A. Details of Fine-Tuning

Algorithm 1: MMCP-GEN Fine-tuning Loop

Input: Frozen modality encoders $\{E^{(m)}\}_{m=1}^{M}$, frozen GVP structure encoder, DPLM backbone with cross-attention adapters;
Trainable: projection heads $\{W^{(m)}\}$, modality indicator tokens $\{t^{(m)}\}$, MMCP-LQ, adapter params, projection g, hyperparam γ
for *each training iteration (batch)* **do**
  Sample batch of training examples $\{(x, \{c^{(m)}\}, s)\}$;
  `// Encode each present modality with frozen experts and project`
  for *each modality m present in the example* **do**
    $H^{(m)} \leftarrow E^{(m)}(c^{(m)})$; `/* frozen encoder outputs (per-token) */`
    $Z^{(m)} \leftarrow W^{(m)}(H^{(m)})$; `/* project to` $\mathbb{R}^{d_{\text{cond}}}$ `*/`
    $\tilde{Z}^{(m)} \leftarrow [\,t^{(m)}; Z^{(m)}\,]$; `/* prepend modality indicator (MMCP-IH) */`
  $C \leftarrow$ concatenate all present $\tilde{Z}^{(m)}$; `/* global condition token stream */`
  `// Fuse multi-modal conditions via MMCP-LQ`
  $Z(c) \leftarrow$ MMCP-LQ(C) ; `/* outputs` K `summary vectors */`
  `// Inject into DPLM and perform denoising step`
  Inject $Z(c)$ into selected DPLM layers (cross-attn adapters);
  $(\hat{x}, h(x)) \leftarrow$ DPLM_forward$(x, Z(c))$; `/*` $\hat{x}$`: logits,` $h(x)$`: sequence embedding */`
  $L_{\text{CE}} \leftarrow$ CrossEntropy$(\hat{x}, x_{\text{target}})$;
  if *structure s present* **then**
    $z_{\text{str}} \leftarrow$ GVP(s) ; `/* frozen structural encoder */`
    $p \leftarrow g(h(x))$; `/* project sequence embedding */`
    $L_{\text{GS}} \leftarrow$ InfoNCE(p, z_{str}) ; `/* batch contrastive loss */`
  else
    $L_{\text{GS}} \leftarrow 0$
  $L \leftarrow L_{\text{CE}} + \gamma\, L_{\text{GS}}$;
  Backpropagate L; update parameters: $\{W^{(m)}, t^{(m)}\}$, MMCP-LQ, adapter params, g;

A.1. Forward Process

Let a protein sequence be denoted by $x^{(0)} = (x_1^{(0)}, \ldots, x_L^{(0)})$, where each residue $x_i^{(0)}$ takes values in a finite vocabulary $\mathcal{V}$ (the 20 amino acids plus a special MASK token).

Forward (corruption) — absorbing categorical Markov process. Unlike continuous diffusion models that add Gaussian noise, DPLM adapts diffusion ideas to this categorical domain via an *absorbing* discrete Markov corruption process and a learned reverse denoising chain. The forward noising process is formulated as a site-wise, time-homogeneous Markov chain of length T, which progressively replaces residues with the MASK token. Starting from the clean sequence $x^{(0)}$, it corrupts residues until reaching the fully masked state $x^{(T)}$, where at each intermediate step $t = 1, \ldots, T$, every position i is updated independently according to a categorical distribution:

$$q\big(x_i^{(t)} \mid x_i^{(t-1)}\big) = \mathrm{Cat}\Big(x_i^{(t)};\ \underbrace{\beta_t\, e_{x_i^{(t-1)}}}_{\text{stay}} + \underbrace{(1-\beta_t)\, e_{\text{MASK}}}_{\text{absorb}}\Big), \tag{18}$$

where e_v denotes the one-hot vector for token v, and $0 \le \beta_t \le 1$ is a schedule parameter. Two immediate consequences are important: *(i) Absorbing mask.* If $x_i^{(t-1)} = \text{MASK}$, then the right-hand side of (18) collapses to $\mathrm{Cat}(\cdot; e_{\text{MASK}})$, i.e. a masked site remains masked (the MASK is an absorbing state). *(ii) Site-wise independence and Markov property.* The forward kernel factorizes over positions, and satisfies the Markov property $q(x^{(t)} \mid x^{(<t)}) = q(x^{(t)} \mid x^{(t-1)})$.

$$q\big(x^{(t)} \mid x^{(t-1)}\big) = \prod_{i=1}^{L} q\big(x_i^{(t)} \mid x_i^{(t-1)}\big), \tag{19}$$

By iterating (18) one obtains a closed-form marginal of the corrupted sequence given the clean sequence:

$$q\big(x^{(t)} \mid x^{(0)}\big) = \prod_{i=1}^{L} \mathrm{Cat}\Big(x_i^{(t)};\ \alpha_t\, e_{x_i^{(0)}} + (1-\alpha_t)\, e_{\mathrm{MASK}}\Big),$$
$$\alpha_t \triangleq \prod_{s=1}^{t} \beta_s. \tag{20}$$

Equation (20) follows by induction and shows that each site independently retains the original residue with probability α_t or is masked with probability $1-\alpha_t$. As $t \to T$, $\alpha_t \to 0$, ensuring that the distribution converges to the stationary all-MASK state at step T.

