

Mathematical Modeling of Genotype-Guided De Novo Molecule Generation

Multi-Omics Drug Discovery Project

December 13, 2025

Contents

1	Introduction and Problem Formulation	4
1.1	Problem Statement	4
1.2	Hypothesis	4
1.3	Objectives	4
1.4	Mathematical Notation	4
2	Architecture Overview	5
2.1	Dual VAE System	5
2.2	Data Flow	5
2.3	Integration Mechanism	5
3	Profile VAE (PVAE) - Transcriptomic Encoder	6
3.1	Input Representation	6
3.2	Encoder Architecture	6
3.3	Transformer Encoder for Profiles	6
3.4	NewGELU Activation	6
3.5	Loss Function	7
4	SMILES VAE (SVAE) - Molecular Encoder	7
4.1	Input Representation	7
4.2	Encoder Architecture	7
4.3	Positional Encoding	7
4.4	Multi-Head Attention	7
4.5	Pooling Strategy	8
4.6	Reparameterization Trick	8
4.7	Loss Function	8
5	Latent Space Integration	8
5.1	Gaussian Addition	8
5.2	Alternative Integration Methods	9
5.2.1	Concatenation	9
5.2.2	Cross-Attention Mechanism	9
5.2.3	Product of Experts	9
5.2.4	Weighted Combination	9
5.3	Proof of Concept: Information Preservation in Gaussian Addition	9

6	Decoder Architecture	9
6.1	Decoder Definition	9
6.2	Autoregressive Generation	10
6.3	State Representation	10
6.4	Transformer Decoder	10
6.5	Generation Process	10
7	Reinforcement Learning Framework	11
7.1	Reward Function	11
7.2	IC50 Prediction Model	11
7.3	Objective Function	11
7.4	Policy Gradient	11
7.5	REINFORCE Algorithm	12
7.6	Baseline Subtraction	12
7.7	Alternative RL Approaches	12
7.7.1	Proximal Policy Optimization (PPO)	12
7.7.2	Actor-Critic	12
8	Training Objectives	12
8.1	Pretraining Phase	12
8.1.1	Profile VAE Pretraining	12
8.1.2	SMILES VAE Pretraining	13
8.2	Fine-tuning Phase	13
8.3	KL Divergence Formulas	13
8.4	Reconstruction Loss	13
9	Alternative Architectures and Approaches	13
9.1	VAE Variants	13
9.1.1	β -VAE	13
9.1.2	Wasserstein Autoencoder (WAE)	14
9.1.3	Vector Quantized VAE (VQ-VAE)	14
9.2	Alternative Attention Mechanisms	14
9.2.1	Sparse Attention	14
9.2.2	Linear Attention	14
9.2.3	Performer	14
9.3	Alternative Pooling Strategies	14
10	Proofs of Concept	14
10.1	POC 1: VAE Enables Smooth Latent Interpolation	14
10.2	POC 2: Information Preservation in Gaussian Addition	15
10.3	POC 3: Conditioning on Transcriptomics Improves Specificity	15
10.4	POC 4: Convergence of RL Objective	15
10.5	POC 5: Transformer Architecture for Sequential Generation	16
11	Implementation Details	16
11.1	Hyperparameters	16
11.2	Training Procedure	16
11.3	Data Preprocessing	17
11.3.1	TCGA Preprocessing	17
11.3.2	SMILES Preprocessing	17
11.3.3	GDSC Preprocessing	17

1 Introduction and Problem Formulation

1.1 Problem Statement

Traditional target-based drug discovery approaches focus on protein-specific information to generate new molecules. However, this methodology has significant limitations:

- **Ignoring biological context:** Target-based models ignore gene expression context and pathway rewiring
- **High rejection rate:** Generated molecules often fail in clinical trials due to lack of disease specificity
- **Toxicity and side effects:** Off-target and on-target side effects are not adequately predicted
- **Poor efficacy prediction:** Models fail to predict true efficacy and toxicity in biological systems

The biological context of cancer lies in the broader picture of cell biology, incorporating:

- Transcriptomic signatures
- Metabolic pathways
- Genetic mutations
- System-level regulatory networks

1.2 Hypothesis

If we condition molecule generation on transcriptomic profiles, the model can learn to design molecules specifically effective for that cancer profile. This approach leverages the central dogma of molecular biology:

DNA (mutations) \rightarrow RNA (transcriptome shift) \rightarrow Protein (abnormal functions) \rightarrow System level (pathway mi

1.3 Objectives

1. Build a generative model to generate new anticancer molecules based only on gene expression profiles for a particular cancer type
2. Use reinforcement learning to ensure generated molecules have high efficacy (low IC50)
3. Integrate multi-omics data (genomics, transcriptomics) into the generation pipeline

1.4 Mathematical Notation

Throughout this document, we use the following notation:

- $x_p \in \mathbb{R}^G$: Gene expression profile vector with G genes
- x_c : SMILES sequence (tokenized molecular representation)
- $z_p \in \mathbb{R}^{d_p}$: Latent representation of profile (Profile VAE)
- $z_c \in \mathbb{R}^{d_c}$: Latent representation of molecule (SMILES VAE)

- $z \in \mathbb{R}^d$: Combined latent representation
- C_T : Complete generated molecule at time T
- S_t : State at time step t during generation
- a_t : Action (token) at time step t
- μ_p, σ_p^2 : Mean and variance of profile latent distribution
- μ_c, σ_c^2 : Mean and variance of molecule latent distribution
- θ : Decoder parameters
- ϕ_p : Profile encoder parameters
- ϕ_c : SMILES encoder parameters
- $f(C_T, x_c)$: IC50 prediction function
- $R(S_T)$: Reward function
- α : Temperature parameter for reward function
- β_p, β_c : KL divergence weights for PVAE and SVAE
- λ : Reinforcement learning loss weight

2 Architecture Overview

2.1 Dual VAE System

The architecture consists of two Variational Autoencoders (VAEs) [1] that are pretrained separately and then jointly fine-tuned:

1. **Profile VAE (PVAE)**: Encodes transcriptomic profiles into latent space
2. **SMILES VAE (SVAE)**: Encodes molecular structures (SMILES) into latent space

2.2 Data Flow

- **Pretraining Phase:**
 - PVAE: Trained on TCGA (The Cancer Genome Atlas) [2] transcriptomic data
 - SVAE: Trained on ChEMBL [3] molecular database
- **Fine-tuning Phase:**
 - Joint training on GDSC (Genomics of Drug Sensitivity in Cancer) [4] dataset
 - GDSC provides: (cell line, drug, IC50) triplets with transcriptomic profiles

2.3 Integration Mechanism

The genetic embedding z_p and SMILES embedding z_c are combined using Gaussian addition:

$$z = z_p + z_c$$

where $z_p \sim \mathcal{N}(\mu_p, \sigma_p^2)$ and $z_c \sim \mathcal{N}(\mu_c, \sigma_c^2)$.

The combined latent z is then decoded using the SMILES decoder to generate new molecules conditioned on the transcriptomic profile.

3 Profile VAE (PVAE) - Transcriptomic Encoder

3.1 Input Representation

The Profile VAE takes gene expression profiles as input:

$$x_p \in \mathbb{R}^G$$

where G is the number of genes (typically $\sim 20,000$ genes from TCGA). The expression values are log2-normalized abundances.

3.2 Encoder Architecture

The encoder maps the gene expression profile to a latent distribution:

$$q_{\phi_p}(z_p|x_p) = \mathcal{N}(z_p; \mu_p(x_p), \sigma_p^2(x_p))$$

where:

$$\mu_p(x_p) = \text{Encoder}_{\mu}(x_p; \phi_p) \tag{1}$$

$$\sigma_p^2(x_p) = \text{Encoder}_{\sigma}(x_p; \phi_p) \tag{2}$$

3.3 Transformer Encoder for Profiles

Following the Transformer VAE architecture [5], the profile encoder uses:

1. **Input Projection:** $x_p \rightarrow E_p \in \mathbb{R}^{L \times d_{model}}$ where L is sequence length (genes can be treated as sequence or flattened)
2. **Positional Encoding:** Sinusoidal positional encodings
3. **Multi-Head Self-Attention** [6]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

4. **Feedforward Network:**

$$\text{FFN}(x) = \text{NewGELU}(W_2 \cdot \text{NewGELU}(W_1x + b_1) + b_2)$$

5. **Pooling:** Mean, start, and max pooling concatenated
6. **Latent Projection:** Pooled features $\rightarrow (\mu_p, \sigma_p^2)$

3.4 NewGELU Activation

The NewGELU activation function is defined as:

$$\text{NewGELU}(x) = 0.5x \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right) \right)$$

3.5 Loss Function

The PVAE loss consists of reconstruction and KL divergence terms:

$$\mathcal{L}_{PVAE} = \mathcal{L}_{recon}^p + \beta_p \mathcal{L}_{KL}^p$$

where:

$$\mathcal{L}_{recon}^p = -\mathbb{E}_{z_p \sim q_{\phi_p}} [\log p_{\theta_p}(x_p|z_p)] \quad (3)$$

$$\mathcal{L}_{KL}^p = \text{KL}(q_{\phi_p}(z_p|x_p) \| p(z_p)) \quad (4)$$

Assuming a standard normal prior $p(z_p) = \mathcal{N}(0, I)$:

$$\mathcal{L}_{KL}^p = \frac{1}{2} \sum_{i=1}^{d_p} [\mu_{p,i}^2 + \sigma_{p,i}^2 - \log(\sigma_{p,i}^2) - 1]$$

4 SMILES VAE (SVAE) - Molecular Encoder

4.1 Input Representation

SMILES sequences are tokenized into discrete tokens:

$$x_c = [t_1, t_2, \dots, t_L]$$

where $t_i \in \{1, 2, \dots, V\}$ and V is the vocabulary size (typically 45 tokens including special tokens: padding, start, end).

4.2 Encoder Architecture

The SMILES encoder follows the Transformer VAE architecture:

$$q_{\phi_c}(z_c|x_c) = \mathcal{N}(z_c; \mu_c(x_c), \sigma_c^2(x_c))$$

4.3 Positional Encoding

Sinusoidal positional encodings are added to token embeddings:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

where pos is the position and i is the dimension index.

4.4 Multi-Head Attention

For h attention heads [6], each head computes:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{model} \times d_k}$ and $W^O \in \mathbb{R}^{hd_k \times d_{model}}$.

4.5 Pooling Strategy

The encoder uses NoAffinePooler that concatenates three pooling strategies:

$$\text{pooled} = [\text{mean_pooled}; \text{start_token}; \text{max_pooled}] \quad (5)$$

$$\text{mean_pooled} = \frac{1}{L} \sum_{i=1}^L h_i \quad (6)$$

$$\text{start_token} = h_1 \quad (7)$$

$$\text{max_pooled} = \max_{i=1}^L h_i \quad (8)$$

where h_i are the hidden states from the Transformer encoder.

4.6 Reparameterization Trick

To enable backpropagation through the stochastic latent variable [1]:

$$z_c = \mu_c + \sigma_c \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

This allows gradients to flow through the sampling operation.

4.7 Loss Function

The SVAE loss:

$$\mathcal{L}_{SVAE} = \mathcal{L}_{recon}^c + \beta_c \mathcal{L}_{KL}^c$$

where:

$$\mathcal{L}_{recon}^c = - \sum_{t=1}^L \log p_{\theta_c}(t_t | z_c, t_{<t}) \quad (9)$$

$$\mathcal{L}_{KL}^c = \frac{1}{2} \sum_{i=1}^{d_c} [\mu_{c,i}^2 + \sigma_{c,i}^2 - \log(\sigma_{c,i}^2) - 1] \quad (10)$$

The reconstruction loss is the cross-entropy over the vocabulary for each token position.

