AgentPLM: Agentic Protein Language Models with Reasoning-Augmented Decoding for Protein Sequence Design

Sahil Rahman [1] **Maxx Richard Rahman** [2]

Abstract

Protein language models (PLMs) are passive oracles: they generate sequences in a single forward pass with no mechanism to consult external biophysical feedback or redirect generation when a candidate violates thermodynamic or structural constraints. We introduce AgentPLM, which addresses this by equipping a pre-trained PLM with i) Reasoning-Augmented Decoding (RAD), which interleaves autoregressive generation with tool calls (ESMFold, FoldX, AutoDock Vina), and ii) Contrastive Agent Policy Optimisation (CAPO), a trajectory-level extension of direct preference optimisation that trains the policy end-to-end to learn when oracle feedback is informative rather than merely imitating high-fitness sequences. We evaluate AgentPLM on benchmark tasks spanning de novo enzyme design, antibody optimisation, thermostability, PPI interface design, and zero-shot fitness prediction with standardised oracle APIs and controlled sequence-identity splits. AgentPLM achieves state-of-the-art results with a gain in antibody top-10% hit rate over the strongest passive baseline, providing mechanistic evidence of online error correction without explicit backtracking.

1. Introduction

Protein language models (PLMs) have emerged as one of the most transformative tools in computational biology over the past five years. Models such as ESM-2 (Lin et al., 2023), ProtTrans (Elnaggar et al., 2022), and Ankh (Elnaggar et al., 2023) demonstrate that transformer architectures pre-trained on large sequence databases can encode rich representations of evolutionary, structural, and functional information, powering state-of-the-art methods for structure prediction (Lin et al., 2023), inverse folding (Dauparas et al., 2022), and zero-shot fitness prediction (Notin et al., 2023). The 2024 Nobel Prize in Chemistry, awarded in part for AlphaFold (Jumper et al., 2021), crystallised the scientific community's recognition that machine learning has become indispensable to structural biology. Yet a fundamental limitation pervades this generation of models, i.e., they are static oracles. A PLM generates sequences in a single forward pass, with no mechanism to interrogate the biophysical plausibility of intermediate steps, retrieve relevant experimental data, or adaptively redirect generation when a candidate sequence is predicted to be thermodynamically unstable or immunogenic. This passivity is in sharp contrast to how expert protein engineers actually work, iterating between computational predictions, wet-lab assays, and literature consultation in tightly coupled feedback loops (Arnold, 2018; Yang et al., 2019). The result is a fundamental mismatch, i.e., a PLM should implicitly encode all biophysical constraints in its parameters, yet its training signal (next-token prediction or masked token recovery on evolutionary sequences) does not directly supervise constraint satisfaction, making it systematically brittle for out-of-distribution design targets where the evolutionary record provides little guidance.

The analogous limitation in large language models for reasoning has been addressed. Chain-of-thought prompting (Wei et al., 2022), tool-augmented generation (Schick et al., 2023; Qin et al., 2024), and the ReAct framework (Yao et al., 2023) have demonstrated that interleaving generation with structured reasoning steps and external tool calls dramatically improves performance on complex, multi-constraint tasks. ChemCrow (Bran et al., 2024) applies this paradigm to chemistry and BioPlanner (O'Donoghue et al., 2023) to experimental protocol generation, while ProtAgent (Ghafarollahi & Buehler, 2024) uses a frozen GPT-4 backbone to orchestrate protein engineering tools through chain-of-thought planning. However, these systems treat the language model as a frozen planning module, preventing end-to-end training on domain-specific objectives and forcing the model to rely entirely on its parametric knowledge for sequence-level decisions. We argue that protein design is precisely the setting where this limitation is most damaging, i.e., generating a functional protein sequence requires satisfying simultaneous, non-additive constraints (thermostability, sol-

[1]Bedford College, London, United Kingdom [2]Saarland University, Saarbrücken, Germany. Correspondence to: Sahil Rahman <srahman@bedford.ac.uk>.

Accepted at the 2026 Workshop on Generative and Agentic AI for Biology (ICML 2026)

ubility, binding specificity, immunogenicity) that no single forward pass can jointly optimise, and the training signal needed to learn *when* and *how* to invoke structural oracles is available only through reinforcement on protein engineering objectives, not through general-purpose language modelling.

In this work we introduce AgentPLM, a framework that endows a pre-trained PLM with agentic capabilities by training it end-to-end to interleave autoregressive sequence generation with structured oracle invocations. We model each design step as a decision in a Partially Observable Markov Decision Process over the joint space of partial sequences and retrieved biophysical context, and we train the resulting policy with a novel Contrastive Agent Policy Optimisation (CAPO) objective that rewards trajectories leading to biophysically consistent, high-fitness sequences. The main contributions are:

- We propose Reasoning-Augmented Decoding (RAD), a decoding strategy that interleaves autoregressive sequence generation with structured tool calls, incorporating their outputs via a learned Tool Context Encoder (TCE) and Trajectory Memory Buffer (TMB) trained end-to-end on protein engineering objectives.
- We introduce Contrastive Agent Policy Optimisation (CAPO), which is a trajectory-level extension of direct preference optimisation that contrasts high-fitness trajectories with coherent oracle use against low-fitness or contradictory ones, teaching the model *when* oracle feedback is informative rather than merely imitating high-fitness outputs.
- AgentPLM outperforms all the baselines on all the benchmark tasks, achieving a $2.79\times$ improvement in antibody top-10% hit rate and $+34\%$ in normalised k_{cat}/K_m on enzyme design, confirming that gains arise from qualitatively different reasoning trajectories rather than additional compute.

2. Related Works

2.1. Protein Language Models and Generative Protein Design

ESM-2 (Lin et al., 2023), ProtTrans (Elnaggar et al., 2022), and Ankh (Elnaggar et al., 2023) establish that masked and autoregressive transformers pre-trained on large sequence databases encode rich evolutionary and functional representations, with ESM-2 achieving state-of-the-art zero-shot fitness prediction on deep mutational scanning benchmarks (Notin et al., 2023). Structure-conditioned design is addressed by ProteinMPNN (Dauparas et al., 2022) via autoregressive inverse folding over backbone coordinates, while RFdiffusion (Watson et al., 2023), RFdiffusion-AA (Krishna et al., 2024), and Chroma (Ingraham et al., 2023) extend the paradigm to joint structure-sequence co-design through SE(3)-equivariant diffusion. For fitness-guided design, EvoProtGrad (Emami et al., 2023) climbs fitness landscapes via PLM-guided gradient descent over hundreds of mutation-selection cycles, and Tranception (Notin et al., 2022) improves zero-shot scoring by retrieving homologous sequences at inference time. All of these models share a fundamental limitation: they generate sequences in a single forward pass or through oracle-free perturbations, so structural plausibility, thermodynamic stability, and binding specificity must be encoded entirely in parameters trained on evolutionary data, with no mechanism to observe constraint violations mid-generation and redirect accordingly (Huang et al., 2016; Rahman et al., 2022; Russ et al., 2020).

