangled Graph Projector* that maps each molecule graph into multiple semantically independent factors. The Disentangled Graph Projector contains K parallel GNN-based encoders, each dedicated to modeling one latent factor of molecular editing. Given an input molecule graph, the k-th GNN outputs node-level features $Z_k = \{z_{k,1}, ..., z_{k,n}\}$, which represent how each atom contributes to the k-th factor. To obtain a fixed number of global graph tokens, we aggregate them using cross-attention with B learnable prototypes $P_k = [\boldsymbol{p}_{k,1}, ..., \boldsymbol{p}_{k,B}]$. The aggregated representation of the k-th factor is computed as:

$$\hat{P}_k = \text{Attn}(P_k, Z_k, Z_k) \tag{3}$$

The aggregated factor representations from all K encoders are concatenated and further projected by an MLP to yield the final graph embedding:

$$H_{\text{graph}} = \text{MLP}([\hat{P}_1, ..., \hat{P}_K]) \tag{4}$$

The resulting H_{graph} serves as a sequence of graph tokens, which are incorporated into the downstream LLM as molecule-conditioned inputs.

While the disentangled projection enforces factor separation, we further observe that the same editing goal can often be expressed through syntactically different instruction prompts. To make the latent factors robust to such linguistic variations, we introduce an *Invariance Loss*, which encourages consistent model behavior across prompts describing the same objective.

Given N semantically equivalent instruction prompts labeled $\{X_1, X_2, ..., X_N\}$, each paired with the same molecule graph, the Disentangled Graph Projector produces corresponding token sequences $\{H_1, H_2, ..., H_N\}$. These are then processed by the LLM to compute their language modeling losses $\{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_N\}$. We define the Invariance Loss as the variance of these losses:

$$\mathcal{L}_{\text{inv}} = \text{Var}(\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_N) \tag{5}$$

This objective penalizes inconsistent model behavior across syntactically or stylistically different prompts with the same intended edit, forcing the model to focus on underlying semantics rather than

superficial linguistic variation. Consequently, it improves both robustness to paraphrases and generalizability under instruction-level distribution shifts.

Finally, each disentangled factor is paired with a distinct LoRA expert in the downstream Factor-aware LoRA MoE architecture. The gating mechanism of the MoE ensures that experts collaboratively contribute to the overall generation, implicitly enforcing each GNN to provide complementary, non-redundant factor representations. Together with the invariance constraint, this design encourages structural and semantic disentanglement among factors.

3.3 Factor-aware LoRA MoE

The Disentangled Graph Projector extracts a set of latent factors crucial to molecular editing, which are incorporated as contextual tokens into the LLM input. These factor tokens provide rich contextual information that should be effectively utilized during molecular editing. To ensure robust generalization to unseen editing goals and instruction phrasings, the model must treat all factors in a balanced and interpretable way. Without explicit control, dominant factors can overshadow weaker yet critical ones, resulting in biased adaptation and incomplete utilization of latent semantics. To address this, we design a *Factor-aware LoRA MoE* mechanism, which consists of multiple LoRA experts. Each LoRA expert is paired with one specific factor and processes it to support editing. This design enhances factor disentanglement and enhances generalization by allowing each expert to specialize in the adaptation behavior corresponding to its assigned factor.

We begin with the standard LoRA formulation, where low-rank adapters are introduced into the LLM's feedforward or attention layers. Given input hidden state $\boldsymbol{h} \in \mathbb{R}^d$, LoRA modifies the layer output as:

$$\text{LoRA}(\boldsymbol{h}) = W\boldsymbol{h} + \alpha BA\boldsymbol{h} \tag{6}$$

where $W \in \mathbb{R}^{d\times d}$ is the original model weight, $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{d\times r}$ are trainable low-rank matrices, and α is a scaling coefficient.

To make the LoRA layer aware of factor information, we add corresponding factor feature $\hat{P}_k$ extracted by the graph projector to $\boldsymbol{h}$ with a learnable projection:

$$\text{FactorLoRA}(\boldsymbol{h}, \hat{P}_k) = W\boldsymbol{h} + \alpha BA(\boldsymbol{h} + W'\text{Flatten}(\hat{P}_k)) \tag{7}$$

where W' is a learnable projection matrix for the k-th factor.

To support diverse modeling of multiple latent factors, we extend this into a Mixture-of-Experts (MoE) setting. We maintain K separate LoRA experts $\{(A_k, B_k, W'_k)\}$, each associated with one factor embedding. Given current hidden state $\boldsymbol{h}$, a gating layer dynamically computes selection weights over these experts with a learnable parameter matrix G:

$$\text{Gate}(\boldsymbol{h}) = \text{softmax}(G\boldsymbol{h}), \quad G \in \mathbb{R}^{K\times d} \tag{8}$$

Each LoRA expert produces its output using FactorLoRA, which are then summed together with gating weights as the final output:

$$\text{MoELoRA}(\boldsymbol{h}, \hat{P}) = \sum_{k=1}^{K} \text{Gate}_k(\boldsymbol{h})\text{FactorLoRA}_k(\boldsymbol{h}, \hat{P}_k) \tag{9}$$

where Gate_k is the k-th output of the gating layer, and FactorLoRA_k is the k-th factor-specific expert. This enables each token in the sequence to benefit from the most relevant factor-specific adaptation.