5 Latent Space Integration

5.1 Gaussian Addition

The primary integration method combines latent representations through Gaussian addition:

$$z = z_p + z_c$$

Given $z_p \sim \mathcal{N}(\mu_p, \Sigma_p)$ and $z_c \sim \mathcal{N}(\mu_c, \Sigma_c)$, the sum follows:

$$z \sim \mathcal{N}(\mu_p + \mu_c, \Sigma_p + \Sigma_c)$$

For diagonal covariance matrices:

$$z \sim \mathcal{N}(\mu_p + \mu_c, \text{diag}(\sigma_p^2 + \sigma_c^2))$$

5.2 Alternative Integration Methods

5.2.1 Concatenation

$$z = [z_p; z_c] \in \mathbb{R}^{d_p+d_c}$$

This doubles the latent dimension but preserves all information from both modalities.

5.2.2 Cross-Attention Mechanism

Use cross-attention to allow the profile to attend to molecular features:

$$z = \text{CrossAttention}(z_p, z_c) = \text{softmax}\left(\frac{z_p z_c^T}{\sqrt{d}}\right) z_c$$

5.2.3 Product of Experts

$$p(z) \propto p_p(z) \cdot p_c(z)$$

where $p_p(z) = \mathcal{N}(\mu_p, \sigma_p^2)$ and $p_c(z) = \mathcal{N}(\mu_c, \sigma_c^2)$.

For Gaussian experts, the product is also Gaussian:

$$p(z) = \mathcal{N}\left(\frac{\mu_p/\sigma_p^2 + \mu_c/\sigma_c^2}{1/\sigma_p^2 + 1/\sigma_c^2}, \frac{1}{1/\sigma_p^2 + 1/\sigma_c^2}\right)$$

5.2.4 Weighted Combination

$$z = \alpha z_p + (1 - \alpha) z_c, \quad \alpha \in [0, 1]$$

This allows controlling the relative importance of profile vs. molecule information.

5.3 Proof of Concept: Information Preservation in Gaussian Addition

Proof of Concept 5.1 (Information Preservation). Gaussian addition preserves information from both modalities when the latent dimensions are aligned. The combined distribution has:

- Mean: $\mu_p + \mu_c$ (additive combination of means)
- Variance: $\sigma_p^2 + \sigma_c^2$ (additive combination of variances)

This preserves the first two moments of both distributions, allowing the decoder to access information from both the transcriptomic profile and molecular structure.

6 Decoder Architecture

6.1 Decoder Definition

The decoder reconstructs molecules from the combined latent representation:

$$q_\theta(x'|z_p + z_c) = q_\theta(x'|z)$$

where x' is the generated SMILES sequence.

6.2 Autoregressive Generation

Molecule generation proceeds autoregressively:

$$p(S_T) = \prod_{t=1}^T p(a_t | S_{t-1})$$

where:

- S_t is the state at time t
- a_t is the action (token) at time t
- T is the total sequence length

6.3 State Representation

The state at time t is defined as:

$$S_t = \text{tuple}(C_t, x_c)$$

where:

- C_t is the partially generated molecule up to time t
- x_c is the conditioning transcriptomic profile (encoded as z_p)

6.4 Transformer Decoder

The decoder uses masked self-attention to ensure causal generation:

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T + M}{\sqrt{d_k}}\right) V$$

where M is a causal mask:

$$M_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{if } i < j \end{cases}$$

This ensures that token i can only attend to tokens $j \leq i$.

6.5 Generation Process

At each step t :

1. Embed current token: $e_t = \text{Embedding}(a_t)$
2. Add positional encoding: $e'_t = e_t + PE_t$
3. Add latent conditioning: $h_t^0 = e'_t + \text{Project}(z)$
4. Pass through decoder layers: $h_t^L = \text{Decoder}(h_t^0, h_{<t})$
5. Project to vocabulary: $p(a_{t+1}) = \text{softmax}(W_o h_t^L)$
6. Sample next token: $a_{t+1} \sim p(a_{t+1})$

7 Reinforcement Learning Framework

7.1 Reward Function

Once a molecule C_T is completely generated, a reward is assigned based on predicted IC50:

$$R(S_T) = \exp\left(-\frac{f(C_T, x_c)}{\alpha}\right)$$

where:

- $f(C_T, x_c)$ is the IC50 prediction model trained on GDSC dataset
- α is a temperature parameter controlling reward sharpness
- Lower IC50 (higher efficacy) \rightarrow higher reward

7.2 IC50 Prediction Model

The IC50 prediction function f is a regression model trained on GDSC data [4, 7]:

$$f(C_T, x_c) = \text{Regressor}(\text{Encoder}(C_T), x_c)$$

This can be implemented as:

- Neural network: $f(C_T, x_c) = W_2 \text{ReLU}(W_1[z_c; x_c] + b_1) + b_2$
- Gradient boosting (XGBoost, LightGBM)
- Random forest regression

7.3 Objective Function

The objective is to maximize the expected reward:

$$J(\theta) = \mathbb{E}[R(S_T)] = \sum_{T \in \mathcal{M}} p(S_T) R(S_T)$$

where \mathcal{M} is the set of all possible molecules.

7.4 Policy Gradient

To optimize the objective, we compute the gradient:

$$\nabla_{\theta} J(\theta) = \mathbb{E}[R(S_T) \nabla_{\theta} \log p(S_T)]$$

Using the log-derivative trick:

$$\nabla_{\theta} \log p(S_T) = \sum_{t=1}^T \nabla_{\theta} \log p(a_t | S_{t-1})$$

Therefore:

$$\nabla_{\theta} J(\theta) = \mathbb{E}\left[R(S_T) \sum_{t=1}^T \nabla_{\theta} \log p(a_t | S_{t-1})\right]$$

7.5 REINFORCE Algorithm

The REINFORCE algorithm [8] estimates the gradient:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{n=1}^N R(S_T^{(n)}) \sum_{t=1}^T \nabla_{\theta} \log p(a_t^{(n)} | S_{t-1}^{(n)})$$

where N is the number of sampled trajectories.

7.6 Baseline Subtraction

To reduce variance, subtract a baseline b :

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{n=1}^N (R(S_T^{(n)}) - b) \sum_{t=1}^T \nabla_{\theta} \log p(a_t^{(n)} | S_{t-1}^{(n)})$$

Common baselines:

- Moving average: $b = \frac{1}{N} \sum_{n=1}^N R(S_T^{(n)})$
- Value function: $b = V(S_{t-1})$ (learned critic)

7.7 Alternative RL Approaches

7.7.1 Proximal Policy Optimization (PPO)

PPO [9] clips the policy update to prevent large changes:

$$L^{CLIP}(\theta) = \mathbb{E} \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{p_{\theta}(a_t | s_t)}{p_{\theta_{old}}(a_t | s_t)}$ is the importance ratio.

7.7.2 Actor-Critic

Uses a value function $V(s)$ to estimate the advantage:

$$\hat{A}_t = R(S_T) - V(S_{t-1})$$

8 Training Objectives

8.1 Pretraining Phase

8.1.1 Profile VAE Pretraining

Train PVAE on TCGA data:

$$\mathcal{L}_{PVAE}^{pretrain} = \mathcal{L}_{recon}^p + \beta_p \mathcal{L}_{KL}^p$$

where:

$$\mathcal{L}_{recon}^p = -\mathbb{E}_{z_p \sim q_{\phi_p}} [\log p_{\theta_p}(x_p | z_p)] \quad (11)$$

$$\mathcal{L}_{KL}^p = \text{KL}(q_{\phi_p}(z_p | x_p) \| \mathcal{N}(0, I)) \quad (12)$$

8.1.2 SMILES VAE Pretraining

Train SVAE on ChEMBL data:

$$\mathcal{L}_{SVAE}^{pretrain} = \mathcal{L}_{recon}^c + \beta_c \mathcal{L}_{KL}^c$$

where:

$$\mathcal{L}_{recon}^c = - \sum_{t=1}^L \log p_{\theta_c}(t_t | z_c, t_{<t}) \quad (13)$$

$$\mathcal{L}_{KL}^c = \text{KL}(q_{\phi_c}(z_c | x_c) \| \mathcal{N}(0, I)) \quad (14)$$

