RipplePLM: Structural and Property Decoupling for Protein Mutation Effect Generation

Liuzhenghao Lv[1,2,3] **Yuyang Liu**[2,3,†] **Yuyang Gao**[1,2]
Li Yuan[2,3] **Yonghong Tian**[1,2,3,†]

[1]Beijing Key Laboratory of Brain-inspired Spiking Large Models,
School of Computer Science, Peking University, Beijing, China

[2]Guangdong Provincial Key Laboratory of AI for Science and Computing (AI4SC),
School of Artificial Intelligence for Science, Shenzhen Graduate School,
Peking University, Shenzhen, China

[3]School of Electronic and Computer Engineering, Shenzhen Graduate School,
Peking University, Shenzhen, China

[†]Corresponding authors.
Emails: lvliuzh@stu.pku.edu.cn, liuyuyang13@pku.edu.cn, yhtian@pku.edu.cn

Abstract

Protein mutation effect generation asks a model to describe the functional consequence of a point mutation in natural language. Existing protein-to-text systems typically encode mutation information into undifferentiated representations, overlooking the organization of mutation-induced evidence across structural and biochemical factors. We propose RipplePLM, a mutation-aware generation framework centered on Direct-Distal Cross-Attention (DDCA). By constructing a residue-level Mutation Perturbation Field from pre-trained protein language models, DDCA leverages predicted contact maps to organize mutation representations into two pathways: the mutation site's immediate contact neighborhood and its multi-hop distal context. To complement this structural decomposition, we further introduce the Property Latent Chain (PLChain), which injects expert-guided supervision of biochemical property changes (e.g., thermostability and optimal pH) into the LLM hidden-state pathway through latent property tokens. On MutaDescribe, RipplePLM improves over mutation-specific baselines on temporal and structural splits; under a matched-backbone comparison, average structural-split ROUGE-L increases from 22.23 to 35.65. Expert evaluation further shows a higher proportion of biologically accurate or relevant descriptions than the mutation-specific baseline. Additional ablations, representation diagnostics, and low-N fitness regression experiments further support the effectiveness of the learned mutation-aware representations. Code: https://github.com/Lyu6PosHao/RipplePLM.

1 Introduction

Understanding the functional consequences of protein mutations is a central problem in protein engineering, disease variant interpretation, and biological discovery [1, 2]. While many mutation-effect models focus on scalar prediction [3, 4, 5, 6], experimental findings are often communicated as natural-language statements, such as whether a mutation weakens activity, alters stability, changes binding, or shifts an enzyme's preferred condition [7, 8, 9, 10]. These observations motivate *mutation effect generation*, where the goal is to generate a natural-language description of the functional consequence of a given protein mutation [10]. This task is challenging because the model must ground fluent text in mutation-specific evidence. A useful description should therefore reflect not only

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the substituted residue itself, but also its structural neighborhood [11, 12, 13], more distal contact context [3, 14], and property-level cues [15]. Rather than treating a mutation as a monolithic sequence change, a principled generation framework should reflect the underlying biophysical dependencies through which a substitution perturbs its immediate structural environment, propagates through the contact network, and gives rise to measurable biochemical changes that shape the functional outcome.

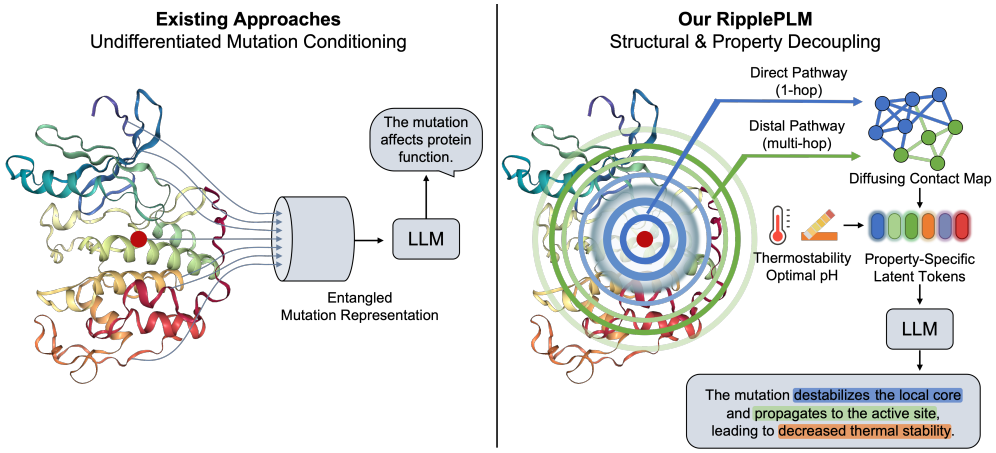


Figure 1: Comparison between existing mutation-effect generation approaches and RipplePLM. Existing methods encode mutation information into undifferentiated representations that obscure structural dependencies and property-level signals. RipplePLM instead organizes mutation representations through direct contact context, distal multi-hop context, and property-aware latent supervision, enabling more structured and biologically grounded generation.

As shown in Figure 1, existing protein–language systems [16, 17, 18, 19, 20, 21] typically align protein encoders with LLMs by projecting sequence or structure representations into the text model. Such interfaces are effective for broad protein captioning and instruction following, but mutation effect generation requires structured conditioning over mutation-specific evidence. MutaPLM [10] moves closer to the target task by modeling protein deltas and using literature-derived reasoning supervision, yet its latent representations remain structurally undifferentiated and do not explicitly separate local contact evidence from distal contact context. In parallel, expert models for biochemical properties such as thermostability and optimal pH provide useful mutation-level signals [22, 23, 24, 25]. Yet ordinary prompt text reduces these signals to coarse symbolic cues and leaves property supervision disconnected from the latent representations that drive generation [26, 27].

We propose RipplePLM, a mutation effect generation framework centered on Direct-Distal Cross-Attention (DDCA). RipplePLM first constructs a residue-level Mutation Perturbation Field (MPF) by subtracting wild-type ESM-2 representations from mutant representations [28]. DDCA then uses the predicted contact map to produce two complementary mutation summaries: a *direct* pathway that attends to the mutation site's contact neighborhood, and a *distal* pathway that aggregates multi-hop contact-graph context. These summaries organize mutation-specific structural evidence into explicit direct and distal pathways before text generation. Complementarily, we introduce Property Latent Chain (PLChain), which injects property-specific supervision into the LLM hidden-state pathway through expert-guided latent tokens for changes in thermostability and optimal pH. Together, DDCA and PLChain implement structural and property decoupling at the representation level: DDCA organizes mutation evidence by contact relationships, while PLChain applies property-change supervision to the hidden states of property tokens.

We evaluate RipplePLM on the temporal and structural splits of MutaDescribe, including a matched-backbone MutaPLM reimplementation for a fairer comparison. RipplePLM improves over evaluated mutation-specific baselines across the main generation metrics, while ablations and representation analyses support the effectiveness of both the structural decomposition and property-aware supervision mechanisms. Expert evaluation further shows a higher proportion of biologically accurate or relevant descriptions than reported for MutaPLM. Our contributions are threefold:

- We introduce DDCA, a contact-guided mutation representation module that separates direct contact-neighborhood evidence from distal contact-graph context;

- We propose PLChain, an auxiliary latent property regularizer that integrates expert signals into the LLM hidden-state pathway;
- We provide a focused empirical study combining lexical and expert-based biological evaluations with ablations and representation diagnostics.

2 Method

Given a wild-type sequence $\mathbf{s}^{\text{wt}}$ of length L, a point mutation m at position i, and optional textual context c (the wild-type function description in our experiments), the task requires producing a description $\mathbf{y}$ of the mutation's functional consequences. We denote the ESM-2 embedding dimension as d_e, the MPF hidden dimension as d, and the number of attention heads as H. The overview of RipplePLM is shown in Figure 2.

2.1 Mutation Perturbation Field (MPF)

Our analysis (Section 3.4) shows that mutation-induced changes in ESM-2 representations extend beyond the mutation site and vary with distance on the predicted contact graph. We define MPF as a residue-indexed latent representation field formed by the differences between mutant and wild-type hidden states of a frozen ESM-2 encoder. DDCA organizes this field into direct (contact-level) and distal (multi-hop) components through contact-guided cross-attention, using predicted contact relationships as structural priors for mutation-effect generation.