In practice, we apply this MoE LoRA specifically to the Feed-Forward Network (FFN) modules of the LLM, while keeping the Attention modules equipped with standard LoRA layers. This balance maintains model efficiency while ensuring factor-aware specialization where it matters most.

3.4 Factor Disentanglement Loss Weighting

To ensure that each expert effectively specializes in tasks associated with its corresponding factor, we introduce a *Factor Disentanglement Loss Weighting* strategy that promotes expert specialization. Our proposed loss weighting strategy is designed to achieve two goals simultaneously: first, to increase the gradient impact of experts on tasks they are good at; and second, to ensure that all experts are trained uniformly, so that their capabilities can be accurately assessed and model collapse is avoided.

To assess the capability of each expert FactorLoRA_k, we run a forward pass of the model where the MoE routing is fixed to select expert k in all layers. Given a batch of N inputs indexed by i, we compute their language modeling losses $\mathcal{L}_{i,k}$ when only expert k is utilized. We define the weighting of expert k for data point i as:

$$W_{i,k} = 1 + \max(0, \lambda(\bar{\mathcal{L}}_i - \mathcal{L}_{i,k})) \tag{10}$$

where $\bar{\mathcal{L}}_i$ is the average loss across all experts for input i. This encourages higher weights for experts that outperform the average. Next, we normalize the weights for each expert k across the batch:

$$W'_{i,k} = \frac{W_{i,k}}{\sum_{i'} W_{i',k}} \tag{11}$$

This normalization guarantees that all experts are trained with equal total amount of loss weighting. The final loss that will be used to optimize expert k becomes:

$$\mathcal{L}_k = \sum_i W'_{i,k}\mathcal{L}_{i,k} \tag{12}$$

This formulation ensures that each expert focuses on the samples where it performs best, while also maintaining equal opportunity for all experts to be evaluated.

3.5 Optimization Procedure

Training the MoFE framework involves balancing two objectives: ensuring that each factor-specific expert learns disentangled representations, and enabling effective collaboration among these experts through gating during inference. Directly optimizing the entire Factor-aware LoRA MoE end-to-end from scratch can lead to unstable convergence, as the gating network may prematurely dominate the optimization before individual experts have sufficiently specialized.

To address this issue, we adopt a two-stage training procedure that separates expert specialization and collaborative fine-tuning. We first pretrain each expert independently, and then fine-tune the complete model with gating to enable coordinated expert collaboration. The overall process is summarized in Algorithm 1 and 2.

Stage 1: Expert Pretraining with Disentanglement. In the first stage, we independently pretrain each factor-specific expert using its corresponding latent factor, without gating or mixture. For each latent factor $\hat{P}_k$, we pair it with its dedicated LoRA expert FactorLoRA_k and fix the expert routing to always select expert k.

Algorithm 1 Training Stage 1: Expert Pretraining with Disentanglement

Initialize graph projector with K graph encoders, K LoRA experts, and gating module
for each training batch **do**
  for each factor $k = 1$ to K **do**
    Compute the k-th factor representation $\hat{P}_k$
    Route tokens only through LoRA expert k
    Compute language modeling loss $\mathcal{L}_{i,k}$
    Sample paraphrased instructions $\{X^{(1)}, \ldots, X^{(N)}\}$ for same target
    Compute invariance loss $\mathcal{L}_{\text{inv},k} = \text{Var}\left(\mathcal{L}_{\text{LM}}(X^{(1)}), \ldots, \mathcal{L}_{\text{LM}}(X^{(N)})\right)$
  end for
  Compute mean loss $\bar{\mathcal{L}}_i = \frac{1}{K}\sum_k \mathcal{L}_{i,k}$
  for each factor $k = 1$ to K **do**
    Compute sample-level weights $W_{i,k} = 1 + \max(0, \lambda(\bar{\mathcal{L}}_i - \mathcal{L}_{i,k}))$
    Normalize across batch: $W'_{i,k} = \frac{W_{i,k}}{\sum_{i'} W_{i',k}}$
    Expert loss: $\mathcal{L}_k = \sum_i W'_{i,k}\mathcal{L}_{i,k}$
  end for
  Total loss for Stage 1: $\mathcal{L}_{\text{Stage 1}} = \sum_k \mathcal{L}_k + \lambda_{\text{inv}}\sum_k \mathcal{L}_{\text{inv},k}$
  Update graph projector and LoRA experts using $\mathcal{L}_{\text{Stage 1}}$
end for

Each training instance is processed using this fixed factor-expert pair, and the overall loss for expert k is computed by combining the language modeling loss $\mathcal{L}_k$ and the invariance loss $\mathcal{L}_{\text{inv}}$, weighted by the Factor Disentanglement Loss Weighting strategy described in Section 3.4. The total pretraining loss is:

$$\mathcal{L}_{\text{pretrain}} = \sum_{k=1}^{K} \mathcal{L}_k + \lambda_{\text{inv}}\mathcal{L}_{\text{inv}} \tag{13}$$

Here, $\mathcal{L}_k$ is the weighted expert loss for factor k, and λ_{inv} is a hyperparameter controlling the contribution of the invariance loss.