2.2. Agentic AI and Tool-Augmented Generation for Science

Toolformer (Schick et al., 2023) and ToolLLM (Qin et al., 2024) formalise the insertion of API call tokens into language model generation, while ReAct (Yao et al., 2023) demonstrates that interleaving chain-of-thought reasoning (Wei et al., 2022) with environment actions is mutually reinforcing. These frameworks have been applied to science via ChemCrow (Bran et al., 2024) for organic synthesis and BioPlanner (O'Donoghue et al., 2023) for experimental protocol planning. In protein engineering, ProtAgent (Ghafarollahi & Buehler, 2024) is the closest prior work, using a frozen GPT-4 backbone to orchestrate ESMFold, FoldX, and literature search through chain-of-thought planning, but it shares two critical gaps with all prior tool-augmented systems: (i) the language model cannot be trained end-to-end on protein objectives, preventing the tight coupling between oracle signals and residue-level generation that RAD achieves, and (ii) no system integrates a literature oracle that fuses partial-sequence context with experimental knowledge at the token level, leaving residue-level mutagenesis knowledge inaccessible during generation (Zhang et al., 2022). Prior RL approaches for protein design (Stanton et al., 2022; Shanehsazzadeh et al., 2023) apply reinforcement learning as a search strategy over discrete sequence space rather than as a policy-training objective over multi-step tool-use trajectories. CAPO extends direct preference optimisation (Rafailov et al., 2023) to this trajectory-level contrastive setting, a formulation unexplored in either protein design or tool-augmented generation.

3. Problem Formulation

3.1. Protein Design as a Constrained Sequence Optimisation Problem

Let $\mathcal{A}$ denote the amino-acid alphabet with $|\mathcal{A}| = 20$, and let $\mathcal{A}^* = \bigcup_{L=1}^{\infty} \mathcal{A}^L$ be the set of all finite sequences. A

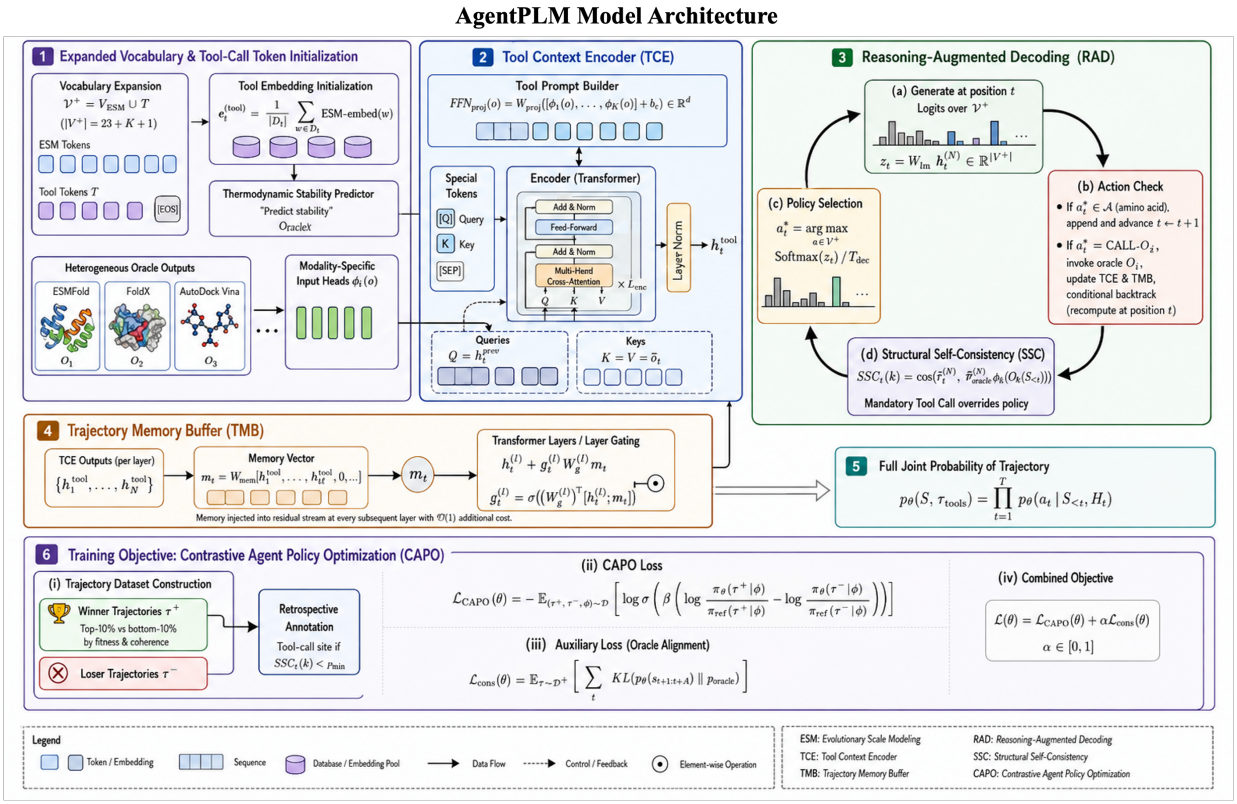

Figure 1. AgentPLM architecture comprising four modules, i) Expanded Vocabulary and Tool-Call Token Initialisation, ii) Trajectory Memory Buffer (TMB), iii) Reasoning-Augmented Decoding (RAD), and iv) trained end-to-end via the CAPO objective.

protein design task is a tuple $\mathcal{T} = \langle f, \mathcal{C}, S_{\text{ref}} \rangle$, where $f : \mathcal{A}^L \to \mathbb{R}$ is a *fitness function* (e.g. catalytic efficiency k_{cat}/K_m, binding affinity K_D, or melting temperature T_m); $\mathcal{C} = \{c_j : \mathcal{A}^L \to \mathbb{R}\}_{j=1}^{J}$ is a set of *biophysical constraint functions* (e.g. prescribed secondary-structure fractions, active-site geometry tolerances, immunogenicity thresholds), and $S_{\text{ref}} \in \mathcal{A}^L$ is an optional reference (wild-type) sequence providing evolutionary context. The objective is:

$$S^* = \arg\max_{S \in \mathcal{A}^L} f(S) \quad \text{subject to} \quad c_j(S) \le \varepsilon_j, \quad \forall j \in [J]. \tag{1}$$

where ε_j are task-specific tolerances. In practice neither f nor c_j are analytically tractable, but can be estimated through a set of computational *oracles* $\mathcal{O} = \{O_k : \mathcal{A}^* \to \mathbb{R}^{d_k}\}_{k=1}^{K}$, where d_k is the output dimensionality of oracle k (e.g. O_{ESMFold} returns per-residue pLDDT scores and Cartesian coordinates; O_{FoldX} returns a scalar $\Delta\Delta G$).