Perturbation field. We encode $\mathbf{s}^{\text{wt}}$ and $\mathbf{s}^{\text{mt}}$ with a frozen ESM-2, extracting final-layer representations $\mathbf{H}^{\text{wt}}, \mathbf{H}^{\text{mt}} \in \mathbb{R}^{L\times d_e}$. The perturbation field is:

$$\mathbf{\Delta} = \mathbf{H}^{\text{mt}} - \mathbf{H}^{\text{wt}} \in \mathbb{R}^{L\times d_e}, \tag{1}$$

where $\mathbf{H}^{\text{mt}}$ is obtained by applying mutation m to $\mathbf{s}^{\text{wt}}$. We project $\mathbf{\Delta}$ through layer normalization and a linear layer to obtain $\hat{\mathbf{\Delta}} \in \mathbb{R}^{L\times d}$, and extract the site feature $\mathbf{z}_{\text{site}} = \hat{\mathbf{\Delta}}_i$. Using the full field $\mathbf{\Delta}$ is important because ESM-2 contextualizes each residue by the whole sequence. Changing one amino acid can therefore change hidden states away from the mutated position. Mean pooling this field would remove information about where the perturbation appears, while using only $\mathbf{\Delta}_i$ would discard non-local changes. DDCA keeps the field spatially indexed until the contact-guided aggregation step.

Direct-Distal Cross-Attention (DDCA). We leverage the contact map $\mathbf{C} \in [0,1]^{L\times L}$ predicted from the wild-type sequence $\mathbf{s}^{\text{wt}}$ by the frozen ESM-2 contact head. ESM-2 has been shown to capture structural contacts with high accuracy from sequence alone [29, 28], thereby avoiding dependence on external structure prediction at inference. The wild-type map provides a shared pre-mutation reference topology for the direct and distal pathways to organize mutation-induced representation changes. Let $\mathbf{m} \in \{0,1\}^L$ be the non-padding mask and $\tilde{\mathbf{C}} = \mathbf{C} \odot (\mathbf{m}\mathbf{m}^\top)$ the masked contact matrix; let $\mathbf{v}_i$ be a binary mask that excludes position i itself. Let $\text{Norm}(\mathbf{x}) = \mathbf{x}/(\sum_j x_j + \epsilon)$ with a small ϵ for empty neighborhoods. The *direct* pathway weights are obtained by ℓ_1-normalizing the mutation site's contact row:

$$\mathbf{w}^{\text{dir}} = \text{Norm}\Big(\tilde{\mathbf{C}}_{i,:} \odot \mathbf{v}_i\Big). \tag{2}$$

For the *distal* pathway, we define a structure diffusion kernel with learnable per-head decay $\alpha_h = \sigma(\tilde{\alpha}_h)$ (where σ denotes the sigmoid function) over the row-normalized contact matrix $\bar{\mathbf{C}}$:

$$\mathbf{S}_h = \sum_{k=1}^{K} \alpha_h^k \cdot (\bar{\mathbf{C}}^k)_{i,:}, \tag{3}$$

where K is the diffusion horizon. The distal weights retain multi-hop ($k \geq 2$) contributions:

$$\mathbf{w}_h^{\text{dist}} = \text{Norm}\big((\mathbf{S}_h - \alpha_h \cdot \bar{\mathbf{C}}_{i,:}) \odot \mathbf{v}_i\big), \quad h = 1, \ldots, H. \tag{4}$$

The direct and distal weights play different roles. The direct weights identify the first structural shell around the mutation site, which is the most immediate source of contact-level evidence. The distal weights use powers of the row-normalized contact matrix to reach residues connected through multiple structural steps. The per-head decay α_h controls the relative contributions of different walk

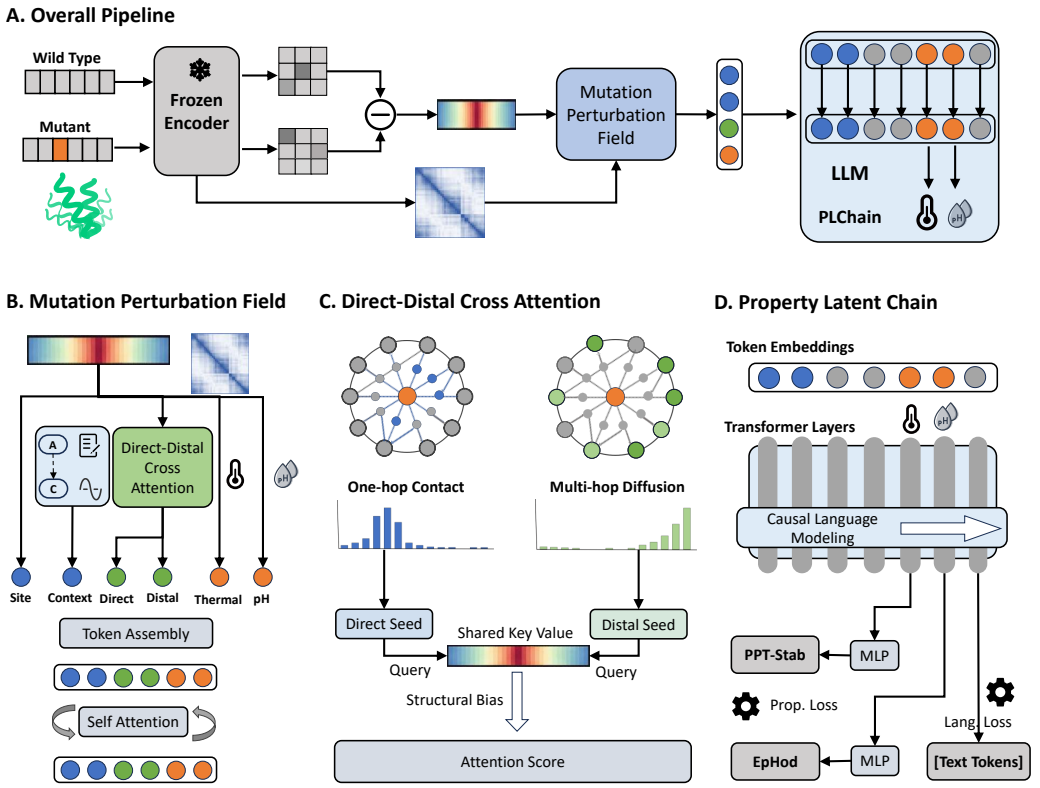


Figure 2: **Architecture of RipplePLM. (A)** Pipeline overview: a frozen ESM-2 extracts the mutation perturbation field $\mathbf{\Delta}$, which MPF compresses into six conditioning tokens for the LLM. **(B)** MPF assembles site, context, DDCA-derived structural tokens (direct/distal), and property queries, refined via self-attention. **(C)** DDCA leverages predicted contact maps to organize mutation representations into direct (1-hop) and distal (multi-hop) cross-attention pathways using additive structural biases. **(D)** PLChain regularizes latent property tokens with expert auxiliary losses while concurrently conditioning the autoregressive generation.

lengths, allowing attention heads to learn different diffusion profiles. Subtracting the first-order term from $\mathbf{S}_h$ retains contributions from walks of lengths 2 through K.

DDCA performs two parallel cross-attention operations over $\hat{\mathbf{\Delta}}$. Each pathway first computes a seed feature $\mathbf{p}$ by weighted-averaging $\hat{\mathbf{\Delta}}$ with its structural weights (direct uses $\mathbf{w}^{\text{dir}}$; distal averages across heads):

$$\mathbf{p}^{\text{dir}} = \sum_j w_j^{\text{dir}} \hat{\mathbf{\Delta}}_j, \quad \mathbf{p}^{\text{dist}} = \tfrac{1}{H} \sum_h \sum_j w_{h,j}^{\text{dist}} \hat{\mathbf{\Delta}}_j. \tag{5}$$

These seeds are projected into pathway-specific queries $\mathbf{q}$, which attend over shared keys/values derived from $\hat{\mathbf{\Delta}}$. The structural weights enter as learnable additive biases on the attention logits, so that each pathway's attention is guided by—but not restricted to—its structural prior:

$$a_{h,j}^{\text{dir}} = \frac{\mathbf{q}_h^{\text{dir}} \cdot \mathbf{k}_{h,j}}{\sqrt{d/H}} + \sigma(\beta_h^{\text{dir}})\, w_j^{\text{dir}}, \qquad a_{h,j}^{\text{dist}} = \frac{\mathbf{q}_h^{\text{dist}} \cdot \mathbf{k}_{h,j}}{\sqrt{d/H}} + \sigma(\beta_h^{\text{dist}})\, w_{h,j}^{\text{dist}}, \tag{6}$$

where $\beta_h^{\text{dir}}, \beta_h^{\text{dist}}$ are learnable scalars controlling the strength of the structural bias. Each pathway produces a pooled vector $\mathbf{o}^{\text{dir}}, \mathbf{o}^{\text{dist}} \in \mathbb{R}^d$. We use additive biases rather than hard structural masks because predicted contacts are noisy and because useful mutation evidence need not be limited to residues selected by the prior. The bias initializes each pathway with a different structural preference, while the content-based attention term can upweight or downweight residues according to the learned generation objective.

During both training stages, we apply a pathway-overlap regularizer to encourage complementary direct and distal structural priors; its definition and the complete training objectives are given in Appendix A.