Stage 2: MoE Fine-tuning with Gating. In the second stage, we enable the full Factor-aware LoRA MoE architecture by activating the gating network. All factor experts are simultaneously available, and token representations are passed through the mixture-of-experts layers with gating as defined in Section 3.3. During this phase, we optimize the entire MoE module end-to-end using only the standard language modeling loss:

$$\mathcal{L}_{\text{MoE}} = \mathcal{L}_{\text{LM}} \tag{14}$$

The gating parameters and LoRA experts are jointly optimized in this stage, while the disentangled graph projector is frozen. This two-phase approach allows experts to first specialize independently and then collaboratively contribute to instruction-conditioned molecular editing under factor-aware modularization.

4 Experiments

We evaluate the proposed MoFE framework through comprehensive experiments. Section 4.1 describes the experimental setup, while Section 4.2 and Section 4.3 report results on I.D. and O.O.D. tasks, respectively. Ablation studies are presented in Section 4.4 to assess the contribution of each component.

Algorithm 2 Training Stage 2: MoE Fine-tuning with Gating

Freeze graph projector; enable all experts and gating network
for each training batch **do**
  Compute K factor embeddings $\{\hat{P}_k\}_{k=1}^{K}$ from projector
  For each LLM layer with MoE, apply gating: $\text{Gate}_i = \text{softmax}(G\boldsymbol{h}_i)$
  Combine expert outputs weighted by Gate_i
  Compute standard language modeling loss $\mathcal{L}_{\text{LM}}$
  Backpropagate and update LoRA experts and gating parameters
end for

4.1 Experimental Setup

Dataset. To evaluate our model's generalization ability, we adopt the MuMOInstruct dataset [7], a large-scale benchmark built on property optimization tasks with paired natural language instructions. MuMOInstruct defines multi-property molecular editing tasks based on six fundamental molecular properties:

- **BBBP (Blood-brain barrier permeability)**: Measures whether a molecule can cross the blood-brain barrier, which is essential for designing drugs targeting the central nervous system.
- **DRD2 (Dopamine receptor binding affinity)**: Assesses the molecule's affinity for binding to the dopamine receptor D2, important in neurological drug development.
- **HIA (Human intestinal absorption)**: Indicates the probability that a molecule can be absorbed through the human intestine, critical for oral bioavailability.
- **Mutag (Mutagenicity)**: Represents the potential of a compound to cause genetic mutations, with lower values being safer and more desirable in drug candidates.
- **plogP (Penalized octanol-water partition coefficient)**: Combines the logP value (lipophilicity) with synthetic accessibility and structural penalties, balancing hydrophobicity with drug-likeness and synthesis complexity.
- **QED (Quantitative estimate of drug-likeness)**: A composite score evaluating how "drug-like" a compound is based on multiple pharmacological properties.

Each task involves modifying a molecule so that it satisfies a specific combination of property constraints, such as "increase QED while decreasing mutagenicity". Task names correspond to the target property combinations (e.g., "**BDP**" for BBBP+DRD2+plogP). Following [7], each model is trained on multiple training tasks, and evaluated in three settings:

- **In-distribution (I.D.) tasks**: Tasks that use both property combinations and instruction formulations observed during training.
- **Seen Tasks with Unseen Instructions (O.O.D.)**: Tasks with property combinations seen during training but paired with paraphrased instructions not used during training, evaluating instruction-level generalization.
- **Unseen Tasks (O.O.D.)**: Tasks composed of novel property combinations (not seen in training) but using instruction styles

Table 1: Overall Performance in in-distribution (I.D.) Tasks. SR denotes the success rate; Sim denotes the molecular structure similarity between generated and original molecules; RI denotes relative property improvement (computed on successful edits). All metrics are averaged over each in-distribution task. ↑ indicates higher is better; Bold highlights the best score within each model category (General Models / Molecule Editing Models), and underline highlights the second-best score.

Model	BDP			BDQ			BPQ			DPQ			BDPQ		
	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑
General Models															
Mistral (0-shot)	6.60	**0.81**	0.68	3.00	**0.76**	0.53	15.80	**0.73**	0.51	2.20	**0.65**	0.41	3.20	0.77	0.87
Llama (0-shot)	22.00	0.73	0.74	2.20	0.64	0.53	28.40	0.64	0.72	2.60	0.62	0.32	5.20	**0.80**	0.62
Claude-3.5 (0-shot)	19.60	0.66	1.05	13.00	0.62	1.14	56.00	0.62	0.86	11.00	0.54	0.51	8.00	0.60	1.34
Mistral (5-shot)	35.20	0.64	2.10	17.00	0.60	2.32	68.60	0.63	0.79	10.40	0.54	1.10	11.00	0.69	0.96
Llama (5-shot)	35.40	0.57	**2.71**	16.60	0.43	**5.70**	34.60	0.70	0.64	8.20	0.44	**3.02**	9.60	0.54	3.45
Claude-3.5 (5-shot)	35.40	0.50	2.43	29.40	0.43	3.80	76.80	0.53	**1.24**	**29.20**	0.37	2.87	**20.80**	0.35	**3.53**
$\text{LlaSMol}_{\text{Mistral}}$	**43.60**	0.62	1.09	**31.40**	0.66	0.93	**86.00**	0.58	0.84	24.00	0.57	0.61	14.00	0.62	1.03
Molecule Editing Models															
ChatDrug	24.20	**0.60**	**5.74**	24.60	**0.60**	3.59	13.00	**0.61**	1.09	19.60	**0.56**	**5.39**	25.20	**0.64**	2.93
GeLLM^3O	77.00	0.53	3.73	79.60	0.56	**5.05**	95.00	0.47	**1.66**	57.00	0.49	2.50	52.20	0.49	**3.48**
Ours	**87.00**	0.58	3.34	**85.20**	**0.60**	4.33	**97.00**	0.55	1.38	**64.40**	0.55	1.87	**59.00**	0.51	3.28

observed during training, assessing generalization to new optimization objectives.