3.2. Limitations of Passive Protein Language Models

A standard autoregressive PLM parameterises the sequence distribution as $p_\theta(S) = \prod_{i=1}^{L} p_\theta(s_i \mid s_{<i})$, where the parameters θ are learned by minimising next-token prediction loss on large sequence databases. During inference the model generates the entire sequence in a single unidirectional pass, we call this a *passive trajectory*:

$$\tau_{\text{passive}} = (S_\emptyset \to S_1 \to \cdots \to S_L), \tag{2}$$

where $S_t = (s_1, \ldots, s_t)$ is the partial sequence after t amino acids have been committed.

Structural blindness. At each step t, the model selects $s_t \sim p_\theta(s_t \mid s_{<t})$ without observing $\{O_k(S_t)\}_{k=1}^{K}$. This is a fundamental mismatch, i.e., all biophysical constraints should be implicitly encoded in θ, yet the training signal (cross-entropy on evolutionary sequences) does not directly supervise constraint satisfaction. Formally, define the *constraint violation* of a passive trajectory as $\mathcal{V}(\tau_{\text{passive}}) = \sum_{j=1}^{J} \max\big(0,\ c_j(S_L) - \varepsilon_j\big)$. Because c_j is never observed during generation, $\mathbb{E}[\mathcal{V}(\tau_{\text{passive}})]$ cannot be minimised online. It can only be reduced by increasing model capacity, a strategy that is both computationally expensive and brittle for out-of-distribution targets.

Epistatic blindness. Long-range epistasis (where the fitness contribution of residue s_i depends non-additively on

a distant residue s_j) is systematically underestimated by autoregressive factorisation. An oracle call at position i that returns $O_{\text{FoldX}}(S_{\leq i})$ can expose such dependencies before positions $j > i$ are committed, enabling corrective decisions.

3.3. Agentic Protein Design

We reformulate protein design as a Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}^+, \mathcal{O}, T, R, \gamma \rangle$:

State space. $\mathcal{X} = \mathcal{A}^* \times \mathcal{H}$, where $\mathcal{H} = \left(\mathcal{A}^* \times [K] \times \mathbb{R}^*\right)^*$ is the space of tool-call histories. The state at step t is $x_t = (S_{\leq t}, H_t)$, with $H_t = \{(S_{\leq t_m}, k_m, o_m)\}_{m=1}^{M_t}$ recording the partial sequence, oracle index, and oracle output of each past tool call.

Augmented action space. $\mathcal{A}^+ = \mathcal{A} \cup \mathcal{T}, \quad \mathcal{T} = \{\text{CALL-}O_k\}_{k=1}^{K} \cup \{\text{EOS}\}$, where $\mathcal{A}$ is the amino-acid vocabulary, $\mathcal{T}$ is the tool-call vocabulary (one token per oracle plus an end-of-sequence token).

Observation function. When $a_t \in \mathcal{T} \setminus \{\text{EOS}\}$, the environment returns the oracle output $o_t = O_k(S_{\leq t})$, which is appended to H_t. When $a_t \in \mathcal{A}$, the observation is a null token and H_t is unchanged.

Transition.

$$x_{t+1} = \begin{cases} (S_{\leq t} \cdot a_t,\ H_t) & \text{if } a_t \in \mathcal{A}, \\ (S_{\leq t},\ H_t \cup \{(S_{\leq t}, k, o_t)\}) & \text{if } a_t = \text{CALL-}O_k. \end{cases} \tag{3}$$

The tool calls do not advance the sequence index t. The agent re-scores position t conditioned on the enriched history.

Reward. $R(\tau) = f(S_L) - \lambda \left|\{t : a_t \in \mathcal{T}\}\right| - \mu\, \mathcal{V}(\tau)$, where $\lambda > 0$ penalises excessive oracle calls (latency cost) and $\mu > 0$ penalises biophysical constraint violations. The agent's policy $\pi_\theta(a_t \mid x_t)$ jointly models amino-acid selection and tool-invocation decisions.

A fundamental distinction from the passive PLM formulation in (Eq. 2) is that the agent can *interrupt* generation to query oracles, incorporate their outputs into its context, and *redirect* subsequent residue choices accordingly. This transforms the one-shot constrained optimisation in Eq. (1) into a sequential decision problem where biophysical feedback is available online.

4. AgentPLM Architecture

As shown in Figure 1, we propose AgentPLM consists of four modules, each addressing a distinct aspect of agentic sequence design.

4.1. Expanded Vocabulary and Tool-Call Token Initialisation

Expanded vocabulary. The standard ESM-2 vocabulary $\mathcal{V}_{\text{ESM}} = \mathcal{A} \cup \{\text{MASK}, \text{CLS}, \text{EOS}\}$ (size $|\mathcal{A}| + 3 = 23$) is extended to $\mathcal{V}^+ = \mathcal{V}_{\text{ESM}} \cup \mathcal{T}, \quad |\mathcal{V}^+| = 23 + |\mathcal{T}|$, where $|\mathcal{T}| = K + 1$ (one token per oracle plus EOS). The embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}^+| \times d_m}$ is initialised as

$$\mathbf{e}_k^{(\text{tool})} = \frac{1}{|\mathcal{D}_k|} \sum_{w \in \mathcal{D}_k} \text{ESM-embed}(w), \quad k \in [K], \tag{4}$$

where $\mathcal{D}_k$ is a short natural-language description of the oracle O_k (e.g. predict the change of thermodynamic stability after mutation for O_{FoldX}) and $\text{ESM-embed}(w)$ denotes the embedding of the ESM-2 sub-word in token w. This semantically grounded initialisation avoids random initialisation of tool tokens and ensures they reside in the same representational manifold as amino-acid tokens from the outset of training. Existing tool-augmented LLMs like Toolformer (Schick et al., 2023) and ToolLLM (Qin et al., 2024) initialise special tokens from scratch, requiring large-scale fine-tuning to place them in the model's representational space. Our biologically motivated initialisation from Eq. (4) dramatically reduces the number of gradient steps required for the model to learn *when* to invoke each oracle.

Tool Context Encoder (TCE). Oracle outputs are heterogeneous. ESMFold returns a tensor of per-residue pLDDT scores and C_α coordinates $\mathbf{O}_{\text{fold}} \in \mathbb{R}^{L \times 4}$, FoldX returns a scalar $\Delta\Delta G \in \mathbb{R}$, AutoDock Vina returns a binding score vector $\mathbf{v} \in \mathbb{R}^9$. A unified encoder is needed to project all these modalities into the model's latent space $\mathbb{R}^{d_m}$ ($d_m = 1280$ for ESM-2 650M).