Token assembly. MPF assembles six tokens: $\mathbf{t}_1$ (site, from $\mathbf{z}_{\text{site}}$), $\mathbf{t}_2$ (direct, from $\mathbf{o}^{\text{dir}}$), $\mathbf{t}_3$ (distal, from $\mathbf{o}^{\text{dist}}$), $\mathbf{t}_4$ (context, from a mutation-type embedding and positional encoding), and two learnable property queries $\mathbf{t}_5, \mathbf{t}_6$ for PLChain. These are stacked into $\mathbf{T}_0 \in \mathbb{R}^{6\times d}$ and refined by self-attention:

$$\mathbf{T} = \text{LayerNorm}(\mathbf{T}_0 + \text{Attention}(\mathbf{T}_0, \mathbf{T}_0, \mathbf{T}_0)) \in \mathbb{R}^{6\times d}. \tag{7}$$

2.2 Property Latent Chain (PLChain)

PLChain introduces a lightweight property-aware pathway for grounding generation in biochemical signals beyond structural perturbation. We focus on thermostability and pH optimum, which characterize the environmental conditions governing protein structure and function: thermostability reflects the ability to retain an active conformation under thermal stress [30], while pH optimum is closely associated with catalytic activity and stability [31, 32]. Modeling changes in these properties aligns auxiliary supervision with the mutation-effect generation objective. Specifically, we construct pseudo-labels for changes in thermostability and pH optimum using PPT-Stab [22] and EpHod [23], respectively. These labels guide the hidden states of *expert-guided latent nodes* within the LLM during training.

Token injection. Special tokens <|protein_token|>, <|thermo_token|>, and <|ph_token|> are added to the LLM vocabulary. During the forward pass, the four structural tokens $\hat{\mathbf{t}}_{1\text{–}4}$ replace repeated <|protein_token|> embeddings, while property queries $\hat{\mathbf{t}}_{5\text{–}6}$ replace <|thermo_token|> and <|ph_token|> via masked scattering.

Expert-guided auxiliary supervision. After the LLM forward pass, lightweight auxiliary heads project the property token hidden states onto expert label spaces:

$$\hat{y}^{\text{thermo}} = \text{MLP}^{\text{thermo}}(\mathbf{h}_{\texttt{thermo}}), \quad \hat{y}^{\text{ph}} = \text{MLP}^{\text{ph}}\big(\mathbf{h}_{\texttt{ph}}\big), \tag{8}$$

aligned with expert pseudo-labels via cross-entropy. The same hidden states also flow into autoregressive generation, so the auxiliary loss acts as a representation regularizer: property tokens are encouraged to encode biochemically grounded information while remaining optimized for generation.

2.3 Training Procedure

RipplePLM is trained in two stages with ESM-2 frozen throughout; this staged approach first learns a structural perturbation interface before introducing the additional property supervision signal, reducing early interference between the auxiliary and language modeling objectives. **Stage 1** trains MPF under the language modeling loss $\mathcal{L}_{\text{LM}}$; property heads are inactive. **Stage 2** activates PLChain with LoRA (r=64) across all projections, optimizing the task objective:

$$\mathcal{L}_{\text{task}}(t) = \big(1 - w_{\text{prop}}(t)\big)\mathcal{L}_{\text{LM}} + w_{\text{prop}}(t)\mathcal{L}_{\text{prop}}, \tag{9}$$

where $w_{\text{prop}}(t)$ is annealed from 0.4 to 0.2. This schedule lets expert supervision shape the property token representations early in training, then gradually gives more weight to the language modeling objective so that generation quality is not dominated by the auxiliary task in later stages.

3 Experiments

3.1 Experimental Setup

Dataset and metrics. We evaluate on MutaDescribe [10], a benchmark providing wild-type sequences, point mutations, and functional text descriptions. We test generalization using two splits: the *temporal split* (chronological partitioning to mitigate information leakage) and the *structural split* (stratified by maximum sequence homology to proteins in the training set, ranging from *Easy* to *Hard*, to assess robustness against distribution shifts). Following prior work [10], we evaluate generation quality using BLEU-2, BLEU-4 [33], ROUGE-1, ROUGE-2, ROUGE-L [34], and METEOR [35]. We report the full suite on the temporal split and use ROUGE-L and BLEU-2 as summary metrics for the structural split.

Table 1: Performance comparison on the **temporal split**. Best results are in **bold**. B/R denote BLEU/ROUGE. *: same base LLM as RipplePLM.

Type	Model	B-2↑	B-4↑	METEOR↑	R-1↑	R-2↑	R-L↑
General	ProLLaMA	0.69	0.21	3.33	0.83	0.04	0.80
	Galactica-6.7B	3.50	1.31	5.61	7.44	0.85	6.17
	Mol-Instructions	0.58	0.08	4.90	5.41	0.13	4.55
Conditioned	GPT-4 (5-shot, kNN)	9.30	4.25	15.08	13.92	2.29	11.84
	AugmentedESM	7.00	3.12	11.29	12.03	2.84	10.12
	Fine-tuned ESM-2	6.90	3.83	13.86	14.21	4.63	12.62
Mutation-spec.	MutaPLM	10.83	6.15	17.84	18.99	6.92	16.51
	MutaPLM*	9.44	3.93	15.28	21.45	7.93	18.22
	RipplePLM	**13.41**	**6.68**	**25.23**	**31.15**	**16.00**	**28.56**

Baselines. We benchmark our framework against four categories of baselines: (1) general protein–language models (ProLLaMA [19], Galactica-6.7B [36], Mol-Instructions [20]); (2) prompted LLMs (GPT-4-0613 [37]); (3) protein-conditioned generation frameworks (AugmentedESM [38], fine-tuned ESM-2 [28], GPT-4 + ESM-2, GPT-4 + OntoProtein [39]); and (4) the state-of-the-art mutation-specific generation model, MutaPLM [10]. To control for LLM backbone choice, we introduce MutaPLM*, a retrained version of MutaPLM using the same base LLM as RipplePLM. For a fair comparison, we use the authors' official implementation and released hyperparameters. This setup enables a controlled comparison between the two frameworks under the same LLM backbone.

Implementation. RipplePLM employs a frozen ESM-2 (`esm2_t30_150M_UR50D`) as the foundational protein encoder, paired with DeepSeek-R1-Distill-Llama-8B [40] as the autoregressive LLM backbone, adapted via LoRA [41]. Within the DDCA module, we configure $H=8$ attention heads and a structural diffusion horizon of $K=4$. Implementation details, including backbone selection rationale, dataset configuration, training hyperparameters, optimization settings, and inference protocols, are provided in Appendix A.

3.2 Main Results

Temporal split. Table 1 details the performance on the temporal split, which explicitly tests generalization to newly documented mutations. RipplePLM consistently outperforms all evaluated baselines across all six metrics. Notably, compared to the previous state-of-the-art MutaPLM, it yields substantial absolute improvements (e.g., $+7.39$ METEOR, $+12.05$ ROUGE-L). Since METEOR accounts for synonym matches, the gain suggests improved alignment with reference descriptions beyond exact-word matching. Furthermore, against the matched-backbone MutaPLM*, RipplePLM still increases ROUGE-L from 18.22 to 28.56, supporting the effectiveness of our framework under the same LLM backbone.

Structural split. Table 2 details performance across varying degrees of structural novelty, serving as a rigorous stress-test for out-of-distribution (OOD) generalization. While baseline models suffer severe performance degradation as test proteins deviate structurally from the training manifold, RipplePLM maintains robust generation fidelity, achieving a striking $+13.42$ absolute gain in average ROUGE-L over the matched-backbone MutaPLM*. More importantly, RipplePLM demonstrates exceptional structural resilience: its performance on the *Hard* subset (ROUGE-L 31.27) surpasses the previous state-of-the-art MutaPLM's performance on the *Easy* subset (ROUGE-L 25.80). This cross-difficulty comparison supports the ability of DDCA's contact-guided representations to generalize across varying degrees of structural similarity to the training set.

Additional evaluation. Expert evaluation yields a higher proportion of biologically accurate or relevant descriptions for RipplePLM than reported for MutaPLM (48.56% vs. 40.06%; Table 8). RipplePLM also remains ahead of MutaPLM across all six temporal metrics after truncating over-length outputs to their reference lengths (Table 9). Bootstrap confidence intervals characterize the sampling uncertainty of the reported scores (Table 10). Recent general-purpose LLMs evaluated in a one-shot setting remain well below RipplePLM on the structural split (Table 11), highlighting the

Table 2: Performance comparison on the **structural split**. R-L denotes ROUGE-L and BL-2 denotes BLEU-2. Best results are in **bold**. *: same base LLM as RipplePLM.