Baselines. We compare our method with the following baselines:

- **General-purpose models** [11]: General-purpose LLMs (not specifically designed for molecule editing) including Mistral-7B Instruct-v0.3[11], Llama-3.1 8B Instruct[9], Claude-3.5[2], and LlaSMol tuned on Mistral-7B[25]. We follow the settings of [7] for these baselines.
- **ChatDrug** [15]: A conversational LLM framework for drug editing that integrates prompt engineering, retrieval-based domain feedback, and multi-round interaction. We evaluate the Llama2-based variant of ChatDrug in our experiments.
- **GeLLM^3O** [7]: Instruction-tuned LLMs specifically designed for multi-property molecule optimization using the MuMOInstruct dataset. We adopt the generalist version $\text{GeLLM}^3\text{O-P(6)}_{\text{Llama}}$, which is fine-tuned on all combinations of six core molecular properties and built upon Llama3.1-8B with LoRA adapters. This choice ensures a fair comparison with our method, as both are based on Llama and trained on the same multi-property editing benchmark.

Evaluation Metrics. We follow prior work [7] and evaluate molecule editing quality using three metrics:

- **Success Rate (SR)**: The percentage of edited molecules that satisfy all desired property constraints. A higher SR indicates better controllability and task completion accuracy.
- **Similarity (Sim)**: The average Tanimoto similarity [3] between the edited molecule and the input molecule, computed over Morgan fingerprints. A higher similarity suggests that the model preserves the core molecular structure while making minimal necessary changes.
- **Relative Improvement (RI)**: The average improvement of each optimized property relative to its initial score in the input molecule. Higher RI values indicate more significant property enhancement on successfully edited molecules.

Among these, **SR** serves as the primary indicator of editing performance. High similarity with low SR often reflects trivial or overly conservative edits, whereas high RI with low SR suggests unstable editing behavior, since RI is computed only over successful edits. Therefore, SR best captures whether the model performs controlled and effective edits that truly achieve the intended molecular transformations.

4.2 In-distribution (I.D.) Task Results

To evaluate the in-distribution performance of our model, we compare it against both general-purpose LLMs and molecule editing-specific baselines across five multi-property editing tasks. As shown in Table 1, our method consistently achieves the highest success rates (SR) among molecule editing models, while maintaining competitive molecular similarity (Sim) and solid relative property improvement (RI), indicating that our model has good performance on in-distribution molecule editing tasks.

We also observe that, although ChatDrug and some general-purpose models yield slightly higher Sim in some tasks, their success rates are substantially lower. This indicates that high similarity alone does not imply successful or meaningful molecular edits, as such results may arise from conservative modifications that leave the molecule largely unchanged. In contrast, our model achieves a balanced performance across SR, Sim, and RI, suggesting that it not only meets the editing objectives more frequently but also generates chemically reasonable and property-improving molecules. Overall, these results confirm the strong in-distribution editing capability of our method.

4.3 Out-of-distribution (O.O.D.) Task Results

We further evaluate the generalization ability of our model through two O.O.D. settings: (1) unseen tasks with seen instructions, and

Table 2: Overall performance in unseen tasks (O.O.D.) with seen instructions. SR denotes the success rate; Sim denotes the molecular structure similarity between generated and original molecules; RI denotes relative property improvement (computed on successful edits). All metrics are averaged over each O.O.D. task. ↑ indicates higher is better; Bold highlights the best score within each model category (General Models / Molecule Editing Models), and underline highlights the second-best score.

Model	MPQ			BDMQ			BHMQ			BMPQ			HMPQ		
	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑
General Models															
Mistral (0-shot)	11.20	**0.57**	0.48	1.20	0.68	0.37	12.71	0.73	1.90	12.57	0.61	0.54	21.88	**0.72**	0.72
Llama (0-shot)	25.80	0.44	0.61	1.20	**0.76**	0.30	11.02	**0.74**	0.68	16.75	0.51	0.57	15.62	0.47	0.60
Claude-3.5 (0-shot)	17.40	0.49	0.52	15.00	0.57	0.87	38.98	0.51	2.35	44.50	0.55	0.85	38.54	0.54	1.01
Mistral (5-shot)	59.60	0.54	0.57	20.40	0.59	1.65	34.75	0.70	1.31	49.21	0.62	0.73	46.88	0.66	0.91
Llama (5-shot)	34.80	**0.57**	0.53	16.80	0.39	**3.22**	36.44	0.67	1.13	31.94	**0.66**	0.60	33.33	0.68	0.61
Claude-3.5 (5-shot)	50.60	0.49	**0.71**	**30.40**	0.49	2.32	52.54	0.48	**2.52**	52.36	0.46	**1.08**	**65.62**	0.48	**1.32**
$\text{LlaSMol}_{\text{Mistral}}$	**76.40**	0.55	0.53	28.20	0.66	0.52	**53.39**	0.62	1.14	**64.92**	0.58	0.57	53.12	0.62	0.70
Molecule Editing Models															
ChatDrug	19.60	**0.60**	**1.21**	24.00	**0.66**	3.22	27.24	**0.60**	0.54	25.31	**0.64**	**4.57**	29.58	**0.61**	1.24
GeLLM^3O	93.60	0.48	0.91	74.20	0.55	3.25	93.22	0.49	**3.57**	**95.29**	0.49	1.20	**97.92**	0.46	**1.76**
Ours	**94.80**	0.55	0.81	**80.60**	0.55	**3.32**	**94.92**	0.55	2.91	**95.29**	0.55	1.37	96.88	0.53	1.34

