MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **97** (2027) 315–353
ISSN: 0340–6253
doi: 10.46793/match.97-2.11226

MetaMolGen: A Neural Graph Motif Generation Model for De Novo Molecular Design

Zimo Yan[1], Jie Zhang[1], Zheng Xie[1,*], Chang Liu[1], Yizhen Liu[1], Yiping Song[1]

[1] *National University of Defense Technology, Changsha, China.*
yanzimo20@nudt.edu.cn, zhangjie@nudt.edu.cn, xiezheng81@nudt.edu.cn, nudt_liuchang_1997@nudt.edu.cn, liuyizhen23@nudt.edu.cn, songyiping@nudt.edu.cn

(Received April 1, 2026)

Abstract

Molecular generation is crucial for drug discovery and materials science, especially under data-scarce conditions where traditional generative models often fail to generalize conditionally. To address this, we propose MetaMolGen, a first-order meta-learning-based molecular generator tailored for few-shot and property-conditioned generation. MetaMolGen standardizes graph motifs by mapping them into a normalized latent space and employs a lightweight autoregressive model to generate SMILES sequences that accurately reflect molecular structures. A learnable property projector is integrated into the generation process to support target property conditioning. Experiments show that MetaMolGen consistently generates valid and diverse molecules in low-data regimes, outperforming conventional baselines and demonstrating strong adaptability and efficient conditional generation.

*Corresponding author: Zheng Xie (xiezheng81@nudt.edu.cn).

1 Introduction

Designing and synthesizing molecules with specific properties is a fundamental challenge in fields like drug discovery, materials science, and environmental chemistry [29, 32]. The vastness of chemical space makes molecular design inherently complex. While over 10^8 molecules have been synthesized, this represents only a tiny fraction of the estimated 10^{23} to 10^{60} potential drug-like molecules [24, 28], highlighting both the immense potential and the difficulty of molecular exploration.

When employing intelligent models for molecular design, ensuring structural validity remains a critical challenge. Current molecular generation models primarily adopt one of three strategies. Graph-based methods generate molecules atom-by-atom or bond-by-bond, enabling high interpretability and direct manipulation of molecular topology; a representative example is StrucGCN [36], which improves graph embedding by incorporating structural information. Sequence-based models utilize linear representations such as SMILES, and although they offer strong scalability, they often struggle with generating valid structures. Approaches like RNNs [31] and Transformer-based models [1] fall into this category. Tree-based methods, such as Junction Tree VAE [14], generate molecules by assembling subgraph fragments in a hierarchical manner, achieving 100% structural validity.

Beyond these mainstream paradigms, emerging strategies are addressing more specific challenges in molecular generation. Diffusion-based models, like PIDiff [5], leverage iterative denoising to construct 3D molecular conformations, while retrieval-augmented methods such as Reinvent [30] enhance synthetic feasibility and diversity by integrating database search with generative models. Recent advancements including MGCVAE [18] and GAN-based frameworks like ORGAN [11] further improve multi-property control and support task-specific molecular generation.

Despite these advancements, existing methods still suffer from persistent limitations that restrict their effectiveness in practical applications. A key challenge lies in the difficulty of simultaneously ensuring both validity and uniqueness, making it problematic to generate chemically diverse yet

valid molecules tailored for specific properties. For instance, MolGAN [6] achieves nearly perfect validity (e.g., 98%–100%) on MOSES benchmarks, but exhibits low uniqueness and diversity (often below 10%), revealing a clear trade-off. To provide a comprehensive evaluation of model performance across multiple objectives, we define an *Overall Score* that integrates validity, uniqueness, diversity, and druglikeness. The formulation of the Overall Score is given in Equation 27.

The data-intensive nature of deep learning approaches presents another significant barrier, typically demanding training sets on the order of 10^6 samples. This requirement becomes prohibitive in data-scarce scenarios such as orphan drug discovery, severely limiting their applicability to smaller datasets. For example, sequence-based methods like Grammar-VAE [17] generally require vast amounts of data to achieve optimal performance, whereas our meta-learning-based approach can effectively leverage prior knowledge to rapidly adapt with only a fraction of the data, yielding superior results in few-shot settings.

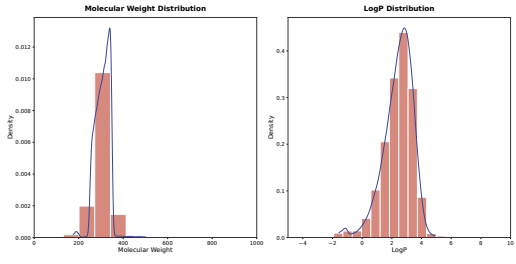


Figure 1. Distribution of Key Molecular Properties

More critically, while many molecular generation models require complete datasets to fully capture the nuances of molecular properties, our preliminary analysis reveals that some key properties, particularly LogP, tend to exhibit approximately unimodal distributions (Fig. 1). In contrast, molecular weight is right-skewed and hydrogen bond acceptor and donor counts display more complex patterns; yet even these can largely be seen as following a unimodal trend. This predominance of unimodal characteristics in continuous properties like LogP suggests a potential statistical bias in molecular datasets that may be underutilized. Notably, this bias implies that our approach could tolerate partial data or even benefit from using only a fraction of the available data, thereby alleviating the reliance

on large-scale, complete datasets. Further investigation is warranted to determine how leveraging this bias might improve generation performance.

Motivation: This leads to the following question: *Can we design a model that not only generates molecules under small-sample conditions but also uses the unimodal property distribution to enhance and balance the model's validity and uniqueness?*

To address this challenge, we designed the Meta-Optimized Molecular Generation model (MetaMolGen), which leverages feature transformation and small-sample learning techniques to overcome these limitations. The MetaMolGen framework focuses on generating valid molecules that maintain diversity while explicitly modeling unimodal distributions in the chemical space.

Developed Methods: We propose MetaMolGen, a molecular generation model based on Conditional Neural Processes (CNPs) framework. To improve generalization under skewed molecular property distributions, we introduce a normalization layer before the context encoder, transforming feature distributions into a quasi-normal form. Additionally, we integrate the Reptile meta-learning algorithm [21] into the CNP framework to enable efficient few-shot learning, allowing training on datasets of size 10^4–10^5 without requiring second-order gradient computations. MetaMolGen achieves high molecular uniqueness and property alignment while maintaining strong validity and diversity.

The primary contributions of this work are as follows:

1. To address the challenge of balancing model validity and uniqueness, we propose MetaMolGen, a novel meta-learning molecular generation model designed for few-shot molecular generation. Additionally, MetaMolGen is capable of conditional generation, enabling the generation of molecules with specific chemical properties based on given target attributes.

2. Through large-scale experiments on multiple datasets, we validated the advantages of MetaMolGen in molecular generation tasks on small to medium-sized datasets. The model significantly outperforms traditional generative models in terms of validity, uniqueness,

property matching accuracy, and generation quality, as reflected in its superior Overall Score across key metrics.

3. We introduce a learnable standardization module within the Conditional Neural Processes framework, which mitigates scale imbalances across molecular features, improving the model's numerical stability and convergence speed.

4. We integrate Reptile into the framework to achieve rapid adaptation to small-scale datasets, and provide a theoretical discussion on the convergence characteristics of Reptile under this setting.

2 Literature review

Recent advances in deep learning have significantly expanded the scope of molecular generative modeling, leveraging architectures such as variational autoencoders (VAEs), generative adversarial networks (GANs), and reinforcement learning (RL)-based methods. These techniques have been applied to a range of tasks, including drug discovery, molecular optimization, and materials design. Despite their success, challenges remain in navigating the high-dimensional molecular space, capturing complex dependencies, and achieving generalization in low-data scenarios.

2.1 Graph-based methods

Graph-based methods represent molecules as undirected graphs, where atoms correspond to nodes and chemical bonds to edges. This structure-aware representation naturally aligns with molecular topology, enabling models to reason over local and global structural information. Graph Neural Networks (GNNs) have been widely used in this context, providing a mechanism for learning rich molecular embeddings.

One early approach, MolGAN, combines GANs with reinforcement learning to generate molecular graphs directly, bypassing the need for SMILES decoding. While MolGAN achieves near-perfect validity on benchmark datasets, it struggles with uniqueness and diversity, achieving less

than 10% uniqueness in some cases. You et al. [35] proposed the Graph Convolutional Policy Network (GCPN), which combines GNNs with RL to generate molecules that satisfy property constraints, offering a goal-directed design capability.

Further advancements, such as StrucGCN, incorporate structural priors into graph embedding processes to improve model expressiveness. Meanwhile, the Crystal Graph Convolutional Neural Network (CGCNN) [34] has demonstrated that graph-based architectures are also effective in predicting material properties. Despite these advances, graph-based models often require large training datasets to learn meaningful patterns and may struggle to generalize in data-scarce settings.

2.2 Sequence-based methods

Sequence-based models represent molecules as one-dimensional strings, typically using the SMILES notation [33], allowing generative tasks to be treated similarly to natural language processing. These methods are generally efficient and easy to scale but often suffer from lower validity due to the strict syntactic constraints of SMILES strings.

Recurrent Neural Networks (RNNs) have been widely applied to SMILES generation, learning to produce valid molecules through sequence modeling. Transformer-based models have improved scalability and attention over long sequences, while Conditional VAEs [19] allow for property-guided generation by conditioning the latent space on molecular attributes. GrammarVAE further incorporates context-free grammar rules into the decoding process, improving validity by enforcing syntactic correctness.

GAN-based methods like ORGAN apply adversarial training and reinforcement learning to SMILES generation, optimizing for target-specific properties via reward signals. Similarly, Reinvent [30] blends SMILES-based generation with retrieval mechanisms to improve synthetic feasibility and diversity. Despite their flexibility and speed, sequence-based methods tend to have difficulty modeling molecular graphs' complex structural dependencies and often require post-processing to ensure chemical validity.

2.3 Tree-based methods

Tree-based approaches decompose molecules into hierarchies of chemical substructures or motifs and model their generation as an assembly of these components. This paradigm aims to guarantee the syntactic and chemical validity of generated molecules by working at the subgraph level.

The Junction Tree Variational Autoencoder (JT-VAE) represents them as junction trees, where each node is a chemical motif. By combining tree-structured generation with graph decoding, JT-VAE guarantees 100% validity while also enabling structured property optimization. An extension [15] incorporates conditional generation by conditioning motif assembly on target properties, supporting controllable molecular design.