	Easy		Medium		Hard		Average	
Model	R-L	BL-2	R-L	BL-2	R-L	BL-2	R-L	BL-2
ProLLaMA	1.02	0.64	1.00	0.91	1.03	0.70	1.02	0.74
Mol-Instructions	5.10	0.65	5.19	0.65	5.56	0.90	5.27	0.73
Galactica-7B	6.53	3.52	7.64	3.58	7.33	2.88	7.13	3.33
GPT-4 (1-shot)	8.04	2.93	9.96	3.42	9.62	2.69	9.14	3.00
GPT-4 (5-shot)	10.46	2.51	10.31	2.81	10.79	1.88	10.52	2.40
GPT-4 (5-shot, kNN)	11.63	9.63	12.98	10.88	12.46	8.63	12.31	9.69
GPT-4 + ESM-2	11.69	11.09	13.02	11.50	12.77	8.48	12.45	10.37
GPT-4 + OntoProtein	11.84	10.93	12.69	11.22	12.81	8.17	12.42	10.13
AugmentedESM	11.60	8.33	11.40	7.46	10.73	6.95	11.26	7.62
Fine-tuned ESM-2	20.49	9.37	11.87	5.95	11.34	3.32	14.88	6.36
MutaPLM	25.80	18.77	21.07	12.59	16.51	8.69	21.34	13.61
MutaPLM*	24.91	11.67	23.71	12.13	18.80	8.17	22.23	10.66
RipplePLM	**42.91**	**22.57**	**31.56**	**12.67**	**31.27**	**11.91**	**35.65**	**15.78**

importance of mutation-specific modeling for this task. Detailed protocols and results are provided in Appendix E.

Qualitative case studies. Beyond aggregate metrics, qualitative examples further illustrate the advantages of structured mutation conditioning. As shown in Appendix D, RipplePLM more consistently preserves mutation-specific mechanistic evidence, including quantitative biochemical changes, interaction targets, and substrate-specific functional effects. For example, on the hard structural-split mutation *N226A*, RipplePLM correctly captures the "100-fold decrease in GTPase activity," whereas MutaPLM* collapses the description into the less specific statement "abolishes GTP hydrolysis." Similarly, for *G93A*, RipplePLM retains the TDIF–TDR interaction context instead of drifting toward a broader developmental phenotype. We also include competitive counterexamples where both models remain imperfect, highlighting realistic failure modes such as missing secondary clauses or incomplete assay-specific details.

3.3 Ablation Studies

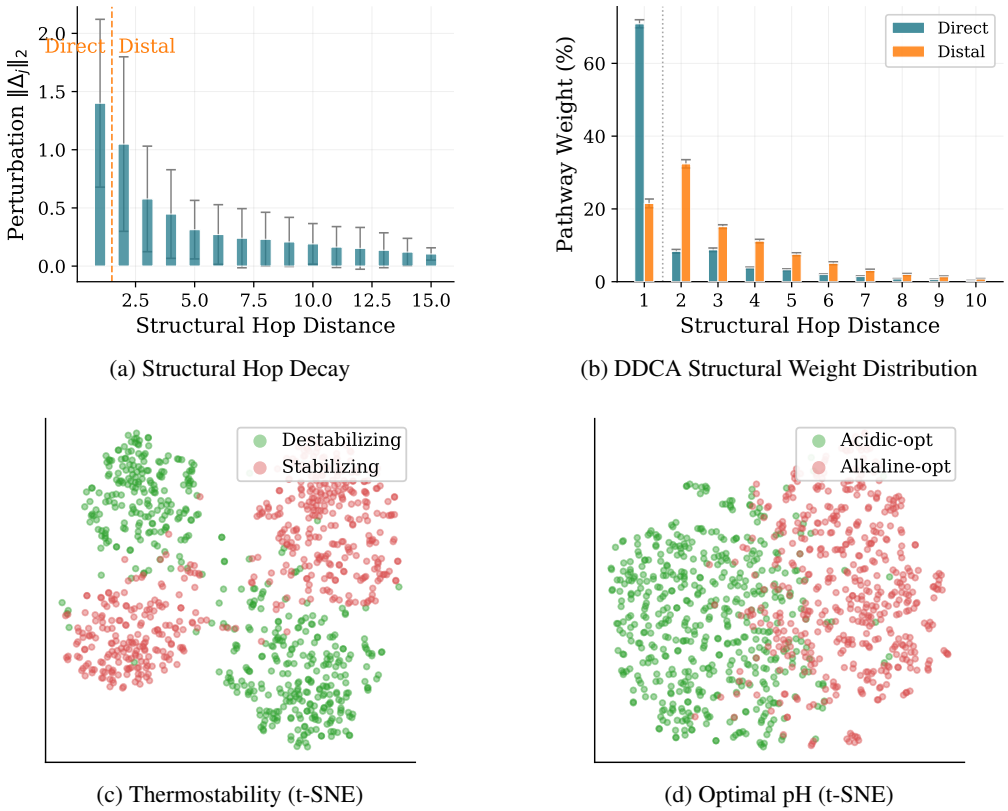
We first ablate the core DDCA design and the auxiliary PLChain signal on the temporal split to isolate their contributions (Table 3). Removing the distal pathway from DDCA degrades BLEU-2 by 1.39 and ROUGE-L by 3.18, validating the necessity of capturing multi-hop structural context beyond the immediate contact shell. Removing PLChain incurs a drop of 2.06 in BLEU-2 and 3.90 in ROUGE-L, supporting the added value of continuous property supervision.

Table 3: **Ablation study on the temporal split.** Replacing PLChain with discrete prompt labels or removing architectural components consistently degrades performance.

Variant	BLEU-2↑	ROUGE-L↑
RipplePLM (Full)	**13.41**	**28.56**
w/o Distal Pathway	12.02	25.38
Discrete Labels	11.70	25.06
w/o PLChain	11.35	24.66

To further examine how property information is incorporated, we replace the latent tokens with discrete text labels in the LLM prompt. This variant also underperforms the full model, supporting the benefit of PLChain's latent regularization over discrete text conditioning in this setting.

Function context and encoder size. Removing the wild-type function description reduces RipplePLM's structural-split ROUGE-L from 35.65 to 28.55, retaining 80.1% of its score and exceeding the published no-function MutaPLM result by 10.01 points (Table 12). Replacing the frozen ESM-2 150M encoder with its 650M counterpart improves four of six metrics under the same training



(a) Structural Hop Decay

(b) DDCA Structural Weight Distribution

(c) Thermostability (t-SNE)

(d) Optimal pH (t-SNE)

Figure 3: **Representation Diagnostics. (a)** Mutation perturbation magnitude remains significantly above background across multiple structural hops. **(b)** DDCA structural pathway weights cover different contact neighborhoods: the direct pathway concentrates on 1-hop residues, while the distal pathway assigns the most weight to 2-hop residues. **(c, d)** t-SNE visualizations of PLChain latent tokens after LLM processing show clear, biologically meaningful clustering for thermostability and optimal pH.

settings, with ROUGE-L increasing to 36.43 (Table 13). These experiments examine the contribution of function context and protein encoder capacity; details are provided in Appendix F.

3.4 Representation Analysis

We conduct representational diagnostics to examine whether the learned intermediate states reflect the intended structural and biochemical roles of DDCA and PLChain (Figure 3).

Perturbation decay. We first evaluate the spatial scope of the ESM-2 perturbation field. Across 200 sampled temporal-test proteins, the per-residue perturbation $\|\mathbf{h}_j^{\text{mt}} - \mathbf{h}_j^{\text{wt}}\|_2$ peaks at the mutation site and decays smoothly. Notably, the near-site signal (≤ 5 residues) is $10.77\times$ the background noise (> 100 residues). On the predicted contact graph, this signal remains $9.1\times$ above background at 2 hops and $5.0\times$ at 3 hops (Figure 3a). These elevated representation differences at multi-hop distances motivate a dedicated distal pathway to capture mutation-induced changes beyond direct contacts.

DDCA structural weights and feature separation. The structural pathway weights exhibit different spatial coverage consistent with the DDCA design (Figure 3b). Across the analyzed temporal-test examples, the direct pathway assigns 70.9% of its normalized weight to 1-hop residues, while the distal pathway assigns 21.5% to 1-hop residues and 58.7% to 2–4 hop residues, peaking at 2 hops. These statistics describe the contact-derived structural priors; the sampling and normalization protocol is given in Appendix A. Separately, direct and distal token representations have a silhouette score of 0.74, indicating separation in feature space.

Property-token specificity. For PLChain, we verify that the latent property tokens actively encode the targeted expert knowledge. Matched linear probes trained on the LLM-output property tokens achieve remarkable accuracies of **90.2%** for thermostability and **88.3%** for optimal pH. Visualized via t-SNE (Figure 3c, d), these representations form highly distinct, property-specific clusters (e.g., stabilizing vs. destabilizing mutations). Crucially, the pre-LLM MPF queries hover near random chance ($\sim$55%), demonstrating that property specificity is not trivially embedded in the initial queries, but dynamically formed during the LLM's forward pass guided by the auxiliary loss.