Table 3: Overall performance in seen tasks with unseen instructions (O.O.D.). SR denotes the success rate; Sim denotes the molecular structure similarity between generated and original molecules; RI denotes relative property improvement (computed on successful edits). All metrics are averaged over each O.O.D. task. ↑ indicates higher is better; Bold highlights the best score in unseen instructions tasks, and underline highlights the second-best score.

Model	Instr	BDP			BDQ			BPQ			DPQ			BDPQ		
		SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑	SR↑	Sim↑	RI↑
ChatDrug	seen	24.20	0.60	5.74	24.60	0.60	5.32	13.00	0.61	1.09	19.60	0.56	5.39	25.20	0.64	2.93
	unseen	20.00	**0.63**	**5.26**	23.20	0.56	3.59	14.60	**0.57**	1.14	22.00	**0.61**	**2.96**	21.40	**0.63**	3.45
GeLLM^3O	seen	77.00	0.53	3.73	79.60	0.56	5.05	95.00	0.47	1.66	57.00	0.49	2.50	52.20	0.49	3.48
	unseen	64.60	0.53	3.06	73.40	0.57	**4.56**	95.60	0.47	**1.66**	53.60	0.48	2.15	46.40	0.48	**3.52**
Ours	seen	87.00	0.58	3.34	85.20	0.60	4.33	97.00	0.55	1.38	64.40	0.55	1.87	59.00	0.51	3.28
	unseen	**83.20**	0.55	3.27	**84.40**	**0.58**	4.33	**96.00**	0.53	1.51	**58.60**	0.53	2.37	**57.00**	0.50	3.37

(2) seen tasks with unseen instructions. These experiments test the model's robustness to new property objectives and linguistic variations.

4.3.1 Unseen Tasks with Seen Instructions. To assess the generalization ability of our model, we evaluate it on editing tasks involving novel combinations of molecular properties not seen during training. As shown in Table 2, our model achieves consistent and significant improvements over both general-purpose LLMs and molecule editing baselines.

Specifically, our method achieves the best SR on most O.O.D. tasks, demonstrating its robust generalization to unseen optimization objectives. Meanwhile, it maintains competitive Sim and strong RI scores, achieving a favorable balance between structural fidelity and property optimization. In contrast, ChatDrug has high similarity but suffers from low SR. Overall, the superior performance of MoFE across all O.O.D. tasks shows that our disentangled, factor-aware editing framework can generalize to unseen molecular objectives, while still performing faithful and minimal structural changes.

4.3.2 Seen Tasks with Unseen Instructions. To evaluate the robustness of our model against natural language variation, we test it on seen property objectives with paraphrased instructions.

As shown in Table 3, our method has competitive SR and similarity in tasks with unseen instructions, demonstrating its ability to generalize to varied task instructions. Furthermore, our method shows strong resilience to unseen instructions, achieving a lower overall performance drop in the success rate from seen instructions to unseen instructions compared to GeLLM^3O. ChatDrug, as a training-free method, shows relatively stable performance between seen and unseen instruction settings, but its overall success rates remain substantially lower across all tasks, suggesting that the lack of targeted training limits its ability to perform complex, property-specific edits.

These results suggest that MoFE effectively captures instruction-invariant semantics, benefiting from its use of invariance loss and disentangled graph representations. This makes it more robust to paraphrased inputs and improves usability in real-world settings.

Table 4: Ablation study on O.O.D. tasks. SR denotes the success rate; Sim denotes the molecular structure similarity between generated and original molecules; RI denotes relative property improvement (computed on successful edits). All averaged over all O.O.D. tasks. "Full MoFE" refers to the complete model, while other rows indicate variants with one core module removed.

Model Variant	SR (O.O.D. Avg)↑	Sim (O.O.D. Avg)↑	RI (O.O.D. Avg)↑
Full MoFE (Ours)	84.17	0.54	2.46
w/o Graph Projector (no factor)	78.90	0.51	2.38
Shared LoRA (no MoE)	80.26	0.52	2.52
w/o Loss Weighting	82.12	0.52	2.43

4.4 Ablation Study

To evaluate the contribution of each component in our proposed MoFE model, we conduct an ablation study under O.O.D. settings. We report the average metrics over all O.O.D. editing tasks.