Tree-based models are particularly effective in generating synthetically feasible and chemically realistic compounds. However, they also introduce complexity in training and inference due to the need for substructure extraction, motif vocabulary construction, and hierarchical decoding. Additionally, they may lack flexibility in discovering entirely novel molecular motifs outside the predefined set.

2.4 Emerging directions and meta-learning for molecular generation

Beyond the three dominant paradigms, newer approaches are emerging to address specific challenges in molecular generation. Diffusion models, such as PIDiff, generate 3D molecular structures through iterative denoising, proving especially valuable for tasks like protein-ligand docking. Retrieval-based generation methods integrate database search with generative modeling to ensure synthetic accessibility and real-world relevance.

In parallel, methods focusing on multi-property control and task-specific compound design have gained increasing attention. MGCVAE enables multi-objective inverse design using conditional graph VAEs, while deep generative frameworks such as those proposed by Zhavoronkov et al. [37] have shown notable success in discovering target-specific compounds.

Among these emerging directions, meta-learning has become a particularly promising framework for few-shot molecular generation, especially

in data-scarce scenarios such as orphan drug discovery. First-order meta-learning algorithms like Reptile offer a computationally efficient alternative to gradient-based methods such as MAML [7], by optimizing task-agnostic initializations through repeated adaptation across tasks without requiring second-order derivatives. Such approaches enable fast generalization using only a handful of examples.

In the molecular domain, meta-learning is gaining momentum as a solution to the data inefficiency of conventional deep models. Hospedales et al. [12] highlight its potential in structured prediction problems, although many existing works still lack explicit integration with molecular graph or motif-based representations.

To this end, our proposed framework, MetaMolGen, adopts a meta-learning strategy tailored to few-shot molecular generation. By combining structural priors with task-adaptive initialization through Reptile, it improves generalization, stability, and sample efficiency in low-resource molecular design scenarios, bridging the gap between data scarcity and effective generative modeling.

3 Preliminaries

3.1 Notations

We consider a molecular dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where each $x_i \in \mathbb{R}^d$ denotes a molecular feature vector and y_i is the corresponding SMILES string (see Section 4.2 for details). The molecular generation process is formulated as learning a mapping function $f_\theta : \mathbb{R}^d \to \mathcal{Y}$, where $\mathcal{Y}$ denotes the space of valid molecular sequences over a vocabulary $\Sigma = \{\texttt{C}, \texttt{N}, \texttt{O}, \texttt{=}, \ldots\}$. Each element in $\mathcal{Y}$ corresponds to a valid tokenized SMILES string, where tokens represent atoms, bonds, or structural symbols.

In meta-learning settings, we assume a distribution over tasks $\mathcal{P}(\mathcal{T})$, where each task $\mathcal{T}_m \sim \mathcal{P}(\mathcal{T})$ consists of a support set $\mathcal{D}_m^{\text{support}}$ and a query set $\mathcal{D}_m^{\text{query}}$. The meta-learning objective is denoted by $\mathcal{L}_{\text{meta}}$, and the model parameters are updated from initial parameters θ to task-specific parameters θ'_m via a few adaptation steps.

3.2 Problem statement

We consider a conditional molecular generation task, where the goal is to generate syntactically valid and property-compliant SMILES sequences given molecular features and target property constraints.

Formally, given a dataset

$$\mathcal{D} = \{(x_i, z_i, y_i)\}_{i=1}^{N},$$

where $x_i \in \mathbb{R}^d$ denotes the molecular feature vector, $z_i \in \mathbb{R}^k$ represents the desired property vector, and $y_i \in \mathcal{Y} \subset \Sigma^*$ is a tokenized SMILES sequence, the objective is to learn a conditional generative function: $f_\theta : (x, z) \mapsto \hat{y}$ that generates sequences $\hat{y}$ that are both chemically valid and aligned with the desired properties.

To ensure numerical stability and consistent feature representation, molecular descriptors are standardized using a dataset-dependent affine transformation:

$$X' = \Sigma^{-1/2}(X - \mu), \quad \text{so that} \quad X' \sim \mathcal{N}(0, I) \tag{1}$$

where the empirical mean and variance are estimated as:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} X_i, \quad \hat{\sigma} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{\mu})^2} \tag{2}$$

During training, the model is optimized via a token-level reconstruction loss to maximize the likelihood of producing ground-truth sequences:

$$\mathcal{L}_{\text{recon}} = \text{CrossEntropy}(\hat{y}, y) \tag{3}$$

For fast task adaptation, the model employs a first-order gradient-based update strategy. For each task $\mathcal{T}_m \sim p(\mathcal{T})$, model parameters are adapted by applying a small number of gradient steps on the support set. A single-step update is expressed as:

$$\theta'_m = \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{D}_m^{\text{support}}; \theta), \tag{4}$$

where $\alpha > 0$ is the inner-loop learning rate. These task-specific parameters θ'_m are then used to compute the Reptile-style meta-update:

$$\theta \leftarrow \theta + \epsilon \cdot \frac{1}{M} \sum_{m=1}^{M} (\theta'_m - \theta), \tag{5}$$

where ϵ is the outer-loop step size. This update rule moves the initialization towards parameters that perform well after a small number of task-specific updates, enabling fast adaptation without requiring second-order gradients. This adaptation rule defines the core mechanism underlying task-level generalization in MetaMolGen, and serves as the basis for our theoretical discussion in Section 5.

At inference time, molecular sequences are generated autoregressively by modeling the conditional distribution:

$$P_\theta(y \mid x, z) = \prod_{t=1}^{T} P_\theta(y_t \mid y_{<t}, x, z) \tag{6}$$

which ensures syntactic *validity* by learning token transitions compliant with SMILES grammar. To promote output *diversity*, tokens are sampled from a temperature-controlled softmax distribution:

$$y_t \sim \text{Categorical}\left(\text{softmax}\left(\frac{f_\theta(y_{<t}, x, z)}{\tau}\right)\right) \tag{7}$$

These components jointly support the generation of valid, diverse, and property-aligned molecular structures under limited supervision.

4 Method

MetaMolGen is a meta-learned molecular generator designed for few-shot molecular generation and multi-objective optimization. The training framework of MetaMolGen consists of motif extraction, motif prediction, and molecular graph assembly. As shown in Fig. 2, unlike traditional generative models, it leverages the Reptile algorithm [21] for rapid task adaptation, Conditional Neural Processes (CNPs) for task-aware molecular represen-

tations, and feature standardization for stable training.

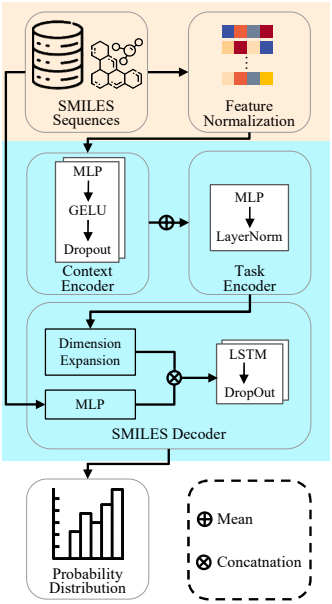


Figure 2. Overview of the MetaMolGen training process.

The framework processes molecular data through a structured pipeline. Molecular descriptors are first standardized for stability (Eq. (1), (2)) and then encoded into a task embedding via a context encoder. To enhance structural diversity, MetaMolGen leverages Conditional Neural Processes (CNPs) to capture task-level uncertainty through latent representations conditioned on support sets, where the resulting predictive distribution is formalized as $p(y \mid x, \mathcal{C}) = \int p(y \mid x, z)\, q(z \mid \mathcal{C})\, dz$, with z sampled from the context-dependent posterior $q(z \mid \mathcal{C})$.

The learning process in MetaMolGen is based on a meta-learning formulation composed of task-specific adaptation and meta-level optimization. For fast task adaptation, the model employs a first-order gradient-based update strategy. For each task $\mathcal{T}_m \sim p(\mathcal{T})$, model parameters are adapted using the support set through one or more gradient steps, as defined in Eq. (4), where $\alpha > 0$ is the inner-loop learning rate. The resulting task-adapted parameters θ'_m serve as intermediate optima that guide the subsequent meta-update. The Reptile algorithm updates the shared initialization θ by averaging the differences between the adapted and original parameters across tasks, as shown in Eq. (5), where ϵ is the outer-loop step size and M denotes the number of sampled tasks. This first-order update enables fast adaptation to novel molecular generation tasks, without relying on second-order derivatives or explicit backpropagation through the adaptation process. It forms the foundation for efficient few-shot generalization in MetaMolGen,

as further elaborated in Section 5.

Sequence generation is performed by an autoregressive decoder, which factorizes the conditional distribution over output tokens as in Eq. (6). This formulation enforces syntactic validity by modeling token transitions aligned with SMILES grammar. To encourage diverse outputs, decoding is performed via temperature-controlled sampling (Eq. (7)).

By integrating these components into a coherent pipeline, MetaMolGen effectively balances syntactic validity, structural diversity, and generalization performance, making it well-suited for applications in drug discovery, materials science, and beyond.

4.1 Feature standardization for molecular representation learning

To improve training stability and generalization in few-shot molecular generation, MetaMolGen incorporates a learnable normalization layer that standardizes molecular descriptors before task encoding. For each input feature x_i, the model applies an affine transformation:

$$x_i' = \frac{x_i - \mu_\theta}{\sigma_\theta + \varepsilon}, \tag{8}$$

where μ_θ and σ_θ are trainable feature-wise statistics, and ε is a small constant to ensure numerical stability. This transformation mitigates input scale disparities and ensures consistent feature distributions across tasks.

Feature normalization plays a critical role in stabilizing optimization and enhancing generalization in few-shot molecular generation. By applying a learnable affine transformation to standardize input features (Eq. (8)), the model mitigates scale imbalance and ensures consistent input distributions across tasks. This reduces sensitivity to outlier features and smooths the optimization landscape, effectively suppressing gradient oscillations during meta-updates, as analyzed in Theorem 2. A visual comparison of training loss trajectories with and without feature standardization is provided in Figure 7.