A.2. Reverse Process

Reverse (denoising) — multi-modal composable conditions. To support multimodal controllability, **MMCP-GEN** extends the reverse process by conditioning on any subset of auxiliary signals $c \in \{c_{\text{str}}, c_{\text{lig}}, c_{\text{text}}, c_{\text{func}}, \ldots\}$:

$$p_\theta(x^{(t-1)} \mid x^{(t)}, c) = \sum_{\hat{x}^{(0)}} q(x^{(t-1)} \mid x^{(t)}, \hat{x}^{(0)})\, p_\theta(\hat{x}^{(0)} \mid x^{(t)}, c), \tag{21}$$

where $q(\cdot)$ denotes the forward posterior and the conditional distribution $p_\theta(\hat{x}^{(0)} \mid x^{(t)}, c)$ is parameterized by **MMCP-GEN**. Each modality is first processed by a dedicated encoder, yielding modality-specific token sets $\{z_m\}$. To make these representations composable, we introduce Modality Indicator Heads (MMCP-IH) that inject learnable embeddings into each token $\tilde{z}_m$, marking its origin and allowing the model to distinguish contributions across modalities. These updated tokens are then aggregated by a *Learnable Query Fusion* module (MMCP-LQ), where a small set of trainable queries $\{q_j\}$ cross-attend to $\{\tilde{z}_m\}$, producing a compact fused representation $Z(c)$ that flexibly integrates whichever modalities are available. The conditional representation is incorporated into the denoising process through cross-attention adapters, yielding updated hidden states:

$$h'(x^{(t)}, c) \;=\; h(x^{(t)}) + \mathrm{CrossAttn}\big(h(x^{(t)}), Z(c)\big), \tag{22}$$

which are finally projected into residue-level logits:

$$p_\theta(\hat{x}^{(0)} \mid x^{(t)}, c) = \mathrm{Softmax}\Big(W\, h'(x^{(t)}, c)\Big), \tag{23}$$

Through this design, **MMCP-GEN** enables flexible composition of multimodal conditions: any subset of structural, ligand, textual, or functional annotations can be seamlessly fused to guide reverse denoising, without large-scale retraining or architecture changes. Moreover, its conditioning interface allows new modalities to be incorporated with minimal adaptation.

A.3. Re-fine-tuning with newly introduced modality conditions

When introducing a new modality m_{new}, **MMCP-GEN** extends its conditioning space in a fully modular manner. Specifically, a pretrained encoder $E^{(m_{\text{new}})}$ is attached to process the raw modality input (e.g., a ligand graph, cryo-EM density map, or experimental annotation). The encoder outputs a sequence of embeddings:

$$\mathbf{h}_i^{(m_{\text{new}})} = E^{(m_{\text{new}})}(u_i), \quad \mathbf{h}_i^{(m_{\text{new}})} \in \mathbb{R}^{d_{m_{\text{new}}}}, \tag{24}$$

where u_i denotes the i-th element of the raw modality input (e.g., an atom, residue, or token). A lightweight projection head then maps these embeddings into the shared conditional token space:

$$\mathbf{z}_i^{(m_{\text{new}})} = W^{(m_{\text{new}})}\mathbf{h}_i^{(m_{\text{new}})} + b^{(m_{\text{new}})}, \quad \mathbf{z}_i^{(m_{\text{new}})} \in \mathbb{R}^{d_{\text{cond}}}. \tag{25}$$

To explicitly mark the modality origin, we prepend a learnable *indicator head* token $t^{(m_{\text{new}})} \in \mathbb{R}^{d_{\text{cond}}}$ to the sequence (*In our design, a single indicator token per modality is sufficient, allocating multiple tokens provides no additional benefit and only increases parameter cost. Consequently, each modality's token stream is preceded by exactly one learned indicator token. This parameter-efficient formulation provides the model with a strong modality-specific label, thereby enhancing cross-modal interaction without introducing significant overhead*), yielding the augmented token set:

$$\tilde{Z}^{(m_{\text{new}})} = \big[t^{(m_{\text{new}})};\, \mathbf{z}_1^{(m_{\text{new}})}, \ldots, \mathbf{z}_n^{(m_{\text{new}})}\big]. \tag{26}$$

Extending MMCP-LQ. The fusion module MMCP-LQ is augmented by introducing a fresh set of modality-specific queries $Q_{m_{\text{new}}} = \{q_j^{m_{\text{new}}}\}_{j=1}^{N_m}$, which are trained to selectively attend to $\tilde{Z}^{(m_{\text{new}})}$. To accelerate convergence and encourage reuse of generic cross-modal patterns, we initialize:

$$q_j^{m_{\text{new}}} = q_j^s + \epsilon_j, \quad \epsilon_j \sim \mathcal{N}(0, \sigma^2 I), \tag{27}$$

where q_j^s denotes the j-th shared query in Q_s.