As shown in Table 4, each module contributes significantly to the overall performance. Removing the disentangled graph projector, which extracts disentangled editing-relevant factors, results in the largest drop in SR, confirming the necessity of modeling editing-relevant structural factors. Replacing the factor-aware MoE with a shared LoRA leads to a decrease in both SR and Sim, indicating that expert specialization is essential for adapting to diverse instructions. Disabling the loss weighting strategy also reduces performance, showing its role in guiding expert–factor alignment and reducing training interference. Overall, these results demonstrate that all three modules jointly contribute to the robust generalization of MoFE under distribution shifts.

5 Related Work

Molecule Editing. Molecule editing aims to modify molecule structures to satisfy desired properties while preserving key scaffolds [10, 17]. It plays a central role in drug discovery, where edits may enhance solubility, reduce toxicity, or improve binding affinity and so on. Traditional approaches typically rely on rule-based fragment modifications or property heuristics [10, 17]. While interpretable, these methods often lack scalability and struggle with multi-objective optimization. Recent work introduces multi-modal molecule structure–text models. MoleculeSTM [14] aligns textual descriptions with 2D molecular structures via contrastive learning, enabling zero-shot molecule editing and retrieval. Building on this, MoleculeSTM-3D [13] incorporates 3D molecular conformations and aligns them with natural language using a geometry-text contrastive framework. More recently, ChatDrug [15] proposes a conversational framework for drug editing using LLMs. It integrates prompt design, retrieval, and domain feedback with interactive editing to support small molecules, peptides, and proteins. Through utilizing MuMOInstruct, a large-scale instruction-tuning dataset tailored to complex multi-property molecule optimization, GeLLM^{3}O [7] introduces a series of instruction-tuned LLMs for molecule optimization. Our method focuses on the challenging setting of O.O.D. generalization in multi-property molecule optimization.

Large language Models on Molecule Graphs. Large Language Models (LLMs) have recently been extended to the molecular graph [5, 12, 16, 18, 19, 27, 28], where molecules are represented as graphs of atoms and bonds. By integrating the structured topology of molecular graphs with rich textual knowledge, LLMs can improve the understanding, generation, and prediction of molecular properties. GIMLET[28] is a unified graph-text model designed for instruction-based zero-shot learning, which encodes both molecular graphs and task instructions without separate GNN modules and decouples encoding of the graph from tasks instructions in the attention mechanism. LLaMo[18] introduces a multi-level graph projector that transforms molecular graphs into language-aligned tokens, enabling end-to-end molecule-language understanding. It uses machine-generated molecular dialogues to instruction-tune LLMs and achieves state-of-the-art performance in tasks like property prediction and molecule captioning.

LoRA Mixture-of-Experts. LoRA Mixture-of-Experts [4, 8, 20, 22, 26] combines the efficiency of Low-Rank Adaptation with the flexibility of Mixture-of-Experts. It enables modular, parameter-efficient fine-tuning while adapting to diverse tasks. MOLE [22] proposes a hierarchical gating mechanism to dynamically combine LoRA experts per layer. LoRAMoE [8] addresses catastrophic forgetting by freezing the base model and routing world-knowledge-preserving LoRAs for instruction tuning. MixLoRA [20] extends LoRA to multimodal instruction tuning by dynamically selecting LoRA modules per instance. These works demonstrate LoRA MoE's effectiveness across NLP and multimodal tasks. Our work further explores its application to molecule editing, aiming to enhance structure-aware generation via modular adaptation.

6 Conclusion

In this work, we propose MoFE, a large language model framework for instruction-guided molecule graph editing under O.O.D. scenarios. To tackle the challenges of generalizing to unseen instructions, we introduce a modular architecture with three key components: a Disentangled Graph Projector with Invariance Loss, a Factor-aware LoRA Mixture-of-Experts module, and a Factor Disentanglement Loss Weighting strategy. Together, these components allow the model to extract stable editing-relevant factors, support flexible and specialized adaptation, and promote robust generalization. Experimental results on the MuMOInstruct benchmark demonstrate that MoFE achieves strong performance on both I.D. and O.O.D. molecule editing tasks, outperforming existing baselines.

Acknowledgments

This work was supported by the National Key Research and Development Program of China No.2023YFF1205001, Beijing National Research Center for Information Science and Technology under Grant No. BNR2023TD03006.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Anthropic. 2024. Claude 3.5 Haiku. https://www.anthropic.com/claude/haiku.

[3] Dávid Bajusz, Anita Rácz, and Károly Héberger. 2015. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics* 7 (2015), 1–13.

[4] Cheng Chen, Junchen Zhu, Xu Luo, Hengtao Shen, Jingkuan Song, and Lianli Gao. 2024. CoIN: A Benchmark of Continual Instruction Tuning for Multimodel Large Language Models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.*

[5] Yongqiang Chen, Quanming Yao, Juzheng Zhang, James Cheng, and Yatao Bian. 2024. HIGHT: Hierarchical Graph Tokenization for Graph-Language Alignment. *CoRR* abs/2406.14021 (2024). arXiv:2406.14021 doi:10.48550/ARXIV.2406.14021

[6] DeepSeek-AI. 2024. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[7] Vishal Dey, Xiao Hu, and Xia Ning. 2025. GeLLM^{3}O: Generalizing Large Language Models for Multi-property Molecule Optimization. *CoRR* abs/2502.13398 (2025). arXiv:2502.13398 doi:10.48550/ARXIV.2502.13398

[8] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating World Knowledge Forgetting in Large Language Models via MoE-Style Plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024.* Association for Computational Linguistics, 1932–1945. doi:10.18653/V1/2024.ACL-LONG.106

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024). arXiv:2407.21783 doi:10.48550/ARXIV.2407.21783

[10] Chunngai Hui, Zhuo Wang, Shiping Wang, and Chunfa Xu. 2022. Molecular editing in natural product synthesis. *Organic Chemistry Frontiers* 9, 5 (2022), 1451–1457.