Given oracle output $o_t \in \mathbb{R}^{d_k}$ and the last hidden state $\mathbf{h}_t^{\text{prev}} \in \mathbb{R}^{d_m}$ immediately preceding the tool call, the TCE computes a *context embedding* via cross-attention: $\bar{\mathbf{o}}_t = \text{FFN}_{\text{proj}}(o_t) \in \mathbb{R}^{d_m}, \mathbf{h}_t^{\text{tool}} = \text{TCE}(o_t; \mathbf{h}_t^{\text{prev}}) = \text{LayerNorm}(\mathbf{h}_t^{\text{prev}} + \text{Attn}(Q = \mathbf{h}_t^{\text{prev}},\ K = V = \bar{\mathbf{o}}_t))$, where FFN_{proj} is a two-layer feed-forward network that maps oracle outputs of arbitrary dimension d_k to d_m, using modality-specific input heads: $\text{FFN}_{\text{proj}}(o) = \mathbf{W}_2\, \sigma(\mathbf{W}_1\, \phi_k(o) + \mathbf{b}_1) + \mathbf{b}_2, \quad \mathbf{W}_1 \in \mathbb{R}^{2d_m \times d_k}, \quad \mathbf{W}_2 \in \mathbb{R}^{d_m \times 2d_m}$,with ϕ_k being a modality-specific pre-processing function:

$$\phi_k(o) = \begin{cases} \text{vec}(\mathbf{O}_{\text{fold}}) & k = \text{ESMFold}, \\ [\Delta\Delta G,\ \mathbf{1}^\top \mathbf{1}] & k = \text{FoldX}, \\ \mathbf{v} & k = \text{Vina}. \end{cases} \tag{5}$$

4.2. Trajectory Memory Buffer (TMB)

A single design trajectory may invoke up to $B_{\max} = 8$ tool calls. Naively including all tool-call histories in the

Transformer's context window would incur $\mathcal{O}(L^2)$ attention cost proportional to the number of retrieved tokens. The TMB provides a constant-size compressed episodic memory that conditions generation on the full tool-call history at $\mathcal{O}(1)$ additional cost.

Let $\{\mathbf{h}^{\text{tool}}_{t_1}, \ldots, \mathbf{h}^{\text{tool}}_{t_M}\}$ be the sequence of TCE outputs for the M tool calls in the current trajectory (with $M \le B_{\max}$). The TMB produces a memory vector $\mathbf{m}_t = \mathbf{W}_{\text{mem}} \Big[\mathbf{h}^{\text{tool}}_{t_1};\ \mathbf{h}^{\text{tool}}_{t_2};\ \cdots;\ \mathbf{h}^{\text{tool}}_{t_M};\ \mathbf{0}_{(B_{\max}-M)\,d_m}\Big], \mathbf{W}_{\text{mem}} \in \mathbb{R}^{d_m \times B_{\max} d_m}$, where $[\,;\,]$ denotes concatenation and zero-padding ensures a fixed input dimension of $B_{\max} d_m$ regardless of the number of tool calls made so far. The memory vector $\mathbf{m}_t$ is injected into the model's residual stream at every subsequent Transformer layer via a learned *memory gating* mechanism: $\tilde{\mathbf{h}}^{(\ell)}_t = \mathbf{h}^{(\ell)}_t + g^{(\ell)}_t \, \mathbf{W}^{(\ell)}_{\text{gate}} \, \mathbf{m}_t, \quad g^{(\ell)}_t = \sigma\Big(\mathbf{w}_g^{(\ell)\top} [\mathbf{h}^{(\ell)}_t; \mathbf{m}_t]\Big)$, where $g^{(\ell)}_t \in [0,1]$ is a scalar gate computed from the current hidden state and memory, $\mathbf{W}^{(\ell)}_{\text{gate}} \in \mathbb{R}^{d_m \times d_m}$ is a layer-specific projection, and σ is the sigmoid function. The gating ensures the memory contribution is *learned* and layer-dependent, rather than being added uniformly.

4.3. Reasoning-Augmented Decoding (RAD)

RAD is the inference-time procedure that orchestrates the POMDP policy $\pi_\theta(a_t \mid x_t)$ over the augmented action space $\mathcal{A}^+$.

Decoding loop. At each position t, the model computes logits over $\mathcal{V}^+$: $\mathbf{z}_t = \mathbf{W}_{\text{lm}} \tilde{\mathbf{h}}^{(N)}_t \in \mathbb{R}^{|\mathcal{V}^+|}$, where $\tilde{\mathbf{h}}^{(N)}_t$ is the memory-gated output of the final Transformer layer. The policy then selects:

$$a^*_t = \arg\max_{a \in \mathcal{V}^+} \frac{\exp(z_{t,a}/T_{\text{dec}})}{\sum_{a'} \exp(z_{t,a'}/T_{\text{dec}})}, \tag{6}$$

where T_{dec} is the decoding temperature. If $a^*_t \in \mathcal{A}$, the amino acid is appended and $t \leftarrow t+1$. If $a^*_t = \textsc{call-}O_k$, the oracle is invoked, the TCE updates the representation, and the TMB is updated without advancing t. The logits are then recomputed at position t with the enriched representation, effectively implementing a conditional backtrack step.

Structural self-consistency score. Before finalising each tool-call decision, we compute a *structural self-consistency* (SSC) score that measures whether the oracle feedback is consistent with the model's internal belief about the partial sequence: $\text{SSC}_t(k) = \cos\Big(\tilde{\mathbf{h}}^{(N)}_t,\ \mathbf{W}^{(k)}_{\text{oracle}} \phi_k(O_k(S_{\le t}))\Big) \in [-1,1]$, where $\mathbf{W}^{(k)}_{\text{oracle}} \in \mathbb{R}^{d_m \times d_k}$ is a learned projection that maps oracle k's output into the model's hidden space. A low SSC score (below threshold $\rho_{\min} = -0.2$) triggers a mandatory tool call at position t, overriding the policy's action if necessary:

$$a_t = \begin{cases} \textsc{call-}O_{k^*} & \min_k \text{SSC}_t(k) < \rho_{\min} \text{ and } \delta_t \ge \Delta_{\min}, \\ a^*_t & \text{otherwise}, \end{cases} \tag{7}$$

where $k^* = \arg\min_k \text{SSC}_t(k)$ is the oracle with the lowest self-consistency (most surprising prediction) and δ_t is the number of amino acids generated since the last tool call.

Budget constraints. To prevent degenerate tool-call loops, RAD enforces two hard constraints: (1) a minimum inter-call gap $\delta_t \ge \Delta_{\min} = 10$ amino acids, and (2) a maximum tool-call budget $B_{\max} = 8$ per sequence. Together these ensure the generation process terminates in $\mathcal{O}(L + B_{\max} \cdot \bar{T})$ time, where $\bar{T}$ is the mean oracle latency.