Training. When incorporating a new modality, only the following parameters are updated: the projection head $(W^{(m_{\text{new}})}, b^{(m_{\text{new}})})$, the indicator head $t^{(m_{\text{new}})}$, the modality-specific queries $Q_{m_{\text{new}}}$, the adapter modules. The pretrained encoders and DPLM backbone remain frozen. The optimization is performed end-to-end using the diffusion denoising loss $\mathcal{L}_{\text{CE}}$, optionally combined with the joint generation-and-scoring objective when structural conditions are present.

A.4. Robustness to Missing Modalities

A.4.1. Fine-Tuning

In practice, not all training samples contain a complete set of modality conditions. To address this, we adopt two complementary strategies. First, we introduce *condition dropout*, where a subset of modalities is randomly masked during training even if available. This prevents the model from over-relying on any single modality and encourages robustness under incomplete conditioning. Second, when a modality is missing, its projection head does not produce valid tokens, instead, we insert a learnable *missing-modality token* $\mathbf{z}_{\text{miss}}^{(m)}$ into the conditional token set. Each modality maintains its own $\mathbf{z}_{\text{miss}}^{(m)}$ as a dedicated trainable parameter, updated jointly with the adapters and query modules. Modality-specific queries for the absent modality still attend to $\mathbf{z}_{\text{miss}}^{(m)}$, ensuring that they continue to receive gradients and explicitly learn to interpret the absence of modality information. Together, modality dropout and missing-modality tokens establish a stable training scheme that balances modality-agnostic robustness with modality-specific specialization.

A.4.2. Inference

At inference time, **MMCP-GEN** naturally supports flexible combinations of available modalities. When all modalities are provided, queries attend to their corresponding token sets as usual. If certain modalities are missing, the system mirrors the training phase by automatically inserting the corresponding learned missing-modality tokens $\mathbf{z}_{\text{miss}}^{(m)}$, rather than requiring manual user input. In this way, modality-specific queries can reliably distinguish between "present but weak condition" and "absent modality." This design enables the model to gracefully degrade under incomplete conditions (e.g., using only backbone + ligand without functional annotations), instead of failing or producing unstable generations. Empirically, this yields robust conditional generation across diverse subsets of modalities, while preserving high quality when full multimodal information is available.

B. Dataset Curation, Implementation Details & Evaluation Metrics

B.1. Dataset

B.1.1. Dataset Curation

The multimodal dataset curated for training **MMCP-GEN** integrates protein sequences, 3D backbone structures, ligand-binding interactions, and various functional annotations across multiple biological contexts. This dataset was specifically designed to facilitate training of a generative model capable of working under diverse, multimodal conditions. This appendix provides a detailed explanation of the dataset curation process, data sources used, and the precise number of entries in each category.

Protein Sequence Collection. To ensure a reliable foundation for our dataset, we collected protein sequences from the UniProtKB/SwissProt database, renowned for its comprehensive manual curation and high-quality functional annotations. We started with a total of 573,661 proteins sourced from the Swiss-Prot subset.

To ensure the proteins in our dataset are functionally annotated, we filtered out sequences that did not have at least one functional label (such as GO, IPR, or EC). After this filtering step, the final collection includes 127,342 non-redundant protein sequences. These proteins span a wide range of organisms and are annotated with Gene Ontology (GO) terms, InterPro (IPR) domains, and Enzyme Commission (EC) numbers, ensuring their functional and structural relevance.

Functional Annotations: GO, IPR, and EC Numbers. Our dataset includes several layers of functional annotations to provide a well-rounded representation of protein activities.

Gene Ontology (GO): We began with all proteins annotated with GO terms, resulting in 124,736 proteins. After filtering, we retained only the GO terms that were associated with at least 120 proteins, resulting in 353 distinct molecular function terms.

InterPro (IPR): Similarly, we included proteins associated with IPR domains, starting with 108,947 proteins. After applying the same filtering criterion (at least 120 proteins per domain), we retained 1,092 distinct IPR domains.

Enzyme Commission (EC) Numbers: For enzyme-related proteins, we included EC numbers that describe enzymatic activity. Starting with 32,561 proteins, we retained proteins associated with 419 distinct EC classes, after applying the filtering criterion of at least 120 proteins per EC class.

This filtering strategy, applied to GO, IPR, and EC annotations, mirrors the approaches used in ProteoGAN and CFP-GEN. The intent behind this approach is to mitigate the impact of long-tail distributions, ensuring a more balanced representation of functional annotations in the dataset. This strategy effectively reduces the skew caused by rare annotations, making the dataset more suitable for training generative models that require diverse and representative input data.

The coverage across GO, IPR, and EC annotations ensures that the dataset is both diverse and rich in functional categories, suitable for complex protein generation tasks under various conditions.