Normalization further improves the conditioning of the loss surface, enabling faster and more stable convergence (Theorem 3). On the gen-

eralization side, by reducing the empirical loss variance across tasks, it tightens the PAC-Bayes generalization bound (Theorem 4) and helps the model capture task-invariant patterns. Together, these effects make feature normalization a fundamental component of MetaMolGen, supporting both learning stability and transferability under distributional shifts.

4.2 Molecular representation

This study adopts the SMILES (Simplified Molecular Input Line Entry System) format to represent molecular structures as linear strings. SMILES encodes molecular graphs using a depth-first traversal, translating atoms, bonds, branches, and rings into a sequence of standardized characters. For example, the molecule ethanol is represented as "CCO," where two carbon atoms are followed by a hydroxyl group. This compact format preserves the topological structure of molecules and is naturally compatible with sequence-based generative models.

In MetaMolGen, SMILES sequences are used as the target output of the decoder. We construct a vocabulary containing special tokens such as {`PAD`}, {`START`}, {`EOS`}, and {`UNK`}, and tokenize each SMILES string into a sequence of discrete tokens. These tokenized sequences are embedded and fed into an LSTM-based decoder, which autoregressively predicts the next token given the previously generated ones. During training, the model learns the correspondence between token sequences and valid molecular syntax, enabling it to generate chemically plausible molecules in a character-by-character manner [3, 23].

4.3 Conditional neural processes

Conditional Neural Processes provide a scalable alternative to Gaussian Processes and Conditional Variational Autoencoders by learning conditional distributions in a data-driven manner, avoiding restrictive priors [10, 16]. They generalize well in low-data settings while maintaining computational efficiency with $O(n + m)$ complexity.

CNPs define a conditional stochastic process Q_θ over function values

$f(x)$ given observations O:

$$Q_\theta(f(T) \mid O, T) = \prod_{x \in T} Q_\theta(f(x) \mid O, x).$$

The encoder h_θ maps observations (x_i, y_i) into latent embeddings:

$$r_i = h_\theta(x_i, y_i), \quad r = \frac{1}{n} \sum_{i=1}^{n} r_i.$$

The decoder g_θ then predicts parameters ϕ_i for each target point $x_i \in T$:

$$\phi_i = g_\theta(x_i, r), \quad f(x_i) \sim \mathcal{N}(\mu_i, \sigma_i^2).$$

CNPs are trained by minimizing the negative log-likelihood over randomly selected subsets of observations:

$$\mathcal{L}(\theta) = -\mathbb{E}_{f \sim P} \left[\mathbb{E}_N \log Q_\theta \left(\{y_i\}_{i=0}^{n-1} \mid O_N, \{x_i\}_{i=0}^{n-1} \right) \right].$$

Monte Carlo methods and stochastic gradient descent (SGD) are used for optimization.

While CNPs offer efficient adaptation, they lack explicit uncertainty quantification, which can lead to overconfidence [8, 14]. Effective subset selection strategies, such as adaptive sampling and curriculum learning, help mitigate variance and overfitting . CNPs are particularly well-suited for tasks requiring fast adaptation, such as online learning and molecular generation [26].

4.4 Reptile meta-learning

Reptile [21] is a simple yet effective first-order meta-learning algorithm designed for rapid adaptation to new tasks using limited data. Unlike Model-Agnostic Meta-Learning (MAML) [7], which optimizes an initialization for sensitivity to few gradient steps and often requires computing second-order gradients, Reptile works by repeatedly sampling a task, training on it for multiple steps using a standard optimizer, and then moving the initial model parameters towards the adapted parameters found for

that task. This process implicitly finds a parameter initialization that is, on average, close to the optimal parameters for various tasks within the distribution, facilitating fast adaptation.

Given a task distribution $\mathcal{P}(\mathcal{T})$, each task $\mathcal{T}_m \sim \mathcal{P}(\mathcal{T})$ consists of a support set:

$$\mathcal{D}_m^{\text{support}} = \{(x_i, y_i)\}_{i=1}^{N_s}$$

and potentially a query set for evaluation (though not directly used in the Reptile update rule):

$$\mathcal{D}_m^{\text{query}} = \{(x_j, y_j)\}_{j=1}^{N_q}.$$

For each task $\mathcal{T}_m$, the model parameters θ are updated for k steps using the support set $\mathcal{D}_m^{\text{support}}$ with a standard optimizer (like SGD or Adam) and an inner-loop learning rate α. Let $U_{\mathcal{T}_m}^k(\theta)$ denote the parameters obtained after performing k optimization steps on task $\mathcal{T}_m$ starting from θ. The task-adapted parameters are thus $\theta'_m = U_{\mathcal{T}_m}^k(\theta)$.

The meta-update rule for Reptile involves moving the initial parameters θ towards the task-adapted parameters θ'_m. Averaged over a batch of M tasks, the update is:

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{m=1}^{M} (\theta'_m - \theta),$$

where ϵ is the outer-loop (meta) step size. This update can be interpreted as performing SGD on the objective $\mathbb{E}_{\mathcal{T}\sim\mathcal{P}(\mathcal{T})}[\frac{1}{2}||\theta - U_{\mathcal{T}}^k(\theta)||^2]$.

Reptile's simplicity and effectiveness make it suitable for molecular generation, where adapting to new molecular property distributions or chemical spaces is crucial. Integrating Reptile into CNPs aims to enhance generalization with minimal fine-tuning, as outlined in Algorithm 1.

5 Theoretical analysis

In this section, we provide a theoretical analysis of the proposed Meta-MolGen model, focusing on its convergence properties and generalization

Algorithm 1 MetaMolGen Parameter Optimization via Reptile

Require: Task distribution $\mathcal{P}(\mathcal{T})$, iteration number T, task batch size M
Require: Inner steps k, inner learning rate α, outer step size ϵ
Require: Initial parameters $\theta^{(0)}$
Ensure: Final meta-learned parameters θ^*

Initialize $\theta \leftarrow \theta^{(0)}$
for $t = 1$ to T **do**
  Sample tasks $\{\mathcal{T}_m\}_{m=1}^{M} \sim \mathcal{P}(\mathcal{T})$
  Initialize $\Delta\theta \leftarrow 0$
  for each $\mathcal{T}_m$ **do**
    Sample support set $\mathcal{D}_m^{\text{support}}$
    $\theta_m \leftarrow \theta$
    for $i = 1$ to k **do**
      $\mathcal{L}_m \leftarrow \mathcal{L}(\mathcal{D}_m^{\text{support}}; \theta_m)$
      $\theta_m \leftarrow \theta_m - \alpha \nabla_\theta \mathcal{L}_m$
    $\Delta\theta \leftarrow \Delta\theta + (\theta_m - \theta)$
  $\theta \leftarrow \theta + \epsilon \cdot \Delta\theta / M$
return θ

guarantees. We establish conditions under which the training loss decreases and derive a comprehensive error bound to quantify performance. This section integrates all formal proofs for completeness.

5.1 Assumptions and preliminaries

We first introduce the fundamental assumptions and key lemmas that underlie our theoretical analysis.

Assumption 1 (Lipschitz Smoothness [22])**.** *Let the task loss $L_T(\theta)$ and the meta-learning objective $\mathcal{L}(\theta)$ be differentiable functions with L-Lipschitz continuous gradients. Then, for all $\theta_1, \theta_2 \in \mathbb{R}^d$,*

$$\|\nabla L_T(\theta_1) - \nabla L_T(\theta_2)\| \leq L\|\theta_1 - \theta_2\|,$$
$$\|\nabla \mathcal{L}(\theta_1) - \nabla \mathcal{L}(\theta_2)\| \leq L\|\theta_1 - \theta_2\|.$$

Assumption 2. *Let $L_T(\theta)$ denote the task loss function and the expected*

loss $\mathbb{E}_{T\sim p(T)}[L_T(\theta)]$ *is* μ*-strongly convex. Then, for all* $\theta \in \mathbb{R}^d$*,*

$$L_T(\theta) \geq L_T(\theta^*) + \frac{\mu}{2}\|\theta - \theta^*\|^2,$$

where θ^* *is the unique minimizer.*

We also introduce two key lemmas that serve as foundational tools for our proofs.

Lemma 1 (Descent Lemma [2])**.** *If* $L_T(\theta)$ *is* L*-smooth, then for any learning rate* $\alpha > 0$*, the loss after a gradient update satisfies:*

$$L_T(\theta') \leq L_T(\theta) - \alpha\|\nabla L_T(\theta)\|^2 + \frac{L\alpha^2}{2}\|\nabla L_T(\theta)\|^2.$$

Lemma 2 (Stability of Task Encoder [4])**.** *Assume the task encoder* r_{task} *in MetaMolGen is Lipschitz smooth with constant* L_r*, i.e., for any two datasets* D_1 *and* D_2*:*

$$\|r_{task}(D_1) - r_{task}(D_2)\| \leq L_r\|D_1 - D_2\|.$$

5.2 Convergence analysis

To analyze the convergence behavior of MetaMolGen, we establish key results on training stability and the benefits of normalization. Theorem 1 proves descent with an appropriate learning rate, while Theorem 2 and Theorem 3 demonstrate that normalization reduces gradient variance and improves loss surface conditioning, thereby supporting faster and more robust convergence.