[11] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *CoRR* abs/2310.06825 (2023). arXiv:2310.06825 doi:10.48550/ARXIV.2310.06825

[12] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large Language Models on Graphs: A Comprehensive Survey. *IEEE Trans. Knowl. Data Eng.* 36, 12 (2024), 8622–8642. doi:10.1109/TKDE.2024.3469578

[13] Haorui Li, Shengchao Liu, Hongyu Guo, and Anima Anandkumar. [n. d.]. Geometry-text Multi-modal Foundation Model for Reactivity-oriented Molecule Editing. In *NeurIPS 2024 Workshop on AI for New Drug Modalities.*

[14] Shengchao Liu, Weili Nie, Chengpeng Wang, Jiarui Lu, Zhuoran Qiao, Ling Liu, Jian Tang, Chaowei Xiao, and Animashree Anandkumar. 2023. Multi-modal molecule structure-text model for text-based retrieval and editing. *Nat. Mac. Intell.* 5, 12 (2023), 1447–1457. doi:10.1038/S42256-023-00759-6

[15] Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. 2024. Conversational Drug Editing Using Retrieval and Domain Feedback. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=yRrPfKyJQ2

[16] Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. 2024. MolecularGPT: Open Large Language Model (LLM) for Few-Shot Molecular Property Prediction. *CoRR* abs/2406.12950 (2024). arXiv:2406.12950 doi:10.48550/ARXIV.2406.12950

[17] Chunhua Ma, Craig W Lindsley, Junbiao Chang, and Bin Yu. 2024. Rational molecular editing: a new paradigm in drug discovery. 11459–11466 pages.

[18] Jinyoung Park, Minseong Bae, Dohwan Ko, and Hyunwoo J. Kim. 2024. LLaMo: Large Language Model-based Molecular Graph Assistant. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.*

[19] Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh V. Chawla, and Chao Huang. 2024. A Survey of Large Language Models for Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024.* ACM, 6616–6626. doi:10.1145/3637528.3671460

[20] Ying Shen, Zhiyang Xu, Qifan Wang, Yu Cheng, Wenpeng Yin, and Lifu Huang. 2024. Multimodal Instruction Tuning with Conditional Mixture of LoRA. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024.* Association for Computational Linguistics, 637–648. doi:10.18653/V1/2024.ACL-LONG.38

[21] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. 2019. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery* 18, 6 (2019), 463–477.

[22] Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of LoRA Experts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=uWvKBCYh4S

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

[24] An Yang, Baosong Yang, Beichen Zhang, and et al. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).

[25] Botao Yu, Frazier N Baker, Ziqi Chen, Xia Ning, and Huan Sun. [n. d.]. LlaSMol: Advancing Large Language Models for Chemistry with a Large-Scale, Comprehensive, High-Quality Instruction Tuning Dataset. In *ACL 2024 Workshop Language+ Molecules.*

[26] Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. 2024. Boosting Continual Learning of Vision-Language Models via Mixture-of-Experts Adapters. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024.* IEEE, 23219–23230. doi:10.1109/CVPR52733.2024.02191

[27] Zhaoning Yu, Xiangyang Xu, and Hongyang Gao. 2024. G2T-LLM: Graph-to-Tree Text Encoding for Molecule Generation with Fine-Tuned Large Language Models. *CoRR* abs/2410.02198 (2024). arXiv:2410.02198 doi:10.48550/ARXIV.2410.02198

[28] Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhihong Deng, Lingpeng Kong, and Qi Liu. 2023. GIMLET: A Unified Graph-Text Model for Instruction-Based Molecule Zero-Shot Learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.*

A Training and Implementation Details

A.1 Training Details

We adopt a two-stage training strategy to effectively optimize the proposed MoFE architecture. In **Stage 1**, we pretrain each factor-specific LoRA expert independently using its associated graph latent representation, without gating. This encourages each expert to specialize in a disentangled editing-relevant factor. In **Stage 2**, we enable the full mixture-of-experts (MoE) architecture, including the gating mechanism, and fine-tune the model in an end-to-end manner.

During both stages, the base LLM remains frozen to ensure parameter efficiency and stability. Stage 1 emphasizes expert disentanglement through factor-specific routing and loss weighting, while Stage 2 learns to dynamically compose experts based on token-level gating.

A.2 Implementation Details

We provide key implementation details for reproducibility here. Our backbone model is Llama-3.1-8B-Instruct. The LoRA layers are created with $r = 4$ and alpha $= 32$. Standard LoRA modules are applied to Q and V projection weights in attention layers, while our proposed MoE LoRA modules are applied to feedforward layers. The graph projector comprises K independent GIN graph encoders. The number of factors is set to $K = 4$ in the experiments.