Structural Data from the Protein Data Bank (PDB). To capture structural features critical to protein function, we incorporated 3D structural data from the Protein Data Bank (PDB). We applied a quality filter by retaining only protein structures with a resolution of 3.5 Å, ensuring structural reliability. After filtering, our dataset contains 72,351 protein structures with backbone atomic coordinates (N, CA, C, O), ensuring comprehensive structural data for protein generation tasks. This structural data plays an important role in constraining the generative model to produce realistic, biologically plausible sequences.

We also incorporated structural information from AlphaFold DB for proteins lacking experimental structures in the PDB, expanding the structural coverage of our dataset.

Ligand-Binding Data from BioLiP. To model protein-ligand interactions, we included data from the BioLiP database, which catalogs protein-ligand interactions. We specifically used the subset of small-molecule binding complexes, which contains 501,832 interactions. These interactions represent associations between proteins and a wide array of ligands, with over 56,000 unique ligands included in the dataset.

To ensure the quality and diversity of the interactions, we removed ligand-protein pairs that were associated with fewer than 10 proteins. This filtering step ensures that only well-represented protein-ligand pairs remain in the dataset, allowing the generative model to focus on high-quality, widely observed interactions. This component of the dataset is crucial for studying ligand binding, and is particularly useful for applications in drug design and functional genomics.

Final Dataset Composition. After curating and merging data from the above sources and applying the necessary filtering steps, the final multimodal dataset comprises:

- 127,342 non-redundant protein sequences (after filtering out proteins without functional annotations).
- 353 distinct GO terms across molecular function, biological processes, and cellular components (after retaining only GO terms associated with at least 120 proteins).
- 1,092 distinct IPR domains across various protein families and structural motifs (after filtering IPR terms with at least 120 associated proteins).
- 419 EC classes related to enzyme activity and catalytic roles (after retaining EC classes with at least 120 associated proteins).
- 56,000 unique ligands, enabling ligand-binding studies and drug discovery.

These proteins span a wide range of functional categories, including enzyme activity, binding, catalytic processes, and structural roles, ensuring the dataset's comprehensive nature.

B.1.2. Validation Set Construction

To ensure unbiased evaluation, we constructed separate validation sets for both GO term prediction and IPR domain prediction:

For GO term validation, we randomly selected 30 protein sequences per GO term, resulting in a validation set of 10,590 sequences (covering 353 GO terms).

For IPR domain validation, we further downsampled the GO validation set using 10-fold cross-validation, yielding a more manageable subset of 1,065 sequences for IPR evaluation.

These validation sets help to assess the model's ability to generalize across diverse functional labels and structural motifs.

B.2. Implementation Details

Computational Resources and Model Details. MMCP-GEN adopts the same training protocol and diffusion hyperparameters as DPLM, unless otherwise noted. All experiments are performed on 8 NVIDIA RTX 4090 GPUs with a global batch size of 1M tokens (*Training is conducted on 8 × RTX 4090 GPUs with* max_tokens*=4096 and* $accumulate\_grad\_batches$*=32, yielding a global batch size of approximately 1M tokens per update*). Training runs for roughly 95 hours and is optimized with AdamW using a maximum learning rate of 4×10^{-5} under a linear warm-up and cosine decay schedule. All training is

conducted in mixed precision (FP16) with gradient checkpointing to reduce memory usage and improve efficiency. During inference, the model performs 100 discrete diffusion sampling steps, consistent with the conditional generation setup in DPLM. Excluding the frozen encoder, **MMCP-GEN** contains approximately 0.9B parameters, comprising the DPLM backbone, multimodal projection heads, modality-indicator embeddings, and the learnable-query fusion module. The total computation amounts to about 7.6×10^2 GPU-hours.

Each modality encoder output is linearly projected into a shared conditional token space of dimension $d_{\text{cond}} = 1024$, aligned with the DPLM embedding dimension. On top of these projected representations, we introduce a learnable-query (LQ) fusion module that aggregates cross-modal condition. The module maintains 5 groups of 4 trainable queries (20 in total), each of dimension 1024, which attend to the modality-specific latent tokens generated by frozen encoders. The aggregated query representations are then processed through 3 lightweight transformer layers for multimodal interaction and fusion. Since the encoder outputs already encode high-level semantic features, a shallow fusion block is sufficient to capture the inter-modal dependencies without excessive computational overhead. This configuration adds approximately 40M parameters (including projection heads), leading to an overall model size of about 0.9B parameters when excluding the frozen structural encoder. Empirically, increasing the number of transformer layers or learnable queries results in only marginal accuracy gains ($< 0.5\%$ AAR improvement) while substantially increasing computation cost. Therefore, we find the 5×4 query configuration and 3-layer transformer depth to achieve a favorable balance between representational capacity, training stability, and efficiency.