Theorem 1. *Let* $L_T(\theta)$ *be an* L*-smooth loss function. If the learning rate* α *satisfies* $0 < \alpha \leq \frac{2}{L}$*, then the sequence of losses* $\{L_T(\theta_k)\}$ *is monotonically decreasing, where*

$$\theta_{k+1} = \theta_k - \alpha\nabla_\theta L_T(\theta_k).$$

Proof. Since $L_T(\theta)$ is assumed to be L-smooth, the Descent Lemma 1

applies:

$$L_T(\theta_{k+1}) \leq L_T(\theta_k) - \alpha \|\nabla L_T(\theta_k)\|^2 + \frac{L\alpha^2}{2} \|\nabla L_T(\theta_k)\|^2 \tag{9}$$

$$\leq L_T(\theta_k) - \left(\alpha - \frac{L\alpha^2}{2}\right) \|\nabla L_T(\theta_k)\|^2. \tag{10}$$

To ensure descent at each iteration, the term multiplying the squared gradient norm must be non-negative:

$$\alpha \left(1 - \frac{L\alpha}{2}\right) \geq 0 \quad \Rightarrow \quad \alpha \leq \frac{2}{L}. \tag{11}$$

Thus, the loss is non-increasing under the given condition, which completes the proof. ■

Theorem 2. *Let $X \in \mathbb{R}^d$ be the input molecular descriptors, $X' = \frac{X - \hat{\mu}}{\hat{\sigma} + \epsilon}$ denote their normalized version. Let θ and θ' be model parameters obtained by training on X and X' respectively. Then,*

$$\mathrm{Var}(\nabla_\theta L_T(\theta')) \leq \mathrm{Var}(\nabla_\theta L_T(\theta)).$$

Proof. The gradient variance is defined as $\mathrm{Var}(\nabla_\theta L_T) = \mathbb{E}[\|\nabla_\theta L_T\|^2] - \|\mathbb{E}[\nabla_\theta L_T]\|^2$. Assuming L_T is twice differentiable, the expected squared norm of the gradient can be approximated by the trace of the product of the Hessian and the input covariance matrix. If the Hessian has eigenvalues λ_j and the input variance per feature is σ_j^2, then:

$$\mathbb{E}[\|\nabla_\theta L_T\|^2] \approx \sum_{j=1}^{d} \lambda_j \sigma_j^2. \tag{12}$$

After normalization, the feature variances are scaled towards unity ($\sigma_j^2 \approx 1$). If the original variances σ_j^2 were significantly larger than 1, normalization reduces this sum, and consequently, the gradient variance. ■

Theorem 3. *Let $L_T(\theta)$ be twice differentiable, with Hessian $H = \nabla_\theta^2 L_T(\theta)$ and condition number $\kappa(H) = \lambda_{\max}(H)/\lambda_{\min}(H)$. Consider normalized inputs $X' = (X - \hat{\mu})/(\hat{\sigma} + \epsilon)$, and let $L'_T(\theta)$ be the corresponding loss with*

Hessian $H' = \nabla_\theta^2 L'_T(\theta)$*. Then,* $\kappa(H') \ll \kappa(H)$*, and the iteration complexity of first-order optimization improves from* $O(\kappa(H)\log\frac{1}{\epsilon})$ *to* $O(\log\frac{1}{\epsilon})$*.*

Proof. Let $x \in \mathbb{R}^d$ denote the input features and define the empirical loss as $L_T(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_\theta(x^{(i)}), y^{(i)})$, where f_θ is a differentiable model and ℓ is a convex, twice-differentiable loss function. The Hessian is $H = \nabla_\theta^2 L_T(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 \ell(f_\theta(x^{(i)}), y^{(i)})$.

If the input second-moment matrix $\Sigma_x = \mathbb{E}[xx^\top]$ is ill-conditioned, this anisotropy propagates through the model's Jacobian $J_\theta(x) := \nabla_\theta f_\theta(x)$ to the Hessian, leading to $\kappa(H) \gg 1$. Feature normalization transforms inputs such that the new covariance matrix $\mathbb{E}[x'x'^\top]$ approaches the identity matrix. This makes the expected Jacobian structure more uniform, leading to a better-conditioned Hessian where $\kappa(H') \approx 1$. The iteration complexity of gradient descent for minimizing a strongly convex function with condition number κ is $T = O(\kappa\log\frac{1}{\epsilon})$. After normalization, this improves from $O(\kappa(H)\log\frac{1}{\epsilon})$ to $O(\kappa(H')\log\frac{1}{\epsilon}) \approx O(\log\frac{1}{\epsilon})$, completing the proof. ■

5.3 Generalization and error analysis

For generalization analysis, we present several critical results. Theorem 4 shows that normalization reduces meta-generalization error. Theorem 5 and Theorem 6 address properties of the stochastic optimization process. Finally, Theorem 7 provides a unified bound decomposing the expected loss into its primary components.

Theorem 4. *Let* $X \in \mathbb{R}^d$ *be molecular descriptors,* $\mathcal{E}_{meta,\ raw}$*,* $\mathcal{E}_{meta,\ std}$ *be the meta-generalization errors under training on* X *and its normalized form* $X' = (X - \hat{\mu})/(\hat{\sigma} + \epsilon)$*, respectively. Then,*

$$\mathbb{E}[\mathcal{E}_{meta,\ std}] < \mathbb{E}[\mathcal{E}_{meta,\ raw}].$$

Proof. According to PAC-Bayes generalization theory [20], for a learned posterior distribution Q over hypotheses and a prior P, the expected gen-

eralization error is bounded by:

$$\mathbb{E}[\mathcal{E}_{\text{meta}}] \leq \mathbb{E}[\mathcal{L}_{\text{train}}] + O\left(\sqrt{\frac{D_{\text{KL}}(Q\|P) + \log(1/\delta)}{N}}\right), \tag{13}$$

where N is the number of meta-training tasks. If Q is modeled as a Gaussian centered at the empirical minimizer, the KL divergence $D_{\text{KL}}(Q\|P)$ scales with the empirical parameter variance, which is related to the variance of the loss across tasks, σ^2. Since the loss function is Lipschitz-continuous with respect to its inputs, reducing feature variance via normalization also reduces the variance of the loss across different tasks, i.e., $\sigma^2_{\text{std}} < \sigma^2_{\text{raw}}$. A smaller loss variance leads to a smaller KL divergence, which in turn tightens the generalization bound. Consequently, $\mathbb{E}[\mathcal{E}_{\text{meta, std}}] < \mathbb{E}[\mathcal{E}_{\text{meta, raw}}]$. ■

Theorem 5. *Let $\mathcal{D}_T = \{(x_i, z_i, y_i)\}_{i=1}^{N_T}$ be the dataset for task T. Let $\mathcal{B} \subset \mathcal{D}_T$ be a mini-batch sampled uniformly. The stochastic gradient estimator $\nabla_{est} L_T(\theta) = \frac{1}{|\mathcal{B}|} \sum_{(x_i,z_i,y_i)\in\mathcal{B}} \nabla_\theta \mathcal{L}(f_\theta(x_i, z_i), y_i)$ is unbiased:*

$$\mathbb{E}_{\mathcal{B}}[\nabla_{est} L_T(\theta)] = \nabla_\theta L_T(\theta).$$

Proof. Let the per-sample gradient be $g_i(\theta) := \nabla_\theta \mathcal{L}(f_\theta(x_i, z_i), y_i)$. The full-batch gradient is $\nabla_\theta L_T(\theta) = \frac{1}{N_T} \sum_{i=1}^{N_T} g_i(\theta)$. By linearity of expectation,

$$\mathbb{E}_{\mathcal{B}}[\nabla_{\text{est}} L_T(\theta)] = \mathbb{E}_{\mathcal{B}}\left[\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} g_i(\theta)\right] = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \mathbb{E}_i[g_i(\theta)]. \tag{14}$$

Since each sample is drawn uniformly, the expectation of any single term is the average over all possible per-sample gradients:

$$\mathbb{E}_i[g_i(\theta)] = \frac{1}{N_T}\sum_{k=1}^{N_T} g_k(\theta) = \nabla_\theta L_T(\theta). \tag{15}$$

Substituting this back, we get:

$$\mathbb{E}_{\mathcal{B}}[\nabla_{\text{est}} L_T(\theta)] = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta L_T(\theta) = \nabla_\theta L_T(\theta). \tag{16}$$

Thus, the stochastic gradient estimator is an unbiased estimator of the true gradient. ■

Theorem 6. *Let $L_T(\theta)$ be an L-smooth and μ-strongly convex loss function, and the stochastic gradient estimator be unbiased with bounded variance $\mathbb{E}[\|\nabla_{est} L_T(\theta_k) - \nabla L_T(\theta_k)\|^2] \leq \sigma^2$. Let θ_k be the parameters after k iterations of SGD with a learning rate α_k, and θ^* be the minimizer of $L_T(\theta)$. Then for a constant step size $\alpha \leq 1/L$, the expected suboptimality satisfies:*

$$\mathbb{E}[\|\theta_k - \theta^*\|^2] \leq (1 - \mu\alpha)^k \|\theta_0 - \theta^*\|^2 + \frac{\alpha\sigma^2}{\mu}.$$

This implies that the algorithm converges to a neighborhood of the optimum, and for non-convex objectives, the expected squared gradient norm diminishes as $\frac{1}{k}\sum_{i=0}^{k-1} \mathbb{E}[\|\nabla L_T(\theta_i)\|^2] = O(1/k)$.

Proof. We analyze the distance to the optimum at step $k+1$:

$$\begin{aligned}\|\theta_{k+1} - \theta^*\|^2 &= \|\theta_k - \alpha_k \nabla_{\text{est}} L_T(\theta_k) - \theta^*\|^2 \\ &= \|\theta_k - \theta^*\|^2 - 2\alpha_k \langle \theta_k - \theta^*, \nabla_{\text{est}} L_T(\theta_k)\rangle + \alpha_k^2 \|\nabla_{\text{est}} L_T(\theta_k)\|^2. \end{aligned} \tag{17}$$

Taking the expectation conditioned on θ_k (denoted by $\mathbb{E}_k$), and using the unbiasedness of the stochastic gradient ($\mathbb{E}_k[\nabla_{\text{est}} L_T(\theta_k)] = \nabla L_T(\theta_k)$):

$$\begin{aligned}\mathbb{E}_k[\|\theta_{k+1} - \theta^*\|^2] =& \|\theta_k - \theta^*\|^2 - 2\alpha_k \langle \theta_k - \theta^*, \nabla L_T(\theta_k)\rangle \\ &+ \alpha_k^2 \mathbb{E}_k[\|\nabla_{\text{est}} L_T(\theta_k)\|^2]. \end{aligned} \tag{18}$$

Using the bias-variance decomposition, $\mathbb{E}_k[\|\nabla_{\text{est}}\|^2] = \|\nabla\|^2 + \text{Var}_k(\nabla_{\text{est}})$. With the bounded variance assumption, this becomes

$$\mathbb{E}_k[\|\nabla_{\text{est}}\|^2] \leq \|\nabla L_T(\theta_k)\|^2 + \sigma^2 \tag{19}$$

. For a μ-strongly convex function, we have the property $\langle \nabla L_T(\theta_k), \theta_k -$

$\theta^*\rangle \geq \mu\|\theta_k - \theta^*\|^2$. Substituting these into Eq. (18):

$$\begin{aligned}\mathbb{E}_k[\|\theta_{k+1} - \theta^*\|^2] \leq \|\theta_k - \theta^*\|^2 - 2\alpha_k\mu\|\theta_k - \theta^*\|^2 \\ + \alpha_k^2(\|\nabla L_T(\theta_k)\|^2 + \sigma^2).\end{aligned} \tag{20}$$

Assuming L-smoothness implies $\|\nabla L_T(\theta_k)\|^2 \leq 2L(L_T(\theta_k) - L_T(\theta^*))$, which is still complex. A more direct approach for strongly convex functions is to use the property $\|\nabla L_T(\theta_k)\|^2 \leq L^2\|\theta_k - \theta^*\|^2$. For a constant step size $\alpha \leq 1/L$, and taking the full expectation:

$$\mathbb{E}[\|\theta_{k+1} - \theta^*\|^2] \leq (1 - 2\alpha\mu + \alpha^2L^2)\mathbb{E}[\|\theta_k - \theta^*\|^2] + \alpha^2\sigma^2. \tag{21}$$

With a small enough constant α, the term $(1-2\alpha\mu+\alpha^2L^2)$ can be bounded by $(1-\alpha\mu)$. Unrolling the recurrence relation over k steps yields the stated bound, showing linear convergence to a noise ball of radius $O(\alpha\sigma^2/\mu)$.