Predicted-contact confidence. The proportion of biologically accurate or relevant descriptions increases from 44.23% to 47.12% and 54.22% across low-, middle-, and high-confidence groups, respectively (Table 14). This association links generation performance to model-reported contact confidence; the grouping protocol is described in Appendix F.3.

3.5 Low-N Fitness Regression

To ascertain whether MPF representations capture generalized biological information beyond text generation, we follow the MutaPLM protocol [10] for low-N fitness regression. Using only 192 labeled single-point mutations for training, we train a lightweight regressor on frozen protein features without any textual supervision. We evaluate on two contrasting ProteinGym datasets: Spike-ACE2 (binding phenotype on a long viral protein) and avGFP (fluorescence on a shorter protein).

Table 4: **Low-N fitness regression** (Spearman ρ, mean$\pm$std over 5 runs). Best mean is in **bold**.

Model	Spike-ACE2	avGFP
Ridge Regression	0.335$\pm$0.052	0.298$\pm$0.071
ESM-2	0.331$\pm$0.041	0.554$\pm$0.013
Augmented ESM	0.363$\pm$0.021	0.497$\pm$0.096
Augmented EVmutation	0.354$\pm$0.044	0.512$\pm$0.034
ConFit	0.412$\pm$0.033	0.564$\pm$0.035
Tranception_L	0.488$\pm$0.040	0.594$\pm$0.019
MutaPLM	0.481$\pm$0.028	0.596$\pm$0.032
RipplePLM	**0.491$\pm$0.061**	**0.598$\pm$0.067**

As detailed in Table 4, RipplePLM achieves the highest mean Spearman correlation on both datasets. The standard deviations overlap with those of the strongest baselines, and the results suggest that the learned mutation representations provide useful features for fitness regression with limited labeled data.

4 Related Work

Protein–language alignment. Protein–language alignment aims to connect protein representations with natural-language descriptions, biomedical knowledge, or instruction-following capabilities [16, 17, 18, 19, 20, 21, 39, 42, 43, 44, 45, 46]. Early approaches such as ProtST [21], OntoProtein [39], and BioTranslator [43] align proteins with text or ontology annotations through contrastive or knowledge-guided pretraining, while Prot2Text [18], ProtT3 [45], and related protein-to-text systems demonstrate protein-conditioned description generation. More recent LLM-based systems, including ProteinChat [16], InstructProtein [17], ProLLaMA [19], Mol-Instructions [20], and ProtLLM [46], project protein representations into general-purpose language models for protein-aware dialogue and instruction following. Although these methods establish the feasibility of protein-conditioned generation, most treat the protein input as a static object. Mutation effect generation instead requires localized and mutation-aware conditioning over wild-type/mutant differences. AugmentedESM [38] and ESM-based generation baselines provide protein-conditioned generation signals, but they do not explicitly separate direct contact-neighborhood evidence from distal structural context. RipplePLM instead introduces contact-guided mutation perturbation representations for structured mutation conditioning.

Mutation effect prediction and generation. Mutation effect prediction has been extensively studied through evolutionary, probabilistic, structure-aware, and protein-language-model-based approaches [1, 3, 4, 5, 47, 48, 49, 50, 51, 52, 53, 54]. Representative methods include EVmutation [3], DeepSequence [4], GEMME [47], EVE [48], ESM-1v [1], Tranception [49], ProGen2 [52], AlphaMissense [53], and ProMEP [54]. Other methods specialize in biochemical properties such as stability, thermostability, solubility, and enzyme optimal pH [11, 13, 22, 23, 24, 25, 55, 56, 57, 58]. However, these systems typically output scalar scores or class labels rather than natural-language explanations.

MutaPLM [10] formalizes mutation effect generation with a protein delta network and literature-mined chain-of-thought-style supervision [59, 60]. RipplePLM differs primarily in its representation decoupling strategy: DDCA organizes mutation representations into direct contact-neighborhood and distal contact-graph pathways before generation, while PLChain introduces auxiliary property-aware latent supervision.

5 Conclusion

We presented RipplePLM, a mutation effect generation framework that organizes mutation evidence through structured conditioning rather than undifferentiated protein representations. At its core, DDCA separates mutation-induced signals into direct contact-neighborhood evidence and distal contact-graph context, while PLChain introduces property-aware latent supervision to incorporate biochemical signals into the generation pathway. Experiments on MutaDescribe show consistent improvements over evaluated mutation-specific baselines on both temporal and structural splits, complemented by expert assessment of biological relevance. Low-N fitness regression further suggests that MPF representations provide useful features for fitness prediction. Future work will extend RipplePLM toward multi-mutation settings and larger community-curated benchmarks that connect mutation descriptions with richer experimental contexts.

Limitations. RipplePLM focuses on single-point mutations, consistent with current mutation-description benchmarks; extending effect generation to multi-mutation settings will require broader community efforts to curate datasets with epistatic annotations. DDCA remains sequence-only and efficient by using ESM-2-predicted contact maps, though its structural signal may depend on contact prediction quality. Performance also depends in part on the availability and quality of wild-type function annotations. Our representation analyses characterize learned mutation signals; linking these signals to biophysical mechanisms requires experimental validation.

Broader Impacts. RipplePLM may assist protein engineering, variant interpretation, and literature-driven biological discovery by generating structured descriptions of mutation effects. Its outputs should be used for hypothesis generation and decision support rather than as substitutes for experimental validation or clinical expertise.

Acknowledgments and Disclosure of Funding

This work was supported by the Guangdong Grants (Grant No. 2023ZT10X075), the National Natural Science Foundation of China (No. 62425101, No. 62606021), and the National Key R&D Program of China (No. 2026ZD0127200).

The authors declare no competing interests.

References

[1] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in neural information processing systems*, 34:29287–29303, 2021.

[2] Nadav Brandes, Grant Goldman, Charlotte H Wang, Chun Jimmie Ye, and Vasilis Ntranos. Genome-wide prediction of disease variant effects with a deep protein language model. *Nature genetics*, 55(9):1512–1522, 2023.

[3] Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta PI Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nature biotechnology*, 35(2):128–135, 2017.

[4] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

[5] Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood Van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, et al. Proteingym: Large-scale benchmarks for protein fitness prediction and design. *Advances in neural information processing systems*, 36:64331–64379, 2023.

[6] Vanessa E Gray, Ronald J Hause, Jens Luebeck, Jay Shendure, and Douglas M Fowler. Quantitative missense variant effect prediction using large-scale mutagenesis data. *Cell systems*, 6(1): 116–124, 2018.

[7] Takeshi Kawabata, Motonori Ota, and Ken Nishikawa. The protein mutant database. *Nucleic acids research*, 27(1):355–357, 1999.

[8] Douglas EV Pires, Tom L Blundell, and David B Ascher. Platinum: a database of experimentally measured effects of mutations on structurally defined protein–ligand complexes. *Nucleic acids research*, 43(D1):D387–D391, 2015.

[9] Uroš Prešern and Marko Goličnik. Enzyme databases in the era of omics and artificial intelligence. *International Journal of Molecular Sciences*, 24(23):16918, 2023.

[10] Yizhen Luo, Zikun Nie, Massimo Hong, Suyuan Zhao, Hao Zhou, and Zaiqing Nie. Mutaplm: Protein language modeling for mutation explanation and engineering. *Advances in Neural Information Processing Systems*, 37:79783–79818, 2024.

[11] Raphael Guerois, Jens Erik Nielsen, and Luis Serrano. Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. *Journal of molecular biology*, 320(2):369–387, 2002.

[12] Catherine L Worth, Robert Preissner, and Tom L Blundell. Sdm—a server for predicting effects of mutations on protein stability and malfunction. *Nucleic acids research*, 39(suppl_2): W215–W222, 2011.

[13] Douglas EV Pires, David B Ascher, and Tom L Blundell. mcsm: predicting the effects of mutations in proteins using graph-based signatures. *Bioinformatics*, 30(3):335–342, 2014.

[14] Bian Li, Dan M Roden, and John A Capra. The 3d mutational constraint on amino acid sites in the human proteome. *Nature Communications*, 13(1):3273, 2022.

[15] Gayatri Ramakrishnan, Coos Baakman, Stephan Heijl, Bas Vroling, Ragna van Horck, Jeffrey Hiraki, Li C Xue, and Martijn A Huynen. Understanding structure-guided variant effect predictions using 3d convolutional neural networks. *Frontiers in molecular biosciences*, 10: 1204157, 2023.

[16] Han Guo, Mingjia Huo, Ruiyi Zhang, and Pengtao Xie. Proteinchat: Towards achieving chatgpt-like functionalities on protein 3d structures. 2023.

[17] Zeyuan Wang, Qiang Zhang, Keyan Ding, Ming Qin, Xiang Zhuang, Xiaotong Li, and Huajun Chen. Instructprotein: Aligning human and protein language via knowledge instruction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1114–1136, 2024.