Effect of Adapter Insertion Locations in the DPLM Backbone. We integrate lightweight cross-attention adapters into the DPLM backbone to enable multimodal conditioning. Specifically, adapters are inserted after the feed-forward sublayer in the last third of the transformer stack (layers 24, 28, and 33 out of 33). This placement balances computational efficiency with representational depth. The adapters allow multimodal conditions to modulate high-level protein representations without interfering with the low-level syntactic features learned during pretraining.

To assess the impact of adapter insertion locations on performance, we experiment with various insertion configurations across different layers of the transformer. We test five distinct insertion patterns, with adapters inserted at the following layers:

- Shallow layers: 1-8
- Shallow + Middle layers: 1-8, 9-16
- Middle + Deep layers: 9-16, 17-24
- Shallow + Deep layers: 1-8, 25-33
- Deep layers: 24-33

Among all configurations, we find that inserting adapters at layers 24, 28, and 33 results in the best performance. The performance under these configurations is shown in **Figure 6**.

Hyperparameter Tuning for γ and δ. We jointly tune the hyperparameters γ and δ that control the guidance signal (GS) loss and conditioning strength. Rather than fixing one hyperparameter and varying the other, we perform a grid search over both dimensions to assess their interaction. This approach allows us to identify the optimal setting for both hyperparameters simultaneously.

The performance of the model is evaluated using inverse folding (AAR) and GO-term prediction (MRR), with the results indicating that the best performance is achieved at $\gamma = 0.3$ and $\delta = 0.6$. This setting provides the best trade-off between accuracy and generation quality across both tasks. The effect of these hyperparameters on model performance is shown in **Figure 7**, where the performance landscape for AAR and MRR is depicted. As shown in the figure, the combination of $\gamma = 0.3$ and $\delta = 0.6$ consistently yields the highest AAR (78.66) and MRR (0.873), confirming the optimal values for these hyperparameters.

Implementation of Baseline PLMs& DLMs (Function Condition). To ensure a rigorous comparison, we adapt each baseline model in accordance with its design assumptions and supported modalities:

- **ProGen2**: We follow its prompt-based generation protocol by conditioning on the first 30 residues of the target sequence together with GO annotations. For EC and IPR tasks, only sequence-derived context is provided, as ProGen2 does not support these labels.
- **ProteoGAN**: Inputs are restricted to GO terms. Since the model only supports a fixed vocabulary of 50 GO categories, out-of-vocabulary terms are mapped to their closest supported ancestor in the GO hierarchy. Sequences with unmappable annotations are excluded.
- **DPLM**: We employ its discrete diffusion framework by conditioning on short functional motifs (30 residues) and applying inpainting to reconstruct the missing subsequence.
- **ESM-3**: Conditioning combines IPR domain descriptors and their start–end positions with the first 30 residues, enabling domain-level semantics to be integrated with sequence-level prompts.

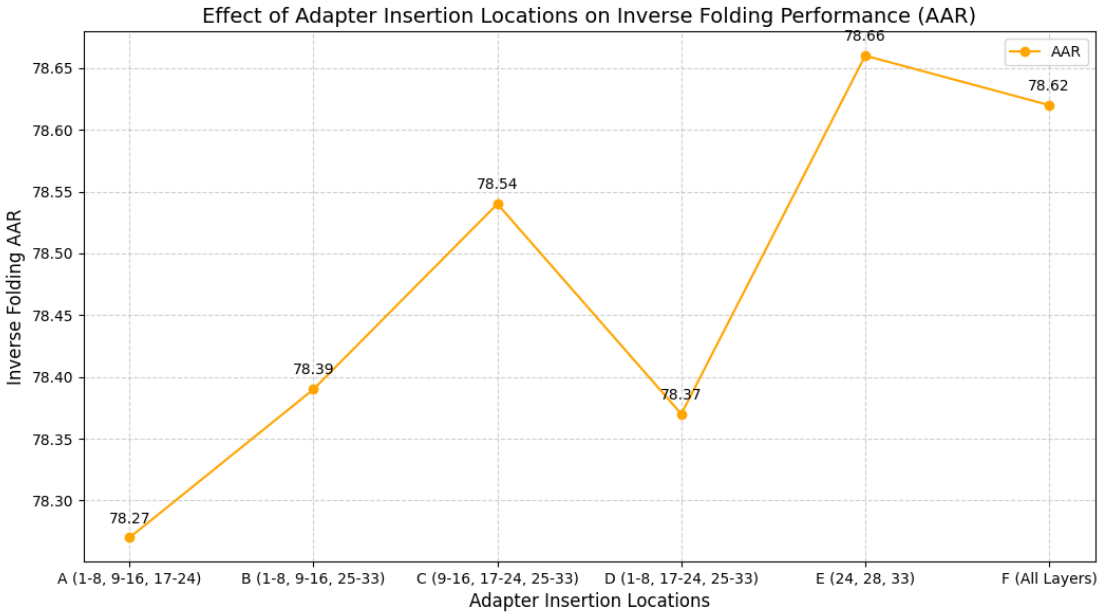

Figure 6. Performance landscape with respect to the adapter insertion locations in the DPLM backbone. The best performance is achieved with adapters inserted at layers 24, 28, and 33.