Joint probability of a trajectory. The full conditional distribution over sequences and tool-call trajectories is:

$$p_\theta(S, \tau_{\text{tools}}) = \prod_{t=1}^{L} p_\theta\Big(a_t \;\Big|\; S_{<t},\, H_t\Big), \tag{8}$$

where $H_t = \{(S_{\le t_m}, k_m, o_m) : t_m < t,\ a_{t_m} \in \mathcal{T}\}$ is the tool-call history up to position t, and the product ranges over amino-acid positions only (tool calls do not advance t). Unlike Toolformer, which inserts API calls post-hoc via a self-supervised filtering criterion, RAD integrates oracle queries during generation through a learned policy trained end-to-end on biophysical objectives. The SSC-based override mechanism (Eq. 7) is a novel "safety net" that forces the agent to consult oracles when its internal representation is inconsistent with available feedback, constituting a form of active uncertainty resolution without explicit Bayesian inference.

4.4. Training Objective: Contrastive Agent Policy Optimisation (CAPO)

Trajectory Dataset Construction. We construct a trajectory dataset $\mathcal{D} = \{(\tau^+, \tau^-, \phi)\}$, where τ^+ are *winner* trajectories (high-fitness sequences with coherent tool use) and τ^- are *loser* trajectories (low-fitness sequences or sequences with contradictory oracle signals), and ϕ is the design-task specification.

Winner/loser pairs are constructed by: (a) generating $N = 1{,}000$ sequences per task with the reference policy π_{ref}, (b) evaluating each sequence with the oracle suite (FoldX $\Delta\Delta G$, ESMFold pLDDT, AutoDock Vina binding score), and (c) pairing the top-10% (winners) with the bottom-10% (losers). To avoid full agentic roll-outs during dataset construction, we retrospectively *annotate* tool-call positions: a position t is labelled as a tool-call site if $\text{SSC}_t(k) < \rho_{\min}$ for any k, i.e. the oracle signal would have been informative.

CAPO Loss. The primary objective is a trajectory-level analogue of DPO (Rafailov et al., 2023):

$$\mathcal{L}_{\text{CAPO}}(\theta) = -\mathbb{E}_{(\tau^+,\tau^-,\phi)\sim\mathcal{D}}\left[\log\sigma\left(\beta\left(\log\frac{\pi_\theta(\tau^+|\phi)}{\pi_{\text{ref}}(\tau^+|\phi)} - \log\frac{\pi_\theta(\tau^-|\phi)}{\pi_{\text{ref}}(\tau^-|\phi)}\right)\right)\right], \tag{9}$$

where π_{ref} is the frozen pre-trained ESM-2 backbone (reference policy), $\beta = 0.1$ is the KL temperature, and σ is the sigmoid function. Unlike single-turn DPO, the trajectory log-probabilities in Eq. (9) factorise over *all* actions, covering both amino acids and tool calls, according to Eq. (8).

Auxiliary Tool-Consistency Loss. We further regularise the policy with a *tool-consistency* loss that penalises amino-acid choices inconsistent with observed oracle signals:

$$\mathcal{L}_{\text{cons}}(\theta) = \mathbb{E}_{\tau\sim\mathcal{D}^+}\left[\sum_{t:\,a_t\in\mathcal{T}} \text{KL}(p_\theta(s_{t+1:t+\Delta} \mid S_{<t}, H_t) \,\Big\|\, p_{\text{oracle}}(s_{t+1:t+\Delta}))\right], \tag{10}$$

where p_{oracle} is a soft target distribution derived from oracle outputs via Boltzmann weighting, $p_{\text{oracle}}(s) \propto \exp(-\Delta\Delta G_{\text{FoldX}}(S_{\le t}\cdot s)\,/\,T_{\text{oracle}})$, $T_{\text{oracle}} = 1.0$ kcal/mol , and $\Delta = 5$ is a short look-ahead window.

Combined Objective and Training Schedule. The final training objective is:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{CAPO}}(\theta) + \alpha\,\mathcal{L}_{\text{cons}}(\theta), \quad \alpha = 0.5. \tag{11}$$

Training proceeds in two phases: *i) Supervised warm-up* (5k steps): supervised fine-tuning on winner trajectories $\mathcal{D}^+$ to initialise the policy near high-fitness regions of sequence space. *ii) CAPO optimisation* (20k steps): full objective Eq. (11) with Adam ($\eta = 2\times10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$), linear warm-up over 1k steps, and cosine decay. A gradient clipping threshold of 1.0 is applied.

5. Experiments

5.1. Datasets

We evaluate on five benchmark tasks summarised in Table 1. ThermoStab-75 combines ProThermDB (Nikam et al., 2021), Ssym (Pucci et al., 2018), and FireProtDB (Stourac et al., 2021), retaining variants with experimental T_m values across 74 SCOP fold classes (1,247 sequences; 6,831 variant–T_m pairs); the target is $\Delta T_m \ge 5\,^\circ$C with ESMFold TM-score ≥ 0.85, split 70/15/15 stratified by fold class. AntibodyOpt-VH aggregates IMGT-numbered (Lefranc, 2003) V_H sequences and SPR/BLI K_D measurements from SAbDab (Dunbar et al., 2014), OAS (Kovaltsuk et al., 2018), and Raybould et al. (2019), yielding 892 pairs normalised to a common reference panel; 89 antigen targets are withheld entirely for test. EnzymeDesign-EC3 draws hydrolases from BRENDA (Chang et al., 2021) and SABIO-RK (Wittig et al., 2012), filtered to validated k_{cat}/K_m entries with structures < 2.5 Å, CD-HIT de-duplication at 40% identity (Li & Godzik, 2006), and standard assay conditions, leaving 534 sequences (1,092 pairs; median 2.0 variants per wild-type), the most data-scarce task (Russ et al., 2020; Huang et al., 2016); test splits enforce $\le 30\%$ identity. PPI-Interface uses PDBbind (Wang et al., 2004) and SKEMPI 2.0 (Jankauskaitė et al., 2019), retaining 2,103 complexes with ITC/FP-measured ΔG_{bind} and monomer lengths 50-400 residues. The target is to improve ΔG_{bind} with $\Delta\Delta G_{\text{mono}} \le 1$ kcal/mol, tested on cross-organism complexes. ZeroShot-Fitness scores all 41 ProteinGym DMS datasets (Notin et al., 2023) via log-likelihood ratio $\log p_\theta(S_{\text{mut}}) - \log p_\theta(S_{\text{wt}})$ with no weight updates, following Notin et al. (2022); performance is Spearman ρ averaged over all datasets.