[18] Hadi Abdine, Michail Chatzianastasis, Costas Bouyioukos, and Michalis Vazirgiannis. Prot2text: Multimodal protein's function generation with gnns and transformers. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 10757–10765, 2024.

[19] Liuzhenghao Lv, Zongying Lin, Hao Li, Yuyang Liu, Jiaxi Cui, Calvin Yu-Chian Chen, Li Yuan, and Yonghong Tian. Prollama: A protein large language model for multi-task protein language processing. *IEEE Transactions on Artificial Intelligence*, 2025.

[20] Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[21] Minghao Xu, Xinyu Yuan, Santiago Miret, and Jian Tang. Protst: Multi-modality learning of protein sequences and biomedical texts. In *International conference on machine learning*, pages 38749–38767. PMLR, 2023.

[22] Purva Tijare, Nishant Kumar, and Gajendra PS Raghava. Prediction and design of thermostable proteins with a desired melting temperature. *Scientific Reports*, 15(1):16683, 2025.

[23] Japheth E Gado, Matthew Knotts, Ada Y Shaw, Debora Marks, Nicholas P Gauthier, Chris Sander, and Gregg T Beckham. Machine learning prediction of enzyme optimum ph. *Nature Machine Intelligence*, 7(5):716–729, 2025.

[24] Huali Cao, Jingxue Wang, Liping He, Yifei Qi, and John Z Zhang. Deepddg: predicting the stability change of protein point mutations using neural networks. *Journal of chemical information and modeling*, 59(4):1508–1514, 2019.

[25] Vineet Thumuluri, Hannah-Marie Martiny, Jose J Almagro Armenteros, Jesper Salomon, Henrik Nielsen, and Alexander Rosenberg Johansen. Netsolp: predicting protein solubility in escherichia coli using language models. *Bioinformatics*, 38(4):941–946, 2022.

[26] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

[27] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[28] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[29] Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=fylclEqgvgd`.

[30] GD Haki and SK Rakshit. Developments in industrially important thermostable enzymes: a review. *Bioresource technology*, 89(1):17–34, 2003.

[31] Kemper Talley and Emil Alexov. On the ph-optimum of activity and stability of proteins. *Proteins: Structure, Function, and Bioinformatics*, 78(12):2699–2706, 2010.

[32] Hans Bisswanger. Enzyme assays. *Perspectives in Science*, 1(1-6):41–55, 2014.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[34] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[35] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[36] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

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

[38] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Learning protein fitness models from evolutionary and assay-labeled data. *Nature biotechnology*, 40(7):1114–1122, 2022.

[39] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Qiang Zhang, Jiazhang Lian, and Huajun Chen. Ontoprotein: Protein pretraining with gene ontology embedding. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=yfe1VMYAXa4`.

[40] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[41] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[42] Liuzhenghao Lv, Hao Li, Yu Wang, Zijun Chen, Zhiyuan Yan, Zongying Lin, Yuyang Liu, Li Yuan, and Yonghong Tian. Navigating chemical-linguistic sharing space with heterogeneous molecular encoding. *Nature Communications*, 2026.

[43] Hanwen Xu, Addie Woicik, Hoifung Poon, Russ B Altman, and Sheng Wang. Multilingual translation for zero-shot biomedical classification using biotranslator. *Nature Communications*, 14(1):738, 2023.

[44] Hao Li, Liuzhenghao Lv, He Cao, Zijing Liu, Zhiyuan Yan, Yu Wang, Yonghong Tian, Yu Li, and Li Yuan. How to detect and defeat molecular mirage: A metric-driven benchmark for hallucination in llm-based molecular comprehension. *arXiv preprint arXiv:2504.12314*, 2025.

[45] Zhiyuan Liu, An Zhang, Hao Fei, Enzhi Zhang, Xiang Wang, Kenji Kawaguchi, and Tat-Seng Chua. Prott3: Protein-to-text generation for text-based protein understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5949–5966, 2024.

[46] Le Zhuo, Zewen Chi, Minghao Xu, He-Yan Huang, Jianan Zhao, Heqi Zheng, Conghui He, Xian-Ling Mao, and Wentao Zhang. Protllm: An interleaved protein-language llm with protein-as-word pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8950–8963, 2024.

[47] Elodie Laine, Yasaman Karami, and Alessandra Carbone. Gemme: a simple and fast global epistatic model predicting mutational effects. *Molecular biology and evolution*, 36(11):2604–2619, 2019.

[48] Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K Min, Kelly Brock, Yarin Gal, and Debora S Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 599(7883):91–95, 2021.

[49] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. In *International Conference on Machine Learning*, pages 16990–17017. PMLR, 2022.

[50] Céline Marquet, Michael Heinzinger, Tobias Olenyi, Christian Dallago, Kyra Erckert, Michael Bernhofer, Dmitrii Nechaev, and Burkhard Rost. Embeddings from protein language models predict conservation and variant effects. *Human genetics*, 141(10):1629–1647, 2022.

[51] Céline Marquet, Julius Schlensok, Marina Abakarova, Burkhard Rost, and Elodie Laine. Expert-guided protein language models enable accurate and blazingly fast fitness prediction. *Bioinformatics*, 40(11):btae621, 2024.

[52] Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.

[53] Jun Cheng, Guido Novati, Joshua Pan, Clare Bycroft, Akvilė Žemgulytė, Taylor Applebaum, Alexander Pritzel, Lai Hong Wong, Michal Zielinski, Tobias Sargeant, et al. Accurate proteome-wide missense variant effect prediction with alphamissense. *Science*, 381(6664):eadg7492, 2023.

[54] Peng Cheng, Cong Mao, Jin Tang, Sen Yang, Yu Cheng, Wuke Wang, Qiuxi Gu, Wei Han, Hao Chen, Sihan Li, et al. Zero-shot prediction of mutation effects with multimodal deep representation learning guides protein engineering. *Cell Research*, 34(9):630–647, 2024.

[55] Elizabeth H Kellogg, Andrew Leaver-Fay, and David Baker. Role of conformational sampling in computing mutation-induced changes in protein structure and stability. *Proteins: Structure, Function, and Bioinformatics*, 79(3):830–838, 2011.

[56] Bian Li, Yucheng T Yang, John A Capra, and Mark B Gerstein. Predicting changes in protein thermodynamic stability upon point mutation with deep 3d convolutional neural networks. *PLoS computational biology*, 16(11):e1008291, 2020.

[57] Lasse M Blaabjerg, Maher M Kassem, Lydia L Good, Nicolas Jonsson, Matteo Cagiada, Kristoffer E Johansson, Wouter Boomsma, Amelie Stein, and Kresten Lindorff-Larsen. Rapid protein stability prediction using deep learning representations. *Elife*, 12:e82593, 2023.

[58] Henry Dieckhaus, Michael Brocidiacono, Nicholas Z Randolph, and Brian Kuhlman. Transfer learning to leverage larger datasets for improved prediction of protein stability changes. *Proceedings of the national academy of sciences*, 121(6):e2314853121, 2024.

[59] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[60] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[61] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE, 2020.

[62] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

A Implementation Details

Model and infrastructure. We use ESM-2 (esm2_t30_150M_UR50D) [28] as the frozen protein encoder and initialize the language decoder from DeepSeek-R1-Distill-Llama-8B. DeepSeek-R1-Distill-Llama-8B is a reasoning-oriented dense language model obtained by distilling the reasoning behaviors of DeepSeek-R1 into the Llama architecture. Unlike standard instruction-tuned LLMs optimized primarily for conversational fluency, DeepSeek-R1-style models are trained to preserve intermediate reasoning patterns during generation.

We adopt this backbone because mutation effect generation is fundamentally an evidence-grounded reasoning task rather than a pure captioning problem. The model must integrate heterogeneous mutation signals, including local structural perturbations, distal contact-mediated propagation, and biochemical property constraints, into coherent functional descriptions. A reasoning-oriented decoder is therefore aligned with the structured conditioning objective of RipplePLM, where generation depends on combining multiple latent evidence pathways rather than decoding from a single undifferentiated representation.

Unless otherwise stated, all ESM-2 parameters are frozen, and the trainable components include the MPF/DDCA modules, PLChain auxiliary heads, the token projection layers, and LoRA adapters in the LLM. All experiments are conducted on a cluster with 8 NVIDIA A800 GPUs, each with 80GB memory. Training uses DeepSpeed ZeRO-2 [61], bfloat16 mixed precision for memory efficiency.