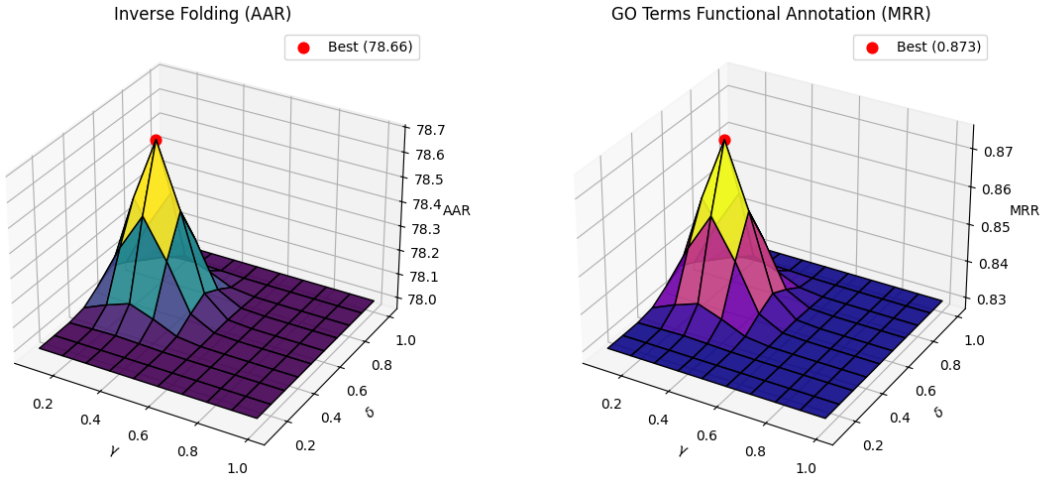

Figure 7. Performance landscape with respect to the hyperparameters γ and δ. Left: Inverse Folding accuracy (AAR). Right: GO terms prediction accuracy (MRR). Both tasks benefit from jointly tuning these hyperparameters, with optimal performance observed at $\gamma = 0.3$ and $\delta = 0.6$.

- **ZymCTRL**: Since ZymCTRL only supports a single EC number per input, we evaluate under a single-label setup. Sequences with unsupported EC annotations are discarded for consistency.
- **CFP-GEN**: We adopt its modality-aware conditioning strategy, incorporating GO and IPR annotations as auxiliary conditions. Supervised fine-tuning (SFT) setting is considered in line with the original protocol.

Following prior work, we condition all baseline models on the first 30 residues, which provides sufficient local functional/structural context (e.g., motifs) while ensuring comparability across methods without trivializing the generation task.

Implementation of Baseline PLMs& DLMs (Inverse Folding). For the inverse folding experiments, we adapt each baseline to our experiments as follows:

- **ProteinMPNN**: Used as released, trained on high-resolution PDB structures. We provide the full backbone coordinates as input.

- **LM-DESIGN**: We follow its official protocol with a UniProt-pretrained language model and a GVP-Transformer encoder trained on the CATH dataset.
- **DPLM**: Evaluated with its pretrained discrete diffusion backbone and GVP-Transformer structural encoder, also trained on CATH.
- **CFP-GEN**: Supervised fine-tuning (SFT) settings are considered. SFT fine-tunes the structural adapter on their in-house dataset with backbone inputs.

B.3. Evaluation Metrics

Table 3. **Evaluation metrics for MMCP-GEN.** Metrics cover sequence distribution, structure consistency, functional annotation, property-level predictions, and unconditional generation quality.

Aspect	Metric	Definition / Formula
Sequence-level consistency	MRR (Mean Reciprocal Rank)	$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i}$, where r_i is the rank of the correct residue at position i.
	MMD / MMD-G	$\mathrm{MMD}^2(X,Y) = \frac{1}{\lvert X\rvert^2}\sum_{x,x'}k(\phi(x),\phi(x')) + \frac{1}{\lvert Y\rvert^2}\sum_{y,y'}k(\phi(y),\phi(y')) - \frac{2}{\lvert X\rvert\lvert Y\rvert}\sum_{x,y}k(\phi(x),\phi(y))$.
Structure-level consistency	TM-score	$\mathrm{TM}(x,s) = \max\frac{1}{L}\sum_i \frac{1}{1+(d_i/d_0)^2}$, measuring fold similarity between predicted and reference backbones.
	AAR (Amino Acid Recovery)	$\mathrm{AAR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}[\hat{y}_i = y_i]$, fraction of correctly recovered residues in masked prediction.
	pLDDT (Predicted Local Distance Difference Test)	Confidence score from structure prediction models (e.g., AlphaFold/ESMFold), reflecting foldability (0–100 scale).
	pdb-TM (Structure Novelty)	$\text{pdb-TM}(x) = \max_{s\in\mathrm{PDB}}\mathrm{TM}(x,s)$, maximum TM-score between a generated structure and all PDB entries; lower is more novel.
	inner-TM (Structural Diversity)	$\text{inner-TM}(X) = \frac{1}{\lvert X\rvert^2}\sum_{x,x'\in X}\mathrm{TM}(x,x')$, average TM-score among generated samples; lower indicates higher diversity.
	Foldability (Aggregate)	Overall structural quality summarized by $\mathrm{Foldability} = \mathbb{E}[\mathrm{pLDDT}]$, assessing geometric plausibility across sequence lengths.
Functional annotation accuracy	Micro-F_1	$F_1 = \frac{2PR}{P+R}$, aggregated across all functional labels.
	Macro-F_1	Same formula, averaged per label (GO, IPR, EC categories).
	AUPR	$\mathrm{AUPR} = \sum_{i=1}^{n}(R_i - R_{i-1})\cdot P_i$, area under precision–recall curve.
	AUC	$\mathrm{AUC} = \Pr(\hat{y}_+ > \hat{y}_-)$, probability a positive is ranked above a negative.
Property-level prediction	Thermostability	Binary classification; ROC-AUC from $\Pr(\hat{y}_{\text{stable}} > \hat{y}_{\text{unstable}})$.
	Metal ion binding	Multi-label classification (e.g., Ca^{2+}, Mg^{2+}, Zn^{2+}), evaluated with micro-F_1.
	HumanPPI	Binary classification of human protein–protein pairs; evaluated with AUC and AUPR.