8.2 Fine-tuning Phase

Joint training on GDSC dataset with combined loss:

$$\mathcal{L}_{joint} = \mathcal{L}_{PVAE} + \mathcal{L}_{SVAE} + \lambda \mathcal{L}_{RL}$$

where:

$$\mathcal{L}_{RL} = -\mathbb{E}[R(S_T) \log p(S_T)] \quad (15)$$

8.3 KL Divergence Formulas

For Gaussian distributions $q(z) = \mathcal{N}(\mu, \sigma^2)$ and $p(z) = \mathcal{N}(0, 1)$:

$$\text{KL}(q \| p) = \frac{1}{2} \sum_{i=1}^d [\mu_i^2 + \sigma_i^2 - \log(\sigma_i^2) - 1]$$

For multivariate case with diagonal covariance:

$$\text{KL}(q \| p) = \frac{1}{2} [\text{tr}(\Sigma) + \mu^T \mu - d - \log |\Sigma|]$$

8.4 Reconstruction Loss

For discrete sequences (SMILES), reconstruction loss is cross-entropy:

$$\mathcal{L}_{recon}^c = - \sum_{t=1}^L \sum_{v=1}^V y_{t,v} \log p_{\theta}(v | z, t_{<t})$$

where $y_{t,v}$ is the one-hot encoding of the true token at position t .

9 Alternative Architectures and Approaches

9.1 VAE Variants

9.1.1 β -VAE

Introduces a hyperparameter β to control the trade-off between reconstruction and regularization [10]:

$$\mathcal{L}_{\beta\text{-VAE}} = \mathcal{L}_{recon} + \beta \mathcal{L}_{KL}$$

Higher β encourages more disentangled representations.

9.1.2 Wasserstein Autoencoder (WAE)

Replaces KL divergence with Wasserstein distance [11]:

$$\mathcal{L}_{WAE} = \mathcal{L}_{recon} + \lambda \mathcal{W}(q(z), p(z))$$

9.1.3 Vector Quantized VAE (VQ-VAE)

Uses discrete latent codes [12]:

$$z_q = \text{Quantize}(z_e) = \arg \min_{z_k \in \mathcal{Z}} \|z_e - z_k\|$$

9.2 Alternative Attention Mechanisms

9.2.1 Sparse Attention

Limits attention to a subset of positions to reduce computational cost.

9.2.2 Linear Attention

Replaces softmax with linear operations:

$$\text{LinearAttention}(Q, K, V) = \frac{QK^TV}{\|QK^T\|_1}$$

9.2.3 Performer

Uses random features to approximate attention with linear complexity.

9.3 Alternative Pooling Strategies

- **Attention Pooling:** Weighted sum using attention scores
- **Last Token:** Use only the last hidden state
- **Learnable Pooling:** Trainable weighted combination

10 Proofs of Concept

10.1 POC 1: VAE Enables Smooth Latent Interpolation

Proof of Concept 10.1 (Latent Interpolation). The VAE latent space [1] is continuous and regularized by the KL divergence term. For two molecules with latents z_1 and z_2 , interpolation:

$$z(\alpha) = (1 - \alpha)z_1 + \alpha z_2, \quad \alpha \in [0, 1]$$

produces valid molecules because:

- The prior $p(z) = \mathcal{N}(0, I)$ encourages the latent space to be smooth
- The KL term penalizes deviations from the prior, ensuring continuity
- Interpolated points remain in regions of high probability under the prior

10.2 POC 2: Information Preservation in Gaussian Addition

Proof of Concept 10.2 (Gaussian Addition Information). Given two independent Gaussian distributions $z_p \sim \mathcal{N}(\mu_p, \Sigma_p)$ and $z_c \sim \mathcal{N}(\mu_c, \Sigma_c)$, their sum $z = z_p + z_c$ has:

$$z \sim \mathcal{N}(\mu_p + \mu_c, \Sigma_p + \Sigma_c)$$

The mutual information is preserved because:

- The mean captures the first moment (expected value) of both distributions
- The covariance captures the second moment (uncertainty) of both distributions
- The decoder can recover information from both modalities through the combined mean and variance

10.3 POC 3: Conditioning on Transcriptomics Improves Specificity

Proof of Concept 10.3 (Transcriptomic Conditioning). Conditioning molecule generation on transcriptomic profiles x_c improves cancer-specificity because:

- **Pathway alignment:** Transcriptomic profiles encode pathway activity, guiding generation toward molecules that target active pathways
- **Disease context:** Different cancer types have distinct transcriptomic signatures, enabling type-specific generation
- **Reduced off-target effects:** Molecules generated for specific profiles are more likely to interact with disease-relevant targets

Mathematically, conditioning increases the likelihood of generating effective molecules:

$$p(\text{effective}|x_c) > p(\text{effective})$$

where effectiveness is measured by IC50.

10.4 POC 4: Convergence of RL Objective

Proof of Concept 10.4 (RL Convergence). The REINFORCE algorithm converges under the following conditions:

1. **Bounded rewards:** $|R(S_T)| \leq R_{\max}$ for all S_T
2. **Policy gradient exists:** $\nabla_{\theta} \log p(S_T)$ is well-defined
3. **Learning rate schedule:** $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$
4. **Baseline reduces variance:** Using a baseline b reduces the variance of gradient estimates

Under these conditions, the policy gradient method converges to a local optimum of $J(\theta)$.

10.5 POC 5: Transformer Architecture for Sequential Generation

Proof of Concept 10.5 (Transformer for Molecules). The Transformer architecture [6] is well-suited for sequential molecule generation because:

- **Long-range dependencies:** Self-attention captures dependencies between distant atoms in the molecule
- **Parallel training:** Unlike RNNs, Transformers can process sequences in parallel during training
- **Positional encoding:** Captures the sequential nature of SMILES strings
- **Autoregressive generation:** Causal masking enables autoregressive generation while maintaining parallel training benefits
- **Context awareness:** Multi-head attention allows the model to attend to different aspects of the molecular structure simultaneously

11 Implementation Details

11.1 Hyperparameters

Based on the current implementation:

- **Model dimensions:**
 - $d_{model} = 512$ (embedding dimension)
 - $d_{latent} = 512$ (can be reduced to ~ 32)
 - $n_{layers} = 8$ (encoder/decoder layers)
 - $n_{heads} = 8$ (attention heads)
 - $d_{ff} = 2048$ (feedforward dimension, $4 \times d_{model}$)
- **Training:**
 - Learning rate: 10^{-4}
 - Optimizer: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.98$
 - Weight decay: 10^{-4}
 - Warmup steps: 4000
 - Gradient clipping: 1.0
- **Loss weights:**
 - $\beta_p = 0.001$ (PVAE KL weight)
 - $\beta_c = 0.001$ (SVAE KL weight)
 - $\lambda = 1.0$ (RL loss weight, tunable)
- **Sequence:**
 - Max length: 122 tokens
 - Vocabulary size: 45 tokens

11.2 Training Procedure

Algorithm 1 Joint Training Procedure

```
1: Input: TCGA profiles, ChEMBL molecules, GDSC triplets
2: Initialize: PVAE, SVAE with random weights
3:
4: Phase 1: Pretraining
5: for epoch in pretrain_epochs do
6:   Train PVAE on TCGA:  $\mathcal{L}_{PVAE}$ 
7:   Train SVAE on ChEMBL:  $\mathcal{L}_{SVAE}$ 
8: end for
9:
10: Phase 2: Fine-tuning
11: for epoch in finetune_epochs do
12:   for batch in GDSC do
13:     Sample  $(x_p, x_c, IC50)$  from batch
14:     Encode:  $z_p = \text{PVAE}(x_p)$ ,  $z_c = \text{SVAE}(x_c)$ 
15:     Combine:  $z = z_p + z_c$ 
16:     Decode:  $C_T = \text{Decoder}(z)$ 
17:     Compute reward:  $R = \exp(-f(C_T, x_c)/\alpha)$ 
18:     Update:  $\mathcal{L}_{joint} = \mathcal{L}_{PVAE} + \mathcal{L}_{SVAE} + \lambda\mathcal{L}_{RL}$ 
19:   end for
20: end for
```