Training hyperparameters. RipplePLM is trained in two stages. Stage 1 performs lightweight adaptation of the mutation-conditioning interface while keeping the LLM update small. In this stage, only the MPF module is trained. We train for 1 epoch on both temporal and structural splits using batch size 6 per GPU, learning rate 2×10^{-4}, and no property supervision. Stage 2 performs full mutation-aware adaptation with PLChain enabled. In this stage, LoRA adapters are applied to all attention and feed-forward projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `gate_proj`, `down_proj`) with rank r=64 and α=128. We train for 10 epochs on the temporal split and 2 epochs on the structural split using batch size 4 per GPU, gradient accumulation 2, and learning rate 5×10^{-5}. PLChain supervision is enabled only in Stage 2 through the thermostability and pH auxiliary heads.

Both stages use AdamW [62] with 5% linear warmup, bfloat16 mixed precision, and DeepSpeed ZeRO-2 [61]. The maximum sequence length is 1024. We use epoch-level checkpointing and retain the best Stage 2 checkpoint according to validation BLEU-4. The effective batch sizes are 48 in Stage 1 and 64 in Stage 2. Training is conducted on 8 NVIDIA A800 GPUs (80GB each). Stage 1 takes approximately 1.2h, while Stage 2 takes approximately 13.3h on the temporal split and approximately 2.7h on the structural split.

Pathway overlap regularization and full training objectives. For each example, we penalize overlap between the normalized direct and distal structural pathway weights:

$$\mathcal{L}_{\text{overlap}} = \frac{1}{H}\sum_{h=1}^{H}\sum_{j=1}^{L} w_j^{\text{dir}}\, w_{h,j}^{\text{dist}}. \tag{10}$$

The loss is averaged over the minibatch. It acts on the contact-derived weights used to construct the pathway seeds and attention biases. With λ_{overlap}=0.05 fixed throughout both stages, the complete objectives are

$$\mathcal{L}_{\text{stage 1}} = \mathcal{L}_{\text{LM}} + \lambda_{\text{overlap}}\mathcal{L}_{\text{overlap}}, \tag{11}$$

$$\mathcal{L}_{\text{stage 2}}(t) = \mathcal{L}_{\text{task}}(t) + \lambda_{\text{overlap}}\mathcal{L}_{\text{overlap}}, \tag{12}$$

where $\mathcal{L}_{\text{task}}(t)$ is the generation–property objective in Section 2.3.

Structural pathway-weight diagnostics. Figure 3b summarizes the structural weights $\mathbf{w}^{\text{dir}}$ and $\mathbf{w}_h^{\text{dist}}$ from DDCA. We analyze 110 temporal-test mutation examples with protein lengths of 100–300 residues and at least three predicted contacts above 0.5 at the mutation site. These criteria limit protein-size variation and ensure a sufficiently populated direct-contact neighborhood. Hop distances are shortest-path distances on the contact graph thresholded at 0.3. Distal weights are first averaged across heads. For each example and pathway, weights are summed by hop distance, restricted to reachable residues at hops 1–10, and renormalized over this range; we then report the mean and

Table 5: Prompt template used by RipplePLM. The protein and property placeholders are replaced by learned latent tokens before being passed to the LLM.

Field	Content
System	`You are an expert biochemist. Your task is to analyze a protein mutation.`
User instruction	`Wild-type protein function: {function description}` `Next is a feature of the mutation {wt} to {mt} at position {pos}. Please generate a {brief summary / long detailed} text to describe it. The feature is` `<|protein_token|><|protein_token|><|protein_token|><|protein_token|>` `<|thermo_token|> <|ph_token|>.`
Assistant target	`{mutation-effect description}`

standard error across examples. This diagnostic characterizes the spatial coverage of the model's structural priors.

Inference. At inference time, the wild-type and mutant sequences are first encoded by frozen ESM-2 to construct the Mutation Perturbation Field and the predicted contact map used by DDCA. Generation uses greedy decoding with maximum output length 1024. PLChain auxiliary losses are disabled during inference, while the learned latent property tokens remain part of the conditioning sequence. Inference is performed on a single NVIDIA A800 GPU with batch size 1. The average end-to-end latency is approximately 1s per mutation, including ESM-2 encoding, DDCA computation, and autoregressive generation.

Compute usage. The reported experiments were conducted on an internal GPU cluster using 8 NVIDIA A800 GPUs (80GB memory each). The main reported runs correspond to one Stage 1 training run and one Stage 2 training run for each split configuration. Based on the measured runtimes, the total training cost for the experiments reported in the paper is approximately 150 GPU-hours. Additional compute was used for preliminary hyperparameter exploration, ablation studies, debugging, and failed runs that are not individually reported in the paper, bringing the estimated total project compute usage to approximately 600 GPU-hours. No large-scale pretraining was performed; all experiments rely on parameter-efficient fine-tuning with frozen ESM-2 representations and LoRA adaptation of the language model.

Dataset. We evaluate RipplePLM on MutaDescribe [10], a mutation effect generation benchmark constructed from literature-derived mutation annotations. Each example contains a wild-type protein sequence, a point mutation, optional textual context, and a natural-language description of the mutation's functional consequence. We follow the official temporal and structural splits. The temporal split evaluates generalization to mutation descriptions from later literature, while the structural split groups test proteins according to their maximum sequence homology to training proteins, computed by MMseqs2. We use the provided mutation-effect descriptions as generation targets and retain the associated protein sequences and mutation identifiers for constructing wild-type/mutant inputs. When property labels are required by PLChain, we use the corresponding pseudo-label columns generated by PPT-Stab and EpHod for thermostability and pH optimum supervision. Detailed dataset construction and split definitions are provided in the original MutaDescribe paper [10].

Prompt format. RipplePLM is trained in a chat-style instruction-following format. Table 5 shows the prompt template used in our experiments. Here, `{wt}` and `{mt}` denote the wild-type and mutant amino acids, respectively, and `{pos}` denotes the 1-indexed mutation position in the protein sequence. The field `{function description}` is the provided wild-type protein function annotation, and `{mutation-effect description}` is the target natural-language description of the mutation's functional consequence.

During training and inference, the four repeated `<|protein_token|>` placeholders are replaced by MPF/DDCA structural tokens derived from the wild-type/mutant perturbation field. The `<|thermo_token|>` and `<|ph_token|>` placeholders are replaced by PLChain latent property-

query tokens for thermostability and pH optimum. The choice between `brief summary` and `long detailed` follows the availability of GPT-augmented descriptions in the training data.

We provide the wild-type protein function description directly as input context. This choice also ensures a fair matched-backbone comparison with MutaPLM*. MutaPLM reports that replacing model-generated function descriptions with ground-truth function descriptions improves performance; therefore, in our MutaPLM* reimplementation, we also provide the ground-truth function description as prompt context. This keeps the comparison focused on the mutation-conditioning mechanism rather than on differences in access to functional context.

B Low-N Fitness Regression

We follow the low-N fitness regression protocol of MutaPLM [10]. Given only 192 labeled single-point mutations for training, we train a lightweight regression head on frozen mutation features and evaluate Spearman correlation over five random runs. For fairness, RipplePLM uses the same frozen ESM-2 feature views as the baseline setting and augments them with MPF features extracted without updating the generator.

RipplePLM obtains the highest mean Spearman correlation on both datasets, although the standard deviations overlap with the strongest baselines. These results provide auxiliary evidence that MPF representations retain fitness-relevant information.

Datasets. We evaluate on two ProteinGym v1.3 datasets: (1) Spike-ACE2: 3802 single-point mutations, wild-type length 1273 aa; (2) avGFP: 1084 single-point mutations filtered from 51714 entries, wild-type length 238 aa.

Training. We use 192 training samples and 48 validation samples. The regression head is trained using mean-squared error.

C Full Structural Split Results

Table 6 reports the full six-metric results for RipplePLM on the structural split.

Table 6: Full RipplePLM results on the structural split across all six metrics.

Level	BLEU-2	BLEU-4	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Easy	22.57	16.07	37.51	45.26	31.75	42.91
Medium	12.67	5.70	24.72	34.65	17.27	31.56
Hard	11.91	5.73	24.30	34.05	17.05	31.27
Average	15.78	9.36	29.30	38.37	22.54	35.65

D Case Studies

To qualitatively analyze model behavior, we compare RipplePLM with MutaPLM* on representative examples from the Easy, Medium, and Hard subsets of the structural split. We select cases that cover partial matches, quantitative effects, interaction-specific descriptions, and competitive counterexamples.

These cases show that RipplePLM is most effective when the reference description depends on mutation-specific structural or mechanistic evidence, such as GTPase reduction, TDIF–TDR interaction, or SEC22B cargo packaging. The examples also reveal realistic error modes: RipplePLM can omit secondary clauses, substitute downstream outcomes, or confuse related biochemical activities. The competitive Q49A case further indicates that structured conditioning improves many mutation-specific descriptions but does not eliminate all enzyme-activity errors.

Table 7: Representative case studies on the structural split of MutaDescribe.