C. Model Size and Performance Analysis

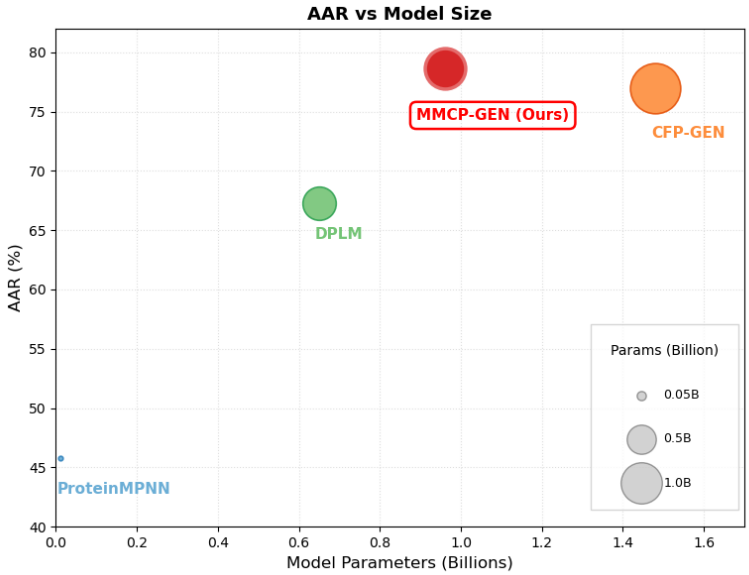

Figure 8. AAR vs Model Size. **MMCP-GEN** achieves the best trade-off between accuracy and model efficiency.

To further examine the efficiency of **MMCP-GEN**, we compare the number of trainable parameters and corresponding sequence recovery (AAR) across representative protein generation models, including ProteinMPNN, DPLM, and CFP-GEN. As shown in Figure 8, **MMCP-GEN** achieves the best overall accuracy among all models, while using substantially fewer parameters than CFP-GEN. This demonstrates that its conditioning mechanism effectively reuses pretrained knowledge without retraining the backbone, highlighting the efficiency and scalability of our framework.

D. Extensibility & Conflict

We simulated post-hoc extension by adding **Ligand** to a Structure+Function+Text MMCP-GEN. CFP-GEN's invasive adaptation **complicates integration** (**+0.83B** params for **3** conditions vs. our **+0.33B** for **4**); stacking Standard Adapters (DPLM-style) leads to **condition conflicts**: Tab. 4 shows combined scores (-6.65) are **worse** than Ligand-only (-6.73), proving destructive interference. We feature a fixed fusion interface to simply instantiate a new query set and reuse it. As **Fig.** 9 shows, we mitigate **conflicts**, getting better performance (**-7.34**) and lower cost (∼**9h vs** ∼**14h**).

Table 4. Post-Hoc Extension Comparison.

Extension Method	**Training Cost (Time)**	**Adaptation Complexity**	**Conflict Resolution**	**Performance (Vina Dock↓)**
Pre-trained MMCP-GEN (w/o Ligand)	-	-	-	-5.76
DPLM (with structure+ligand adapter)	∼14h	Med (Insert Layers)	Weak (Linear Summation)	-6.65
DPLM (with ligand-only adapter)	∼10h	Med (Insert Layers)	-	-6.73
MMCP-GEN (Ours)	**∼9h**	**Low (Reuse Interface)**	**Strong (Attentive Fusion)**	**-7.34**

E. Ablation Study

Component Ablation Study. Replacing **MMCP-LQ** with *Concatenation* degrades AAR (**78.66%** → **75.10%**, Tab. 5). It isolates benefit of our design: unlike Concatenation, which forces the backbone to process dimension-expanding inputs, LQ projects multimodal signals into a unified space via learnable queries. This mechanism **harmonizes heterogeneous conditions** and reduces the **cost** of long sequences. Removing **MMCP-IH** drops AAR to **76.36%**, showing that indicator heads are vital to prevent feature-space **modality confusion** (**Fig.** 9). Removing **GS Loss** reduces structural consistency (scTM **0.912** → **0.906**).