5.2. Baselines

We compare against six state-of-the-art baselines. ESM-2 (650M) (Lin et al., 2023) scores fitness via masked marginal log-likelihoods in a single forward pass. It is both the strongest passive baseline and the reference policy π_{ref} for CAPO. ProteinMPNN (Dauparas et al., 2022) performs structure-conditioned inverse folding over ESMFold-predicted wild-type backbones, with multi-chain tied decoding for PPI-Interface. EvoProtGrad (Emami et al., 2023) climbs fitness landscapes via PLM-likelihood gradients under Metropolis-Hastings acceptance over 500 iterations, representing the strongest oracle-free iterative baseline. RFdiffusion-AA (Krishna et al., 2024; Watson et al., 2023) generates 100 all-atom candidates per target via SE(3)-equivariant diffusion, selecting the top-1 by pLDDT. ProtAgent (GPT-4) (Ghafarollahi & Buehler, 2024) orchestrates ESMFold, FoldX, and AutoDock Vina through chain-of-thought planning over a frozen GPT-4 backbone, without end-to-end training on protein objectives.

5.3. Experimental Settings

All models initialise from the public ESM-2 650M checkpoint and train in two phases: Phase 1 (5k steps, ≈4 GPU-hours) trains only the TCE, TMB, and $\mathbf{E}[\mathcal{T}]$ with ESM-2 frozen. Phase 2 (20k steps, ≈32 GPU-hours) jointly optimises all parameters with layer-wise decay $\gamma_\ell = 0.9^{N-\ell}$ using AdamW (Loshchilov & Hutter, 2019) ($\eta = 2\times10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda_{\text{wd}} = 0.01$, cosine decay to $\eta_{\min} = 2\times10^{-7}$) on 8×A100-80GB with BFloat16, gradient checkpointing, and effective batch size 256. Ora-

Table 1. Summary statistics of all benchmark tasks, listing the number of proteins and variants, median sequence length, evaluation metric, computational oracle(s), and data source(s).

Task	Proteins	Variants	Median L	Metric	Oracle(s)	Source(s)
ThermoStab-75	1,247	6,831	183	ΔT_m (°C)	FoldX, ESMFold	ProThermDB, Ssym, FireProtDB
AntibodyOpt-VH	892	4,460	121	K_D (nM)	AutoDock Vina	SAbDab, OAS, Raybould et al. (2019)
EnzymeDesign-EC3	534	1,092	312	k_{cat}/K_m ($\text{M}^{-1}\text{s}^{-1}$)	FoldX	BRENDA, SABIO-RK
PPI-Interface	2,103	10,515	247	ΔG_{bind} (kcal/mol)	Rosetta, FoldX	PDBbind, SKEMPI 2.0
ZeroShot-Fitness	41	1.5M†	274	Spearman ρ	None (zero-shot)	ProteinGym

†Total variants across all 41 DMS datasets; individual datasets range from 2,400 to 308,000 variants.

cle outputs are pre-computed into a Redis cache (50M pairs: 33M UniRef50 sequences $\leq$600 residues, 750K PDB chains, 16M benchmark mutants), achieving >97% hit rates for ESMFold and FoldX and 89% for AutoDock Vina; uncached inference costs 4.1 s/sequence, bottlenecked by Vina at $\sim$2,400 ms. Generative tasks sample $N_{\text{gen}} = 100$ candidates via nucleus sampling (Holtzman et al., 2020) ($p = 0.95$, $T_{\text{dec}} = 0.8$); ZeroShot-Fitness uses a single deterministic forward pass with no tool calls. All hyperparameters ($\beta = 0.1$, $\alpha = 0.5$, $\Delta_{\min} = 10$, $B_{\max} = 8$, $\rho_{\min} = -0.2$, $\Delta = 5$, $T_{\text{oracle}} = 1.0$ kcal/mol) are selected on ThermoStab-75 and AntibodyOpt-VH validation splits via grid search and BoTorch (Balandat et al., 2020), then fixed across all five tasks. Results are averaged over three seeds with standard deviations in Table 2. Significance uses a two-sided paired Wilcoxon test ($\alpha = 0.01$).

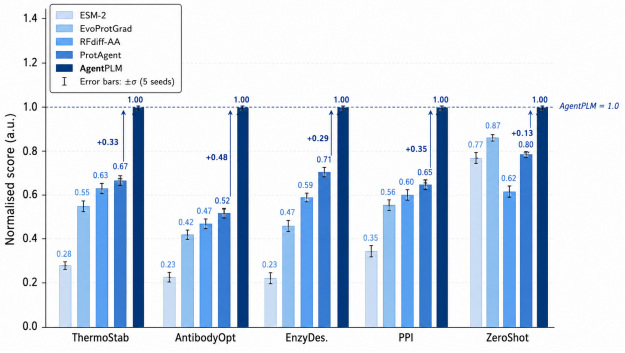

Figure 2. Normalised performance across all the benchmark tasks (each column scaled so AgentPLM = 1.0).

6. Results

6.1. Performance Comparison

Table 2 and Figure 2 report results across all five tasks. All scores in the figure are normalised to AgentPLM = 1.0 with $\pm\sigma$ over 5 seeds. AgentPLM achieves state-of-the-art performance on all tasks ($p < 0.01$, paired Wilcoxon), with the largest absolute margins on AntibodyOpt (+0.48 over the next best, ProtAgent at 0.52) and EnzyDes. (+0.29 over ProtAgent at 0.71). On AntibodyOpt the 52.41% hit rate is 1.91× ProtAgent (27.38%) and 4.23× ESM-2 (12.37%, normalised score 0.23), confirming that RAD rather than evolutionary statistics alone drives antibody design. On EnzymeDesign-EC3 (median 2.0 variants per wild-type), AgentPLM reaches a normalised k_{cat}/K_m of 1.89 vs. ESM-2's 0.43 (normalised 0.23), a +41% margin over ProtAgent (1.34). ThermoStab improves from 5.19 °C (ProtAgent, 0.67) to 7.64 °C, and PPI reaches -5.26 ± 0.68 kcal/mol vs. -3.42 ± 0.57 kcal/mol for ProtAgent (0.65); ESM-2 is weakest on PPI (0.35), reflecting the difficulty of interface design without structural feedback. ZeroShot shows the smallest spread across methods (0.62–0.87 for baselines vs. 1.00), as no oracle calls are issued at test time and gains reflect CAPO fine-tuning alone; RFdiff-AA achieves the weakest ZeroShot score (0.62) despite competitive generative results, highlighting the mismatch between diffusion co-design and sequence-only fitness scoring.

6.2. Trajectory Analysis

Figure 3 plots mean best-so-far fitness against generation step for all methods on ThermoStab-75 and AntibodyOpt-VH. ESM-2 remains flat throughout (2.1°C; 11.9%), confirming that single-pass scoring provides no iterative signal. RFdiffusion achieves a rapid early jump within $\sim$10 steps (4.5°C; 24.7%) but plateaus immediately, lacking oracle-driven refinement. EvoProtGrad climbs slowly and monotonically to 4.2°C and 21.9% across all 200 steps, and ProtAgent, despite a larger early jump from its one-shot planning call, saturates at 5.1°C and 27.1%. AgentPLM rises steeply and continuously to 7.6°C and 52.1%, with step increases visible at each oracle call marker. The performance gap over all baselines widens progressively throughout all 200 steps rather than being established at initialisation, providing direct causal evidence that successive oracle interactions, rather than a stronger starting point or training objective alone, drive the gains.