We optimize the model using AdamW with a batch size of 64. The learning rates are 2×10^{-5} for the LLM and 1×10^{-4} for the GNN projector. Stage 1 and Stage 2 are trained for 1 and 2 epochs, respectively. All experiments are conducted using 4 NVIDIA A100 GPUs (40GB).

Table 5: Task information

Type	Task	#Train	#Val	#Test	#Mols	Properties
In-distribution (I.D.)	BDP	2,064	230	500	2,449	BBBP, DRD2, plogP
	BDQ	4,472	497	500	4,614	BBBP, DRD2, QED
	BPQ	4,048	450	500	6,953	BBBP, plogP, QED
	DPQ	2,114	235	500	2,589	DRD2, plogP, QED
	BDPQ	624	70	500	802	BBBP, DRD2, plogP, QED
Out-of-distribution (O.O.D.)	MPQ	3,132	349	500	5,384	Mutag, plogP, QED
	BDMQ	601	67	500	791	BBBP, DRD2, Mutag, QED
	BHMQ	191	22	118	333	BBBP, HIA, Mutag, QED
	BMPQ	373	42	191	690	BBBP, Mutag, plogP, QED
	HMPQ	234	26	96	417	HIA, Mutag, plogP, QED

Table 6: Additional baselines on multi-property optimization tasks. Higher is better for SR and Sim.

Model	BDP (SR↑/Sim↑)	BDQ (SR↑/Sim↑)	BPQ (SR↑/Sim↑)	DPQ (SR↑/Sim↑)	BDPQ (SR↑/Sim↑)
GPT-o4-mini	32.20/0.58	22.80/0.63	58.00/0.67	14.20/0.55	6.60/0.42
LLaMo	8.03/0.62	2.94/0.42	20.49/0.47	0.76/0.80	0.00/nan

Table 7: Results on OpenMolIns-large. Higher is better for SR and Sim.

Model	AddComponent (SR/Sim)	DelComponent (SR/Sim)	SubComponent (SR/Sim)
Llama3.1-8B (OpenMolIns-large)	58.22/0.65	51.04/0.51	54.40/0.63
Ours (OpenMolIns-large)	70.60/0.68	83.88/0.62	59.32/0.74

Table 8: Results on MolOpt-Instructions. Higher is better for SR and Valid.

Model	QED (SR/Valid)	BBBP (SR/Valid)
DrugAssist (MolOpt-Instructions)	0.76/0.99	0.82/0.99
Ours	0.85/1.00	0.99/1.00

Table 9: Sensitivity analysis on the number of latent factors K.

K (factors)	SR (Avg)	Sim (Avg)
2	79.20	0.60
4	82.29	0.55
6	80.00	0.53

MoFE is trained in about 1.5 days on 4×A100 GPUs. The memory footprint is very close to the baseline, indicating that the added modules introduce no noticeable overhead.

B Theoretical Discussion on Invariance Loss

The proposed invariance loss enforces consistent behavior across paraphrased instructions sharing the same editing goal. This design is rooted in the principle that the model should behave similarly for inputs with equivalent intent.

Formally, let $\mathcal{X}_{\text{eq}} = \{X_1, X_2, ..., X_N\}$ be a set of instruction prompts expressing the same desired molecular transformation, and let $H(X_i)$ be the latent token representation after conditioning on the graph. Then, the expected language modeling loss is:

$$\mathbb{E}_{X \in \mathcal{X}_{\text{eq}}} [\mathcal{L}_{\text{LM}}(X)]$$

To ensure consistency across variants, we minimize the empirical variance:

$$\mathcal{L}_{\text{inv}} = \text{Var} (\mathcal{L}_{\text{LM}}(X_1), \ldots, \mathcal{L}_{\text{LM}}(X_N))$$

This encourages the model to learn representations invariant to linguistic variation, forcing attention to stable structural-property patterns encoded in the graph tokens. In practice, this helps mitigate overfitting to prompt phrasing and improves robustness in unseen instruction forms.

C Dataset Details

Dataset Split. We adopt the MuMOInstruct benchmark, which includes over 1.2M (molecule, instruction)–edit pairs across six key drug-relevant properties: BBBP, DRD2, HIA, Mutag, QED, and plogP. We use the official train/validation/test splits. Refer to Table 5 for more information about the tasks.

D Additional Experimental Results

D.1 Comparison with more baselines

We additionally include results of GPT-o4-mini and LLaMo on MuMOInstruct for comparison. Both perform poorly: GPT-o4-mini lacks molecular reasoning capability in this setting, while LLaMo is designed for molecule understanding rather than multi-property optimization.

D.2 Results on more datasets

We further evaluate on MolOpt-Instructions and OpenMolIns, two datasets on instruction-guided molecule editing.

On OpenMolIns, both models are fine-tuned on the large-scale split, and MoFE outperforms the baseline (Llama3.1-8B) across editing tasks on both SR and Sim scores (Table 7).

On MolOpt-Instructions, since the dataset has overlapping tasks with MuMOInstruct, we perform direct inference on these tasks using our trained model, yielding higher SR and Valid than the baseline (Table 8) and confirming its consistent advantage in instruction-guided molecule editing tasks.

D.3 Hyperparameter Sensitivity

Here we provide a sensitivity analysis on the number of latent factors K. Table 9 shows that $K = 4$ achieves the best balance between success rate and similarity.