For the general non-convex but L-smooth case, we return to the Descent Lemma. From the proof of Theorem 1, we had:

$$\begin{aligned}\mathbb{E}[L_T(\theta_{k+1})] \leq \mathbb{E}[L_T(\theta_k)] \\ - \alpha\left(1 - \frac{L\alpha}{2}\right)\mathbb{E}[\|\nabla L_T(\theta_k)\|^2] + \frac{L\alpha^2}{2}\sigma^2.\end{aligned} \tag{22}$$

Rearranging and setting $\alpha = 1/L$:

$$\frac{\alpha}{2}\mathbb{E}[\|\nabla L_T(\theta_k)\|^2] \leq \mathbb{E}[L_T(\theta_k)] - \mathbb{E}[L_T(\theta_{k+1})] + \frac{L\alpha^2}{2}\sigma^2. \tag{23}$$

Summing from $k = 0$ to $K - 1$ (telescoping sum):

$$\begin{aligned}\frac{\alpha}{2}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla L_T(\theta_k)\|^2] &\leq L_T(\theta_0) - \mathbb{E}[L_T(\theta_K)] + \frac{KL\alpha^2}{2}\sigma^2 \\ &\leq L_T(\theta_0) - L_T(\theta^*) + \frac{KL\alpha^2}{2}\sigma^2.\end{aligned} \tag{24}$$

Dividing by K and $\alpha/2$, we get the average squared gradient norm:

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla L_T(\theta_k)\|^2] &\leq \frac{2(L_T(\theta_0)-L_T(\theta^*))}{K\alpha}+L\alpha\sigma^2 \\ &= O\left(\frac{1}{K}\right).\end{aligned} \tag{25}$$

This guarantees that SGD finds a point with a small gradient on average, which is the standard convergence result for non-convex optimization. ■

Theorem 7. *Let θ_k be the model parameters after k updates, and θ^* be the optimal parameters minimizing the true expected loss over the data distribution. The total expected error satisfies:*

$$\begin{aligned}\mathbb{E}_{T\sim p(T)}[L_T(\theta_k)]-L_T(\theta^*) \leq\ & \underbrace{O\left(\frac{L}{k}\right)}_{\textit{Optimization Error}} + \underbrace{O\left(\sqrt{\frac{D_{KL}(Q\|P)}{N}}\right)}_{\textit{Generalization Error}} \\ &+ \underbrace{O\left(\frac{\sigma}{\sqrt{B}}\right)}_{\textit{Gradient Noise}} + \underbrace{O(\epsilon_{approx})}_{\textit{Approximation Error}}.\end{aligned}$$

Proof. The total expected loss gap $\mathcal{E}(\theta_k) = \mathbb{E}_{T\sim p(T)}[L_T(\theta_k)] - L_T(\theta^*)$, can be analyzed by decomposing it into several constituent error terms. We introduce two intermediate quantities: $\theta^*_{\mathcal{F}} = \arg\inf_{\theta\in\mathcal{F}} \mathbb{E}_T[L_T(\theta)]$, the best parameters within our model class $\mathcal{F}$, and $\hat{\theta}^* = \arg\min_{\theta\in\mathcal{F}} \hat{L}_T(\theta)$, the empirical risk minimizer on the finite training set. The decomposition is as follows:

$$\begin{aligned}\mathcal{E}(\theta_k) = &\left(\mathbb{E}[L_T(\theta_k)] - \mathbb{E}[L_T(\hat{\theta}^*)]\right) \\ &+ \left(\mathbb{E}[L_T(\hat{\theta}^*)] - \mathbb{E}[L_T(\theta^*_{\mathcal{F}})]\right) + \left(\mathbb{E}[L_T(\theta^*_{\mathcal{F}})] - L_T(\theta^*)\right).\end{aligned} \tag{26}$$

The first term represents the optimization error, which measures how close our algorithm's output θ_k is to the empirical minimizer $\hat{\theta}^*$. As established in Theorem 6, the convergence of stochastic gradient methods ensures this error diminishes with the number of iterations k, typically bounded by $O(L/k)$ for L-smooth objectives. This term also implicitly contains the gradient noise from mini-batching, where the standard deviation of the parameter error scales as $O(\sigma/\sqrt{B})$ with batch size B.

The second term is the generalization error, quantifying the discrepancy between the empirical and true risk due to a finite sample of N tasks. We bound this using the PAC-Bayes framework [20]. For a learned posterior Q and a prior P, the generalization gap is bounded with high probability by $O(\sqrt{(D_{\mathrm{KL}}(Q\|P) + \log(N/\delta))/N})$, where $D_{\mathrm{KL}}(Q\|P)$ reflects the model complexity.

The final term, $\epsilon_{\mathrm{approx}} = \mathbb{E}[L_T(\theta^*_{\mathcal{F}})] - L_T(\theta^*)$, is the approximation error. This is the irreducible error inherent to the chosen model class $\mathcal{F}$, representing the gap between the best function in our hypothesis space and the true optimal function. This error is independent of the training process and can only be reduced by employing a more expressive model architecture.

Combining these distinct sources of error, we arrive at the unified bound. Each component highlights a different facet of the meta-learning challenge and points to specific avenues for improvement: increasing the number of training iterations (k), the number of meta-training tasks (N), the mini-batch size (B), or the expressive power of the model family ($\mathcal{F}$). ■

6 Experimental setup

This section details the datasets, baseline models, model configuration, and evaluation metrics used to validate MetaMolGen. All experiments are designed to assess molecular generation capabilities in low-data, few-shot scenarios.

6.1 Datasets and task construction

Our model is meta-trained on a large, diverse dataset and evaluated on tasks constructed from several standard benchmarks to assess generalization.

Meta-Training Source. We use a filtered subset of 100,000 molecules from **ChEMBL** [9] (molecular weight ≤ 500 Da, QED ≥ 0.5) as the source for meta-training tasks.

Few-Shot Tasks. For evaluation, we construct few-shot tasks from three benchmarks. **QM9** [27] is a dataset of approximately 134,000 small organic molecules; we use a subset of 30,000 molecules to create tasks focused on quantum chemical properties. **ZINC** [13] is a database of commercially available compounds; we use a subset of 50,000 molecules and cluster molecules by Morgan fingerprints to create structurally diverse tasks that test generalization to new scaffolds. **MOSES** [25] is a standardized benchmarking platform; we use its 60,000-molecule set to evaluate performance on standard metrics such as validity and novelty. For all datasets, molecules are preprocessed using RDKit for sanitization and canonicalization. A summary of the processed datasets is provided in Table 1.

Table 1. Benchmark dataset statistics after preprocessing.

Dataset	# Mol.	Avg. SMILES Length	Unique Scaffolds
ChEMBL	100,000	27.4	15,320
QM9	30,000	18.6	3,802
ZINC	50,000	24.1	7,632
MOSES	60,000	25.3	9,105

6.2 Baselines and comparison protocol

MetaMolGen is benchmarked against a diverse set of representative molecular generation models covering different architectural paradigms.

- **MolGAN** [6]: A graph-based model combining a Generative Adversarial Network (GAN) with Reinforcement Learning (RL).
- **GraphVAE** [17]: A variational autoencoder operating directly on molecular graphs using graph convolutional networks.
- **CharRNN** [31]: A character-level Recurrent Neural Network (GRU-based) that autoregressively generates SMILES strings.
- **TransformerVAE** [19]: A SMILES-based variational autoencoder employing a Transformer architecture with masked attention.

Fair Comparison Protocol: To ensure a fair comparison, all baselines were adapted to our few-shot protocol. They were first pre-trained on the

same ChEMBL dataset and then fine-tuned on the task-specific support sets using their respective optimization strategies (e.g., reconstruction loss, adversarial training). All models were exposed to the same number of training episodes and molecules.

6.3 MetaMolGen configuration

Our model generates molecular sequences under the few-shot settings outlined in Algorithm 2. It comprises a feature normalization layer, a context encoder, a task encoder, and a SMILES decoder.

Algorithm 2 MetaMolGen Forward Molecular Generation Process

Require: Context features $X \in \mathbb{R}^{B\times N\times d}$, input SMILES $S_{\text{in}} \in \mathbb{N}^{B\times L}$, model parameters θ

Ensure: Predicted token logits $P \in \mathbb{R}^{B\times L\times V}$

1: Normalize features: $\hat{X} \leftarrow \dfrac{X-\mu}{\sigma+\epsilon}$

2: Encode context points: $H \leftarrow \text{ContextEncoder}(\hat{X})$

3: Aggregate context: $r \leftarrow \text{mean}(H, \text{dim} = 1)$

4: Refine task representation: $r_{\text{task}} \leftarrow \text{TaskEncoder}(r)$

5: Generate SMILES logits: $P \leftarrow \text{SMILESDecoder}(S_{\text{in}}, r_{\text{task}})$ **return** P

6.3.1 Architecture design

The context encoder is a three-layer MLP (hidden and latent dimensions: 256 and 128) with GELU activations. The task representation is fed into a two-layer LSTM decoder (hidden/embedding dimension: 128) with layer normalization and dropout (0.2) to autoregressively generate SMILES sequences.