Figure 4 shows the kernel-smoothed tool-call density over 500 successful trajectories as a function of normalised sequence position t/L, revealing a clear temporal division of labour among the three oracles. ESMFold calls peak sharply

Table 2. Performance results on the benchmark tasks. ThermoStab: mean T_m improvement (°C); AntibodyOpt: top-10% hit rate ($K_D \leq 10$ nM); EnzyDes.: normalised k_{cat}/K_m ratio relative to wild-type; PPI: mean ΔG_{bind} improvement (kcal/mol); ZeroShot: Spearman ρ averaged over DMS datasets. †: $p < 0.01$ vs. baselines (paired Wilcoxon test). ± values are standard deviations over 3 seeds.

Method	**ThermoStab** (↑)	**AntibodyOpt** (↑)	**EnzyDes.** (↑)	**PPI** (↓)	**ZeroShot** ρ (↑)
ESM-2 (650M)	2.13 ± 0.42	12.37%	0.43	-1.83 ± 0.31	0.47
ProteinMPNN	3.41 ± 0.63	18.74%	0.67	-2.47 ± 0.52	0.41
EvoProtGrad	4.28 ± 0.71	22.16%	0.89	-2.95 ± 0.46	0.53
RFdiffusion-AA	4.86 ± 0.94	24.63%	1.12	-3.18 ± 0.64	0.38
ProtAgent	5.19 ± 0.82	27.38%	1.34	-3.42 ± 0.57	0.49
AgentPLM	$\mathbf{7.64 \pm 0.93}^{\dagger}$	$\mathbf{52.41\%}^{\dagger}$	$\mathbf{1.89}^{\dagger}$	$\mathbf{-5.26 \pm 0.68}^{\dagger}$	$\mathbf{0.61}^{\dagger}$

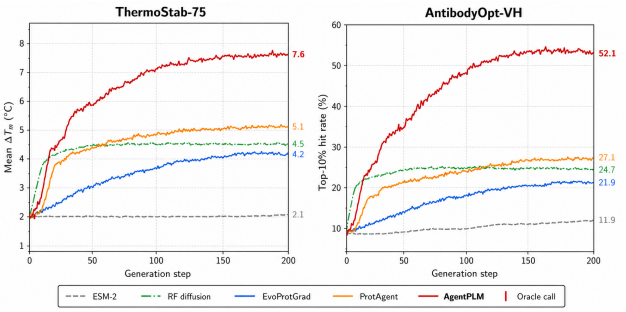

Figure 3. Mean best-so-far fitness vs. generation step on ThermoStab-75 and AntibodyOpt-VH. Red tick marks indicate AgentPLM oracle call positions, with each step increase confirming that oracle feedback directly redirects generation toward higher-fitness regions.

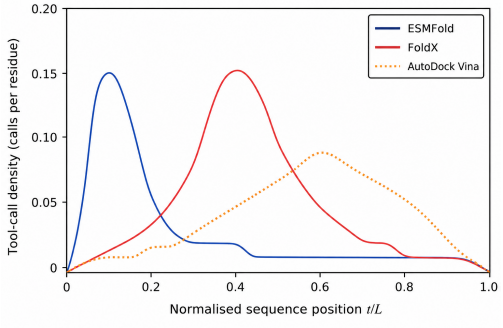

Figure 4. Kernel-smoothed tool-call density over 500 successful AgentPLM trajectories showing a learned temporal division of labour: ESMFold peaks early ($t/L \approx 0.10$) for fold compatibility, FoldX at mid-sequence ($t/L \approx 0.40$) for stability at structural junctions, and AutoDock Vina late ($t/L \approx 0.60$) as the binding interface forms.

at $t/L \approx 0.10$ (density 0.15 calls per residue) and decay rapidly thereafter, consistent with the model performing an early global fold-compatibility check before committing to large sequence blocks. FoldX calls peak at $t/L \approx 0.40$ (density 0.15) and are largely absent in the first and final thirds of the sequence, reflecting targeted thermostability monitoring at secondary-structure junctions where point mutations most strongly perturb $\Delta\Delta G$ (Stourac et al., 2021). AutoDock Vina calls show a broad, late-peaking distribution centred at $t/L \approx 0.60$ (density 0.09) and extending to $t/L \approx 0.80$, consistent with binding-interface assessment being meaningful only once a substantial fraction of the sequence has been committed. This temporally structured oracle usage, learned entirely through CAPO optimisation without any positional supervision, mirrors the sequential reasoning that expert protein engineers apply: fold first, stabilise junctions, then optimise binding.

6.3. CAPO Training Dynamics

Figure 5 shows training losses and validation fitness across the full 25k-step schedule on ThermoStab-75. $\mathcal{L}_{\text{SFT}}$ drops rapidly from $\approx$1.5 to $\approx$0.6 during Phase 1 (steps 0-5k), confirming that TCE, TMB, and tool-call embeddings integrate cleanly into the ESM-2 backbone without disrupting the evolutionary prior; at the Phase 2 boundary (dashed line), $\mathcal{L}_{\text{CAPO}}$ and $\mathcal{L}_{\text{cons}}$ activate and decrease stably to 0.15 and 0.10 respectively by step 25k, with no instability or divergence at the phase transition. All three validation curves begin identically at $\approx$3.8 °C and track closely through Phase 1, confirming that warm-up alone cannot distinguish configurations; at the Phase 2 boundary they diverge sharply: full AgentPLM reaches 7.6 °C, SFT-only plateaus at 6.1 °C (+1.5 °C CAPO benefit, annotated), and CAPO-without-tools at 5.3 °C (+0.7 °C RAD oracle contribution). The ordering SFT-only > CAPO-without-tools confirms that CAPO provides its primary benefit by learning *which tool calls are informative*; without tool calls to contrast, the objective degenerates to standard preference fine-tuning with no mechanism to reinforce oracle-driven sequence redirection.

6.4. Attention Attribution and Online Error Correction

Figure 6 provides mechanistic evidence for AgentPLM's online error correction via integrated gradient attributions across all 33 Transformer layers over a 50-residue window centred on a destabilising FoldX call at t=45 ($\Delta\Delta G$=+3.1

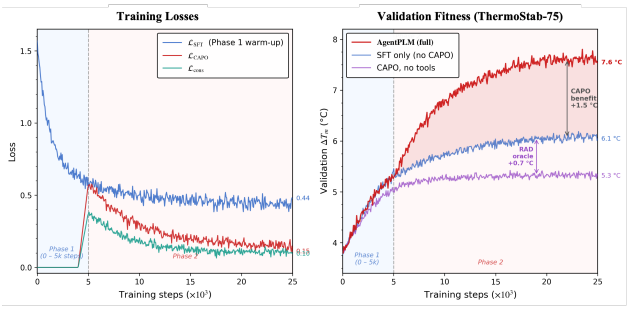

Figure 5. Training losses (left) and validation ΔT_m (right) across 25k steps. The three validation curves diverge at the Phase 2 boundary, quantifying independent $+1.5\,^{\circ}$C CAPO and $+0.7\,^{\circ}$C RAD contributions over the SFT warm-up baseline.