Mutation	Reference	RipplePLM	MutaPLM*
N226A	100-fold decrease in GTPase activity. 5-fold decrease of affinity for GTP.	100-fold decrease in GTPase activity.	Abolishes GTP hydrolysis.
G93A	Compromised interaction with TDR.	Compromised interaction with TDR. The mutation in TDIF reduces its binding affinity to TDR.	Reduced ability to repress tracheary element differentiation.
Q100L	Probable constitutively active mutant unable to hydrolyze GTP; increases degradation of the transferrin receptor.	Probable constitutively active mutant unable to hydrolyze GTP; localizes to the cytoplasm.	In a breast cancer sample; somatic mutation.
A715R	Increased ability to package the SNARE SEC22B cargo into COPII vesicles. Has no effect on other cargos packaged into COPII vesicles.	Increased ability to bind and package the SNARE SEC22B into transport vesicles.	Enables interaction with SEC31A.
Q49A	Reduces ATP binding and ATPase activity in vitro.	Reduces RNA binding affinity. However, no specific functional changes are described for this mutation.	Loss of ATPase activity.

E Additional Generation Evaluation

E.1 Biology-Aware Evaluation

Following the four-category framework in MutaPLM [10] (Appendix C.3, Table A8), a PhD-level biology expert evaluated all 1,248 RipplePLM outputs on the structural test set against their reference descriptions. Each output was assigned to Accurate, Relevant, Opposite, or Irrelevant based on agreement in the affected biological function or process and the direction of the mutation effect. Table 8 reports the expert-reviewed distribution alongside the published MutaPLM results. RipplePLM achieves an Accurate-or-Relevant rate of 48.56% and an Opposite rate of 6.41%; the corresponding rates reported for MutaPLM are 40.06% and 7.13%. This evaluation complements the lexical metrics by assessing agreement with the reference descriptions on biological effects.

Table 8: Biology-aware evaluation on the structural split. MutaPLM results are reported in the original paper [10]; RipplePLM results are based on expert review of 1,248 outputs. Values are percentages.

Category	MutaPLM	RipplePLM
Accurate ↑	17.47	21.55
Relevant ↑	22.60	27.00
Accurate + Relevant ↑	40.06	48.56
Opposite ↓	7.13	6.41
Irrelevant ↓	52.80	45.03

E.2 Response Length

RipplePLM generates an average of 30.65 words per description on the temporal split, compared with 28.03 words in the references. On the structural split, its descriptions are shorter than the references (21.73 vs. 31.87 words). To assess performance under a reference-length constraint, we truncated the 732 temporal outputs that exceeded their reference lengths, leaving the remaining outputs unchanged, and rescored all 1,611 examples using the same tokenizer and evaluation metrics.

As shown in Table 9, truncation increases ROUGE-L from 28.56 to 32.68 and also improves METEOR, ROUGE-1, and ROUGE-2. BLEU-2 and BLEU-4 decrease slightly, while all six scores remain above the published MutaPLM results. Output length is negatively correlated with per-example ROUGE-L on both splits (Pearson $r \approx -0.32$ temporal and -0.24 structural). These results indicate that additional output length is unlikely to be the main source of the generation gains.

Table 9: Temporal-split performance before and after truncating over-length RipplePLM outputs to their reference lengths. B/R denote BLEU/ROUGE. The published MutaPLM scores are included for comparison.

Model / output setting	B-2	B-4	METEOR	R-1	R-2	R-L
MutaPLM	10.83	6.15	17.84	18.99	6.92	16.51
RipplePLM	13.41	6.68	25.23	31.15	16.00	28.56
RipplePLM (truncated)	12.58	6.58	25.59	35.33	18.64	32.68

E.3 Statistical Uncertainty

We estimate 95% bootstrap confidence intervals for RipplePLM by resampling test examples 10,000 times within each split. Table 10 reports intervals for BLEU-2 and ROUGE-L on the temporal ($n = 1,611$) and structural ($n = 1,248$) splits. These intervals characterize sampling uncertainty over test examples. Their lower bounds exceed the corresponding published MutaPLM point estimates on both splits.

Table 10: RipplePLM scores with 95% bootstrap confidence intervals.

Metric	Temporal	Structural
BLEU-2	13.41 [12.43, 14.43]	15.78 [14.42, 17.15]
ROUGE-L	28.56 [27.51, 29.63]	35.65 [34.31, 37.01]

E.4 Recent General-Purpose LLMs

We evaluate GPT-5.4, DeepSeek-V4-Flash, and DeepSeek-V4-Pro on all 1,248 structural-test examples in a one-shot setting without external tools or retrieval. Table 11 compares their results with the reported GPT-4-0613 one-shot baseline and RipplePLM. The recent LLMs achieve broadly comparable scores to GPT-4-0613, while RipplePLM retains a substantial advantage on both metrics.

Table 11: Recent general-purpose LLMs on the structural split. Prompted LLMs use one-shot evaluation; GPT-4-0613 results are from MutaPLM [10].

Model	BLEU-2	ROUGE-L
GPT-4-0613	3.00	9.14
GPT-5.4	2.07	8.49
DeepSeek-V4-Flash	3.13	10.13
DeepSeek-V4-Pro	3.31	9.96
RipplePLM	15.78	35.65

F Additional Model and Input Analyses

F.1 Wild-Type Function Context

RipplePLM and MutaPLM* receive the same ground-truth wild-type function description in the matched-backbone comparison. To assess the contribution of this input, we remove the function description from RipplePLM on the structural split and compare with the function ablation reported in Table 4 of MutaPLM [10]. Table 12 reproduces both MutaPLM conditions from that ablation, whose with-function score differs from the main benchmark result in Table 2.

Removing function context reduces RipplePLM's ROUGE-L by 7.10 points (19.9%), compared with a 5.26-point decrease (22.1%) for MutaPLM. RipplePLM retains 80.1% of its score, exceeds no-function MutaPLM by 10.01 points, and exceeds MutaPLM with the ground-truth function description by 4.75 points. The function descriptions themselves have a ROUGE-L overlap of 9.76 with the mutation-effect references, and no reference is an exact substring of its function description. Function context provides useful information, while the removal control shows that RipplePLM retains its comparative advantage without this input.

Table 12: Effect of wild-type function descriptions on structural-split ROUGE-L. MutaPLM values are from its published function ablation [10], Table 4.

Model	With function	Without function	Drop	Relative drop
MutaPLM	23.80	18.54	5.26	22.1%
RipplePLM	35.65	28.55	7.10	19.9%

F.2 Protein Encoder Size

We replace the frozen ESM-2 150M encoder with ESM-2 650M, adjusting only the MPF input projection to accommodate the embedding dimension and retaining the other hyperparameters and training settings. The larger encoder improves four of six structural-split metrics (Table 13), including METEOR (29.30 to 30.45) and ROUGE-L (35.65 to 36.43), with small decreases in BLEU-2 and BLEU-4. This comparison suggests potential benefits from greater encoder capacity under the existing training configuration. We use the 150M encoder for the main experiments to balance predictive performance and computational cost.

Table 13: Effect of frozen ESM-2 encoder size on the structural split. B/R denote BLEU/ROUGE.

Encoder	B-2	B-4	METEOR	R-1	R-2	R-L
ESM-2 150M	15.78	9.36	29.30	38.37	22.54	35.65
ESM-2 650M	15.12	9.22	30.45	38.84	23.63	36.43

F.3 Predicted-Contact Confidence

We summarize model-reported contact confidence as the mean ESM-2 150M contact probability over the top-L long-range residue pairs ($i < j$, $j - i \geq 24$), where L is the wild-type sequence length. When fewer than L eligible pairs are available, we average over all of them. Prior work on attention-based contact prediction motivates the use of these probabilities as a confidence statistic [29]. Of the 1,248 structural-test examples, 1,247 have eligible long-range contacts; we divide these examples into three approximately equal-sized groups by confidence and apply the expert-reviewed labels from Appendix E.1.

Table 14 shows that the Accurate-or-Relevant rate increases from 44.23% in the lower-confidence group to 54.22% in the higher-confidence group. This analysis measures the association between generation performance and model-reported contact confidence, rather than contact accuracy against experimentally determined structures. The pattern is consistent with the use of predicted contacts as a structural prior, with content-based attention contributing alongside that prior.

Table 14: Biology-aware evaluation across predicted-contact confidence groups on the structural split. Category values are percentages.

Confidence group	N	Accurate	Relevant	Accurate + Relevant
Lower third	416	19.47	24.76	44.23
Middle third	416	19.71	27.40	47.12
Higher third	415	25.54	28.67	54.22

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: NA

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code, data access, and training and evaluation instructions are available at `https://github.com/Lyu6PosHao/RipplePLM`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all the training and test details

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report 95% bootstrap confidence intervals for generation metrics in Appendix E.3 and standard deviations over five runs for low-N fitness regression in Table 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides sufficient information on the computer resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both potential positive societal impacts and negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper follows the safeguards provided by the models and datasets using in this work.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets, used in the paper, are properly credited. And the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: NA

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.