6.3.2 Meta-training and optimization

The model is meta-trained using the Reptile algorithm [21]. We use the Adam optimizer (meta-learning rate: 0.001, weight decay: 0.01) and train for 150 epochs. Each meta-update averages gradients from 10 tasks, with each task containing 16 molecules. As motivated by our theoretical analysis, input features are standardized to zero mean and unit variance within

each batch (Theorem 3), and we use a mini-batch size of 64 to reduce gradient estimation noise, whose effect on error scales as $O(\sigma/\sqrt{B})$ (Theorem 7).

6.4 Evaluation measures

To provide a holistic assessment, we define an *Overall Score* as the unweighted average of seven normalized metrics: (1) Validity, (2) Uniqueness, (3) Time efficiency, (4) Diversity, (5) Druglikeness (QED), (6) Synthesizability (SA Score), and (7) Solubility (logP).

Each metric is scaled to a [0, 1] range based on the minimum ($x^{\min}$) and maximum ($x^{\max}$) values observed across all models. For metrics where higher is better, we use Normalized = $(x-x^{\min})/(x^{\max}-x^{\min})$. For metrics where lower is better (Time), we use the inverted form Normalized = $(x^{\max} - x)/(x^{\max} - x^{\min})$. The final Overall Score is the mean of these normalized values:

$$\text{Overall Score} = \frac{1}{7}\sum_{i=1}^{7}\text{Normalized}_i \tag{27}$$

6.5 Ablation study

To assess the impact of the standardization layer, we compare MetaMolGen's performance on molecular generation across dataset sizes (1,000 to 60,000 molecules) with and without feature normalization. Results show that removing the standardization layer reduces validity, novelty, and diversity while increasing errors in property optimization, especially in low-data settings. This highlights its role in stabilizing feature distributions, improving generalization, and enhancing robustness in few-shot molecular generation.

For details on datasets, baselines, and hyperparameter configurations, please refer to `https://github.com/yzz980314/MetaMolGen`.

7 Experimental results and analysis

We present a comprehensive evaluation of MetaMolGen across multiple perspectives, including performance comparison with mainstream baselines, adaptability under low-resource settings, property-controllable molecule generation, and the impact of model design choices on training stability and robustness. Results consistently show that MetaMolGen not only outperforms baselines in key generation metrics but also exhibits stronger generalization and faster adaptation to new molecular design tasks.

Table 2 presents a detailed comparison of MetaMolGen against four baseline models (ORGAN [11], MolGAN [6], RNN [31], and MolGPT [1]) using a fixed training dataset of 5,000 molecules. The metrics include generation validity, uniqueness, diversity, drug-likeness, synthesizability, solubility, and overall performance. These results complement the main findings and are visualized in Figure 3.

Table 2. Comparison of molecular generation performance across models. Gray cells indicate metrics where MetaMolGen outperforms all baselines.

Objective	Algorithm	Valid (%)	Unique (%)	Time (h)	Diversity	Druglikeness	Synthesizability	Solubility	Overall Score
All/Alternated	ORGAN	96.1	97.2	10.2	0.92	0.52	0.71	0.53	0.4183
All/Simultaneously	MolGAN	97.4	2.4	2.12	0.91	0.47	0.84	0.65	0.5419
All/Simultaneously	MolGAN (QM9)	98.0	2.3	5.83	0.93	0.51	0.82	0.69	0.5356
Sequence-based	RNN	$48.40_{\pm 4.29}$	$99.18_{\pm 0.82}$	1.02	$0.8774_{\pm 0.0070}$	$0.5716_{\pm 0.0213}$	$0.8259_{\pm 0.0122}$	$0.7392_{\pm 0.0366}$	0.5690
Sequence-based	MolGPT	25.2	100.0	1.50	0.8637	0.5162	0.8213	0.6440	0.4836
All/Simultaneously	MetaMolGen	$75.40_{\pm 5.87}$	$99.92_{\pm 0.08}$	0.05	$0.8367_{\pm 0.0043}$	$0.8165_{\pm 0.0060}$	$0.8540_{\pm 0.0061}$	$0.8891_{\pm 0.0147}$	0.7085

7.1 Comparison with baseline models

MetaMolGen achieves the highest overall performance across models. While its validity is slightly lower than that of ORGAN and MolGAN, it excels in uniqueness, drug-likeness, synthesizability, solubility, and overall score. These results indicate that MetaMolGen provides a better balance between molecular property optimization, generation diversity, and efficiency. The reported standard errors were calculated by performing 10 independent generation runs with 500 molecules each, capturing the model's robustness and consistency across multiple runs.

The performance gain can be attributed to MetaMolGen's meta-learning framework, which enables efficient adaptation across different molecular tasks. Unlike ORGAN and MolGAN, which depend heavily on rein-

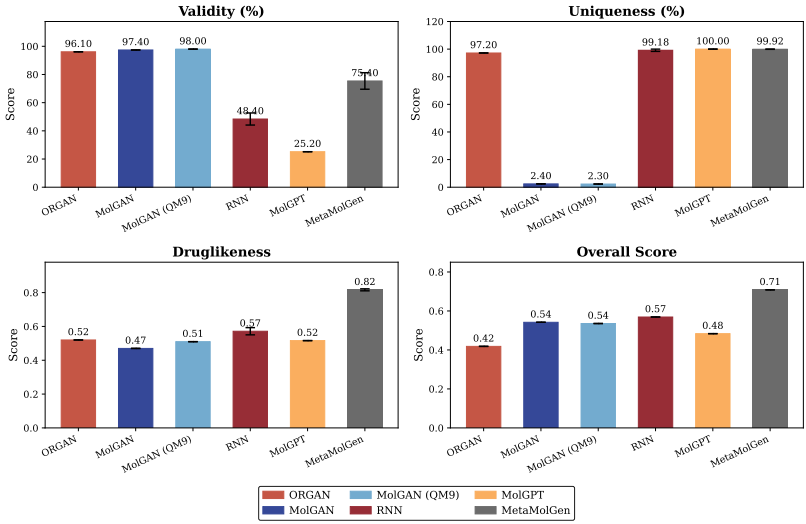


Figure 3. Comparison of key generation metrics across different models, including validity, uniqueness, druglikeness, and overall performance score.

forcement learning and suffer from limited structural exploration, MetaMolGen leverages task-level representations that promote diversity without compromising validity. Additionally, the use of a property projector allows direct control over molecular properties during generation, minimizing reliance on post-hoc filtering. This mechanism contributes to the model's superior drug-likeness and diversity metrics, despite slightly lower validity.

7.2 Performance under few-shot settings

To assess the model's performance in low-resource regimes, we compare MetaMolGen, RNN, and MolGPT under training set sizes ranging from 1,000 to 10,000 samples. MolGAN is excluded due to poor uniqueness in preliminary experiments.

As shown in Figure 4, MetaMolGen consistently outperforms both baselines across all training set sizes, particularly under extremely low-data conditions. It maintains high validity and stability, alongside strong

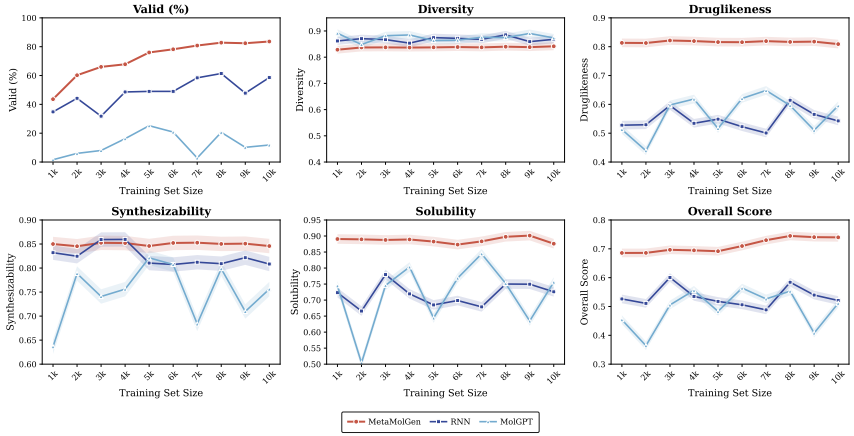


Figure 4. Few-shot performance comparison of MetaMolGen, RNN, and MolGPT across training set sizes (1k–10k) on six key metrics: validity, diversity, druglikeness, synthesizability, solubility, and overall score.

performance in drug-likeness, synthesizability, and solubility, showcasing its robustness across various evaluation metrics.

In contrast, RNN experiences significant fluctuations in performance as the data size changes, while MolGPT, although competitive in uniqueness and diversity, suffers from low validity and weaker overall generation quality.

MetaMolGen's stability under few-shot conditions arises from its meta-learning algorithm, which optimizes for fast adaptation by learning a transferable initialization across tasks. This design enables the model to converge quickly on new tasks even with limited data. The feature standardization module also contributes to stabilizing gradients and preventing dominance of certain features, further enhancing robustness in low-data regimes. Detailed metric breakdowns across all training sizes are provided in Figure 4 and Table 3.

7.3 Conditional property control

We evaluate MetaMolGen's ability to generate molecules with controlled chemical properties using conditional prompts derived from four represen-

Table 3. Comparison of model performance across training sizes and metrics. Gray cells indicate the best value per column (i.e., per training size). All values represent mean ± standard error across 10 independent runs.