11.3 Data Preprocessing

11.3.1 TCGA Preprocessing

- Log2 normalization: $x_{norm} = \log_2(x + 1)$
- Gene selection: Filter to top G variable genes
- Standardization: $x_{std} = \frac{x - \mu}{\sigma}$

11.3.2 SMILES Preprocessing

- Tokenization: Split SMILES into tokens (atoms, bonds, rings, etc.)
- Padding: Pad sequences to max length
- Special tokens: Add start, end, and padding tokens

11.3.3 GDSC Preprocessing

- Match cell lines to TCGA profiles
- Match drugs to ChEMBL SMILES
- Filter: Keep only triplets with valid IC50 values
- Split: Train/validation/test splits

12 Conclusion

This document presents the mathematical foundations for genotype-guided de novo molecule generation. The architecture combines:

- Dual VAE system for encoding transcriptomic profiles and molecular structures
- Gaussian addition for integrating multi-modal latent representations
- Transformer-based decoder for autoregressive molecule generation
- Reinforcement learning for optimizing molecular efficacy (IC50)

The mathematical framework provides a solid foundation for implementation and experimentation with alternative approaches.

References

References

- [1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2014.
- [2] J. N. Weinstein, E. A. Collisson, G. B. Mills, K. R. M. Shaw, B. A. Ozenberger, K. Ellrott, I. Shmulevich, C. Sander, and J. M. Stuart, "The cancer genome atlas pan-cancer analysis project," *Nature genetics*, vol. 45, no. 10, pp. 1113–1120, 2013.
- [3] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, *et al.*, "ChEMBL: a large-scale bioactivity database for drug discovery," *Nucleic acids research*, vol. 40, no. D1, pp. D1100–D1107, 2012.
- [4] W. Yang, J. Soares, P. Greninger, E. J. Edelman, H. Lightfoot, S. Forbes, N. Bindal, D. Beare, J. A. Smith, I. R. Thompson, *et al.*, "The genomics of drug sensitivity in cancer (gdsc): a resource for therapeutic biomarker discovery in cancer cells," *Nucleic acids research*, vol. 41, no. D1, pp. D955–D961, 2012.
- [5] Y. Yoshikai, T. Mizuno, S. Nemoto, and H. Kusuvara, "A novel molecule generative model of vae combined with transformer for unseen structure generation," *[Journal/Conference - to be updated]*, [Year - to be updated].
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [7] H. Li, Z. Gao, L. Kang, H. Zhang, K. Yang, K. Yu, X. Luo, W. Zhu, K. Chen, J. Shen, *et al.*, "Large-scale exploration and assessment of chemical drug space with predicted target profiles," *Journal of chemical information and modeling*, vol. 55, no. 4, pp. 843–856, 2015.
- [8] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.
- [9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [10] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β -vae," *arXiv preprint arXiv:1804.03599*, 2018.
- [11] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2018.

- [12] A. Van Den Oord, O. Vinyals, *et al.*, “Neural discrete representation learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [13] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, “Automatic chemical design using a data-driven continuous representation of molecules,” *ACS central science*, vol. 4, no. 2, pp. 268–276, 2018.
- [14] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, “Grammar variational autoencoder,” *International conference on machine learning*, pp. 1945–1954, 2017.
- [15] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, “Molecular de-novo design through deep reinforcement learning,” *Journal of cheminformatics*, vol. 9, no. 1, pp. 1–14, 2017.
- [16] M. Popova, O. Isayev, and A. Tropsha, “Deep reinforcement learning for de novo drug design,” *Science advances*, vol. 4, no. 7, p. eaap7885, 2018.
- [17] Y. Hasin, M. Seldin, and A. Lusi, “Multi-omics approaches to disease,” *Genome biology*, vol. 18, no. 1, pp. 1–15, 2017.