Table 5. Ablation Study on Inverse Folding.

Configuration	**AAR (%)**	**scTM**	**Note**
Full MMCP-GEN	**78.66**	**0.912**	All Conditions & GS Loss (SFT)
w/o MMCP-IH	76.36	0.883	Raw features only → Modality confusion
w/o MMCP-LQ (Concat)	75.10	0.879	Direct concatenation baseline
w/o GS Loss	78.17	0.906	CE Loss only (Lower structural fidelity)
Baseline (DPLM)	*67.24*	*0.876*	*Reference (Same backbone, Structure-Only)*

Table 6. Ablation on LQ Transformer Depth.

Depth (L)	**Params**	**Relative Cost**	**AAR (%)**
1	∼15M	∼ 0.8×	77.24 (*Underfitting*)
3	**∼40M**	**1.0×**	**78.66 (Best Balance)**
6	∼75M	∼ 1.6×	78.81 (*Diminishing returns*)

LQ Depth Ablation Study. Tab. 6 shows $L = 1$ lacks capacity (dropping AAR by **1.42%**). Increasing depth **beyond 3** (e.g., $L = 6$) incurs ∼**60%** overhead for marginal gain (**0.15%**). Thus, $L = 3$ balances accuracy and efficiency.

F. Functional Annotation Process

We map functional annotation (e.g., *EC:1.1.1.1*) to its UniProt-curated descriptions (*alcohol dehydrogenase*). Although ProtBERT is not trained for language understanding, it treats descriptions as structured string patterns rather than semantics; recurring substrings (e.g., *dehydrogenase*) yield consistent fingerprints. While this does not explicitly preserve hierarchical

information, such statistical regularities provide signals exploited by learnable queries. Empirically, this representation is sufficient: MMCP-GEN achieves SoTA performance in functional prediction (main paper Tab. 1).

G. How Q_s & Q_m Attend to Different Modalities

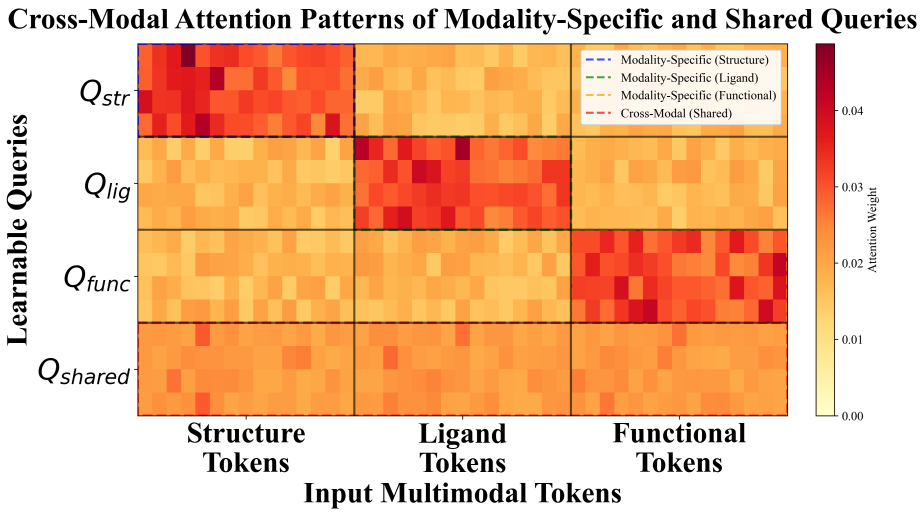

Figure 9. Attention Heatmap: 16 Queries (Y) and input tokens (X). Sequences are truncated to 15 tokens/modality for clarity.

Fig. 9 reveals distinct query roles. In implementation, the initial Q_m are initialized by broadcasting the corresponding Indicator Head $t^{(m)}$ across the query group with perturbations, creating an inductive bias that locks attention to specific modalities. For extensions, since existing queries are frozen, gradients compel new queries (initialized from Q_s) to align with $t^{(new)}$ and attend to corresponding tokens. These yield block-diagonal Q_m attention that **disentangles modalities**, while Q_s aggregate **cross-modal context** via aligned input projections.

H. Ligand-Conditioned Quantification

Table 7. Quantitative Evaluation. Ligand improves *Vina Docking Score* from **-5.76** to **-7.34**.

Conditions	**Vina Dock** (kcal/mol) ↓	**Physical Interpretation**
w/o Ligand	-5.76	Weak / Nonspecific Binding
All Conditions	**-7.34**	**Strong / Native-like Affinity**