Metric	Model	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
Valid (%)	RNN	34.80±4.12	44.20±3.89	31.80±4.31	48.60±3.95	48.40±4.29	49.00±4.02	58.40±3.76	61.40±4.14	47.80±4.48	58.60±3.92
	MolGPT	1.60±0.87	6.00±1.45	8.00±1.89	16.20±2.56	25.20±3.17	20.80±2.94	2.80±1.12	20.60±3.08	10.20±2.33	11.80±2.41
	MetaMolGen	43.60±5.22	60.20±5.47	66.00±5.53	67.80±5.68	75.40±5.87	78.20±5.24	80.80±5.31	82.80±4.95	82.40±5.16	83.60±4.78
Diversity	RNN	0.862±0.008	0.871±0.007	0.867±0.008	0.854±0.006	0.877±0.007	0.872±0.006	0.868±0.007	0.885±0.008	0.859±0.009	0.868±0.007
	MolGPT	0.891±0.012	0.848±0.015	0.882±0.014	0.885±0.013	0.863±0.016	0.865±0.014	0.876±0.015	0.875±0.011	0.891±0.012	0.874±0.010
	MetaMolGen	0.829±0.004	0.837±0.004	0.837±0.005	0.837±0.004	0.837±0.004	0.839±0.005	0.837±0.004	0.840±0.005	0.838±0.004	0.842±0.005
Druglikeness	RNN	0.528±0.023	0.529±0.021	0.596±0.024	0.534±0.022	0.572±0.021	0.523±0.019	0.501±0.020	0.614±0.025	0.565±0.023	0.543±0.021
	MolGPT	0.513±0.038	0.439±0.042	0.598±0.035	0.618±0.031	0.516±0.034	0.621±0.033	0.648±0.042	0.596±0.038	0.510±0.035	0.595±0.032
	MetaMolGen	0.813±0.007	0.813±0.006	0.822±0.005	0.820±0.006	0.816±0.006	0.816±0.005	0.819±0.006	0.817±0.005	0.818±0.006	0.809±0.006
Overall Score	RNN	0.526±0.018	0.511±0.017	0.601±0.021	0.535±0.019	0.569±0.020	0.506±0.017	0.488±0.018	0.584±0.021	0.540±0.020	0.521±0.019
	MolGPT	0.454±0.028	0.364±0.031	0.505±0.027	0.556±0.025	0.484±0.026	0.564±0.027	0.526±0.029	0.553±0.024	0.408±0.028	0.511±0.025
	MetaMolGen	0.686±0.012	0.686±0.011	0.697±0.010	0.695±0.011	0.708±0.010	0.710±0.009	0.730±0.011	0.745±0.010	0.741±0.010	0.740±0.009

tative compounds: Aspirin, Tamiflu, Amoxicillin, and Chloroquine. These compounds span multiple therapeutic domains and provide diverse targets for property conditioned generation.

To support conditional generation, MetaMolGen integrates a lightweight property projector into its latent space. This allows direct control over attributes such as logP, TPSA, SAS, and QED in an end-to-end manner. As shown in Figure 5, the model successfully aligns the distributions of generated molecules with the target values. Strong alignment is observed for QED and SAS, while TPSA remains more challenging to constrain due to its broader natural variability.

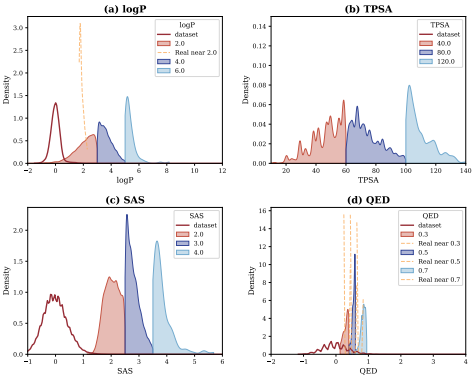


Figure 5. Property distributions under different target constraints: (a) logP, (b) TPSA, (c) SAS, and (d) QED.

To further evaluate conditional generation performance, we compare MetaMolGen with MolGPT under four property targets. As summarized in Table 4, MetaMolGen consistently achieves high validity (≥94.9%), uniqueness (≥99.3%), and novelty across all conditions, while generating more realistic and chemically diverse molecules. Although its MAD and SD values are higher—particularly for TPSA and QED—this reflects an intentional trade-off that enhances structural variety and supports exploratory

drug design. Representative conditional outputs achieving property similarity of 1.0 are shown in Figure 6.

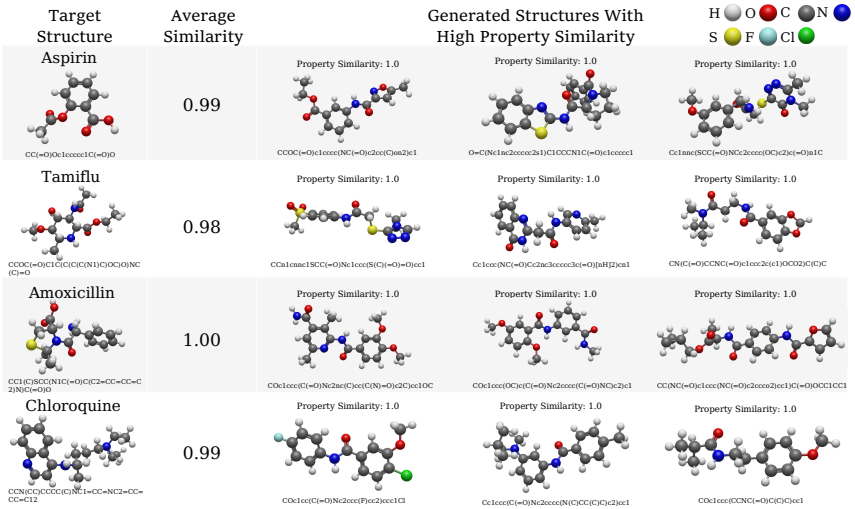


Figure 6. Representative molecules generated by the MetaMolGen model using the condition vectors of Aspirin, Tamiflu, Amoxicillin, Chloroquine. Molecules exhibit high property alignment.

Table 4. Performance comparison under different property constraints (metrics adapted from Table 4 in [1]).

Condition	Model	Val.	Uni.	Nov.	MAD	SD
LogP	MolGPT	0.971	0.998	1.000	0.23	0.31
	MetaMolGen	0.949	0.993	0.997	1.54	0.89
TPSA	MolGPT	0.972	0.996	1.000	3.52	4.66
	MetaMolGen	0.949	0.993	0.997	123.40	31.20
SAS	MolGPT	0.977	0.995	1.000	0.13	0.20
	MetaMolGen	0.949	0.993	0.997	0.57	0.61
QED	MolGPT	0.975	0.998	1.000	0.056	0.075
	MetaMolGen	0.949	0.993	0.997	1.00	0.93

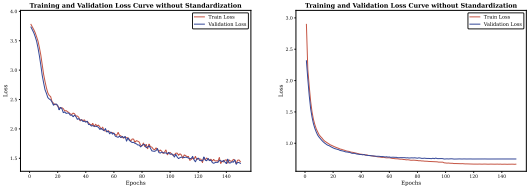
7.4 Impact of feature standardization

Feature standardization plays a crucial role in MetaMolGen's stability and generalization, especially under few-shot conditions. As illustrated in Fig-

ure 8, removing this module results in significant declines in key metrics such as validity, novelty, and conditional generation success rate (CGSR), with the most pronounced performance gaps observed when training data is scarce.

To mitigate issues like gradient instability and feature imbalance, MetaMolGen employs a learnable statistics-based standardization layer to normalize molecular descriptors. This approach enhances training convergence and improves property alignment. Compared to baseline models lacking such mechanisms, MetaMolGen consistently demonstrates superior generation quality and efficiency (Table 5).

The necessity of feature standardization is further underscored by the observed performance drop when it is removed. Without normalization, the model becomes more sensitive to variations in feature scale and less effective in aligning outputs with target properties, particularly in low-data scenarios.



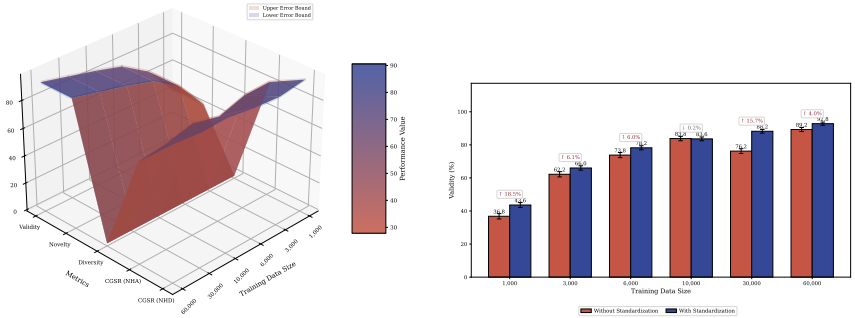
(a) Without standardization **(b)** With standardization

Figure 7. Training loss trajectories with and without feature standardization.

Figure 7 provides a comparison of training loss trajectories with and without feature standardization. The inclusion of feature standardization leads to smoother convergence and reduced variance during meta-adaptation, highlighting its critical role in enhancing model robustness and performance.

Table 5. Ablation study: Evaluation of validity, novelty, diversity, and CGSR with and without standardization. Gray cells indicate better performance. Values represent mean ± standard error from multiple runs.

Training Data Size	Validity (%)		Novelty (%)		Diversity		CGSR (NHA) (%)		CGSR (NHD) (%)	
	Without	With	Without	With	Without	With	Without	With	Without	With
60,000	89.25±1.23	92.80±0.97	89.25±1.18	92.80±1.02	0.8500±0.0016	0.8485±0.0012	74.12±1.25	71.86±0.92	93.42±0.87	95.67±0.64
30,000	76.20±1.46	88.20±1.13	76.20±1.26	88.20±0.93	0.8492±0.0018	0.8451±0.0015	80.07±1.16	74.78±1.31	94.26±0.95	94.03±0.91
10,000	83.80±1.32	83.60±1.15	83.80±1.22	83.60±1.05	0.8462±0.0021	0.8426±0.0018	81.26±1.08	79.71±1.22	95.55±0.81	95.13±0.77
6,000	73.80±1.55	78.20±1.24	73.80±1.35	78.20±1.14	0.8476±0.0012	0.8532±0.0010	72.16±1.15	75.26±1.07	93.24±0.96	92.65±1.04
3,000	62.20±1.58	66.00±1.33	62.20±1.38	66.00±1.23	0.8434±0.0014	0.8503±0.0012	82.32±1.12	81.48±1.18	92.93±1.02	91.58±0.98
1,000	36.80±1.62	43.60±1.48	36.80±1.52	43.60±1.38	0.8332±0.0018	0.8476±0.0015	80.98±1.25	82.13±1.19	91.85±1.05	94.67±0.91



(a) Effect of the standardization layer across evaluation metrics.

(b) Impact of the standardization layer on validity.

Figure 8. Ablation study on the effect of the standardization layer across training sizes.

8 Conclusions and limitations

8.1 Technical discussions

This work proposes MetaMolGen, a meta-learning-based framework designed to address few-shot molecular generation and multi-objective optimization under data scarcity. Leveraging the first-order meta-learning algorithm Reptile, MetaMolGen efficiently adapts to new molecular tasks with limited supervision, improving generalization while maintaining both structural validity and alignment with target properties.

Experimental evaluations on ChEMBL, QM9, ZINC, and MOSES demonstrate the model's strong performance in low-data regimes. With only 1,000–10,000 training samples, MetaMolGen consistently achieves high validity, novelty, and diversity, and exceeds 95% conditional generation success rate (CGSR) under hydrogen bond donor/acceptor (HBD/HBA) constraints, outperforming conventional baselines.

Beyond generative performance, MetaMolGen also enhances molecular screening efficiency by reducing computational overhead through its controllable generation mechanism. This is enabled by a learnable property projector that integrates target attributes (e.g., LogP, TPSA, QED, SAS) into the latent space, allowing the model to generate property-aligned molecules directly, without reliance on post-hoc filtering.

8.2 Existing limitations

While MetaMolGen demonstrates strong adaptability and generalization, several limitations remain. First, task-specific adaptation in latent space is still coarse-grained, potentially limiting fine-grained property tuning in complex design scenarios. Second, the chemical viability of some generated molecules has not yet been fully validated under real-world synthesis constraints. Finally, while conditional control via latent projection is effective, it may require retraining when generalizing to entirely new property domains.

8.3 Future extensions

Future work will focus on several key directions. First, we aim to explore advanced meta-learning algorithms for fine-grained control over molecular attributes, including gradient-based and metric-based adaptation hybrids. Second, we plan to integrate active learning strategies to further reduce data requirements and improve sample efficiency. Finally, we envision extending the MetaMolGen framework to broader tasks such as reaction pathway prediction, multi-step synthesis planning, and macromolecular design involving proteins and peptides.

***Acknowledgment*:** This work was supported by the National Natural Science Foundation of China [61773020] and the Graduate Innovation Project of National University of Defense Technology [XJQY2024065]. The authors would like to express their sincere gratitude to all the referees for their careful reading and insightful suggestions.

Data and Software Availability

The source code for the MetaMolGen model developed in this study is publicly available on at `https://github.com/yzz980314/MetaMolGen`. The datasets used for evaluation are all publicly available. The ChEMBL database can be accessed from `https://www.ebi.ac.uk/chembl/`. The QM9 dataset is available at `http://quantum-machine.org/datasets/`. The ZINC database is available from `https://zinc.docking.org/`. The

MOSES benchmark framework and its associated data splits can be found at `https://github.com/molecularai/moses`.

References

[1] V. Bagal, R. Aggarwal, P. K. Vinod, U. D. Priyakumar, MolGPT: molecular generation using a transformer-decoder model, *J. Chem. Inf. Model.* **62** (2022) 2064–2076.

[2] A. Beck, *First-Order Methods in Optimization*, Society for Industrial and Applied Mathematics, 2017.

[3] E. J. Bjerrum, SMILES enumeration as data augmentation for neural network modeling of molecules, *arXiv* (2017) **doi:** `https://doi.org/10.48550/arXiv.1703.07076`.

[4] O. Bousquet, A. Elisseeff, Stability and generalization, *J. Mach. Learn. Res.* **2** (2002) 499–526.

[5] S. Choi, S. Seo, B. J. Kim, C. Park, S. Park, PIDiff: physics informed diffusion model for protein pocket-specific 3D molecular generation, *Comput. Biol. Med.* **180** (2024) #108865.

[6] N. De Cao, T. Kipf, MolGAN: an implicit generative model for small molecular graphs, *arXiv* (2018) **doi:** `https://doi.org/10.48550/arXiv.1805.11973`.

[7] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning, *Proc. Mach. Learn. Res.* **70** (2017) 1126–1135.

[8] A. Y. K. Foong, W. P. Bruinsma, J. Gordon, Y. Dubois, J. Requeima, R. E. Turner, Meta-learning stationary stochastic process prediction with convolutional neural processes, *Adv. Neural Inf. Process. Syst.* **33** (2020) 8284–8295.

[9] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, J. P. Overington, ChEMBL: a large-scale bioactivity database for drug discovery, *Nucleic Acids Res.* **40** (2012) D1100–D1107.

[10] J. Gordon, W. P. Bruinsma, A. Y. K. Foong, J. Requeima, Y. Dubois, R. E. Turner, Convolutional conditional neural processes, *arXiv* (2019) **doi:** `https://doi.org/10.48550/arXiv.1910.13556`.

[11] G. L. Guimaraes, B. Sanchez-Lengeling, P. L. C. Farias, A. Aspuru-Guzik, Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models, *arXiv* (2017) **doi:** `https://doi.org/10.48550/arXiv.1705.10843`.

[12] T. M. Hospedales, A. Antoniou, P. Micaelli, A. J. Storkey, Meta-learning in neural networks: a survey, *IEEE Trans. Pattern Anal. Mach. Intell.* **44** (2022) 5149–5169.

[13] J. J. Irwin, B. K. Shoichet, ZINC: a free database of commercially available compounds for virtual screening, *J. Chem. Inf. Model.* **45** (2005) 177–182.

[14] W. Jin, R. Barzilay, T. Jaakkola, Junction tree variational autoencoder for molecular graph generation, in: N. Brown (Ed.), *Artificial Intelligence in Drug Discovery*, Royal Soc. Chem., London, 2018, pp. 2323–2332.

[15] W. Jin, R. Barzilay, T. Jaakkola, Hierarchical generation of molecular graphs using structural motifs, in: Hal Daumé, Aarti Singh (Eds.), *ICML'20: Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2020, pp. 4839–4848.

[16] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, S. M. Ali Eslami, D. Rosenbaum, O. Vinyals, Y. W. Teh, Attentive neural processes, in: International Conference on Learning Representations, 2019.

[17] M. J. Kusner, D. He, H. Larochelle, Grammar variational autoencoder, in: Proceedings of the 34th International Conference on Machine Learning, *Proc. Mach. Learn. Res.* **70** (2017) 1945–1954.

[18] M. Lee, K. Min, MGCVAE: multi-objective inverse design via molecular graph conditional variational autoencoder, *J. Chem. Inf. Model.* **62** (2022) 2943–2950.

[19] J. Lim, S. Ryu, J. W. Kim, W. Y. Kim, Molecular generative model based on conditional variational autoencoder for de novo molecular design, *J. Cheminform.* **10** (2018) #31.

[20] D. A. McAllester, Some PAC-Bayesian theorems, in: Proceedings of the 11th Annual Conference on Computational Learning Theory, ACM Press, 1998, pp. 230–234.

[21] A. Nichol, J. Achiam, J. Schulman, On first-order meta-learning algorithms, *arXiv* (2018) **doi:** `https://doi.org/10.48550/arXiv.1803.02999`.

[22] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Acad. Pub., Boston, 2004.

[23] H. Öztürk, A. Özgür, P. Schwaller, T. Laino, E. Ozkirimli, Exploring chemical space using natural language processing methodologies for drug discovery, *Drug Discov. Today* **25** (2020) 689–705.

[24] P. G. Polishchuk, T. I. Madzhidov, A. Varnek, Estimation of the size of drug-like chemical space based on GDB-17 data, *J. Comput. Aided Mol. Des.* **27** (2013) 675–679.

[25] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Johansson, H. Chen, S. Nikolenko, A. Aspuru-Guzik, A. Zhavoronkov, Molecular sets (MOSES): a benchmarking platform for molecular generation models, *Front. Pharmacol.* **11** (2020) #565644.

[26] S. T. Radev, M. Schmitt, L. Schumacher, L. Elsemüller, V. Pratz, Y. Schälte, U. Köthe, P. C. Bürkner, BayesFlow: Amortized bayesian workflows with neural networks, *J. Open Source Softw.* **8** (2023) #5702.

[27] R. Ramakrishnan, P. O. Dral, M. Rupp, O. A. von Lilienfeld, Quantum chemistry structures and properties of 134 kilo molecules, *Sci. Data* **1** (2014) #140022.

[28] J. L. Reymond, R. van Deursen, L. C. Blum, L. Ruddigkeit, Chemical space as a source for new drugs, *MedChemComm* **1** (2010) 30–38.

[29] T. Scior, A. Bender, G. Tresadern, J. L. Medina-Franco, K. Martínez-Mayorga, T. Langer, K. Cuanalo-Contreras, D. K. Agrafiotis, Recognizing pitfalls in virtual screening: a critical review, *J. Chem. Inf. Model.* **52** (2012) 867–881.

[30] H. H. Loeffler, J. He, A. Tibo, J. P. Janet, A. Voronov, L. H. Mervin, O. Engkvist, REINVENT 4: modern AI-driven generative molecule design, *J. Cheminform.* **16** (2024) #20.

[31] M. H. S. Segler, T. Kogej, C. Tyrchan, M. P. Waller, Generating focused molecule libraries for drug discovery with recurrent neural networks, *ACS Cent. Sci.* **4** (2018) 120–131.

[32] B. K. Shoichet, Virtual screening of chemical libraries, *Nature* **432** (2004) 862–865.

[33] D. Weininger, SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules, *J. Chem. Inf. Comput. Sci.* **28** (1988) 31–36.

[34] T. Xie, J. C. Grossman, Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties, *Phys. Rev. Lett.* **120** (2018) #145301.

[35] J. You, B. Liu, R. Ying, V. Pande, J. Leskovec, Graph convolutional policy network for goal-directed molecular graph generation, *Adv. Neural Inf. Process. Sys.* **31** (2018) 6410–6421.

[36] J. Zhang, M. Li, Y. Xu, H. He, Q. Li, T. Wang, StrucGCN: structural enhanced graph convolutional networks for graph embedding, *Inf. Fusion* **117** (2025) #102893.

[37] A. Zhavoronkov, Y. A. Ivanenkov, A. Aliper, M. S. Veselov, V. A. Aladinskiy, A. V. Aladinskaya, V. A. Terentiev, D. A. Polykovskiy, M. D. Kuznetsov, A. Asadulaev, Y. Volkov, A. Zholus, R. R. Shayakhmetov, Z. Johnson, K. S. Chen, L. Tsetlin, A. Shevtsov, A. Shneyderman, K. Finogenov, T. Oprea, I. V. Ozerov, E. Putin, Deep learning enables rapid identification of potent DDR1 kinase inhibitors, *Nat. Biotech.* **37** (2019) 1038–1040.