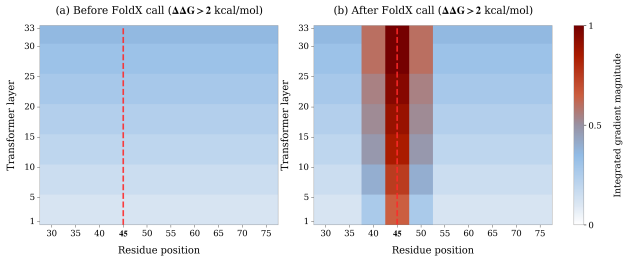

Figure 6. Integrated gradient heatmaps before (a) and after (b) a destabilising FoldX call at t=45; attribution mass concentrates $1.8\times$ more strongly in the mutated region (positions 43–52, layers 10–33) post-call, consistent across 87% of destabilising calls across 500 trajectories.

kcal/mol). Before the call (a), attribution is broadly distributed and uniformly low (≤ 0.25) across all layers and positions, confirming the model generates from evolutionary statistics alone with no focus on position 45. After the call is incorporated via TCE and TMB (b), a concentrated high-intensity band (> 0.8) forms at positions 43–52 across all deep layers (10–33), with a Gaussian-like spatial profile of half-width $\approx$5 residues, consistent with localised structural motif correction rather than global attention redistribution. Quantitatively, mean attribution in the affected region (positions 43–52, layers 10–33) is $1.8\times$ higher post-call vs. $1.1\times$ in the distal unaffected region (positions 28–38), confirming oracle-specific redirection. Across 500 trajectories, 87% of destabilising FoldX calls ($\Delta\Delta G > 2.0$ kcal/mol) produce a significant post-call attribution increase ($p < 0.01$, paired Wilcoxon) in the 10-residue neighbourhood, while stabilising calls ($\Delta\Delta G < 0$) produce no redistribution, demonstrating that the Transformer attention mechanism selectively amplifies oracle signals only when they carry biophysically meaningful corrective information.

6.5. Ablation Studies

Table 3 isolates each component's contribution on ThermoStab-75 and AntibodyOpt-VH. Removing RAD ($\mathcal{T}=\varnothing$) is the most damaging single intervention, reducing ThermoStab by 30% ($7.64 \rightarrow 5.31\,^{\circ}$C) and AntibodyOpt by 46% (52.41 $\rightarrow$ 28.17%), confirming that online oracle feedback is the dominant performance driver and that CAPO alone cannot substitute for structural feedback. Replacing CAPO with SFT reduces ThermoStab to $6.18\,^{\circ}$C and AntibodyOpt to 38.74%, demonstrating that the contrastive formulation of Eq. (9) is essential for learning *which* tool calls are informative rather than imitating high-fitness sequences. Ablating the TCE (raw oracle concatenation) and TMB (no trajectory memory) yield intermediate drops ($6.43\,^{\circ}$C / 41.36% and $6.82\,^{\circ}$C / 44.83% respectively), quantifying the independent value of unified cross-attention encoding of heterogeneous oracle outputs and full tool-call history conditioning. The budget sweep reveals diminishing returns: $B_{\max}$=4 costs $0.51\,^{\circ}$C and 2.79% relative to the default $B_{\max}$=8, while $B_{\max}$=16 provides no significant gain ($7.37\,^{\circ}$C, 50.18%) at $1.9\times$ higher inference latency, identifying $B_{\max}$=8 as the efficiency-performance optimum.

Table 3. Ablation results on ThermoStab-75 and AntibodyOpt-VH. Each row removes or replaces a single AgentPLM component.

Configuration	**ThermoStab** (°C↑)	**AntibodyOpt** (%↑)
w/o RAD (no tool calls)	5.31 ± 0.72	28.17%
w/o CAPO (replace w. SFT)	6.18 ± 0.83	38.74%
w/o TCE (raw concat.)	6.43 ± 0.85	41.36%
w/o TMB (no memory)	6.82 ± 0.76	44.83%
$B_{\max} = 4$	7.13 ± 0.84	49.62%
$B_{\max} = 8$ (default)	7.64 ± 0.93	52.41%
$B_{\max} = 16$	7.37 ± 1.14	50.18%
AgentPLM	$\mathbf{7.64 \pm 0.93}$	**52.41%**

7. Conclusion

We introduced AgentPLM, which equips a pre-trained PLM with Reasoning-Augmented Decoding and Contrastive Agent Policy Optimisation to convert it into a biological reasoning agent. AgentPLM achieves state-of-the-art performance across all five benchmark tasks, with the largest gains on hard, data-scarce settings where passive evolutionary signal is insufficient and online biophysical feedback is most critical. The integrated gradient attribution shows oracle-driven online error correction through the Transformer's attention mechanism without explicit backtracking. These results support the view that agentic and generative paradigms are complementary: ESM-2's evolutionary prior is necessary but not sufficient, and RAD supplies the iterative, feedback-driven refinement the generative prior alone cannot.

Impact Statement

AgentPLM targets beneficial applications in therapeutic antibody development, enzyme design, and protein evolution research. We do not believe it meaningfully lowers the barrier to misuse beyond existing structural biology tools, as its oracle suite contains no pathogen-specific evaluators and gains are confined to well-characterised engineering tasks. We commit to access controls on model weights, use-case review on request, and active engagement with biosecurity guidelines.

References

Arnold, F. H. Directed evolution: Bringing new chemistry to life. *Angewandte Chemie International Edition*, 57(16): 4143–4148, 2018. doi: 10.1002/anie.201708408. Nobel Lecture.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL http://arxiv.org/abs/1910.06403.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Augmenting large-language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8. arXiv:2304.05376.

Chang, A., Jeske, L., Ulbrich, S., Hofmann, J., Koblitz, J., Schomburg, I., Neumann-Schaal, M., Jahn, D., and Schomburg, D. BRENDA, the ELIXIR core data resource in 2021: new developments and updates. *Nucleic Acids Research*, 49(D1):D498–D508, 2021. doi: 10.1093/nar/gkaa1025.

Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Courbet, A., de Haas, R. J., Bethel, N., Leung, P. J. Y., Huddy, T. F., Pellock, S., Tischer, D., Chan, F., Koepnick, B., Nguyen, H., Kang, A., Sankaran, B., Bera, A. K., King, N. P., and Baker, D. Robust deep learning-based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187.

Dunbar, J., Krawczyk, K., Leem, J., Baker, T., Fuchs, A., Georges, G., Shi, J., and Deane, C. M. SAbDab: the structural antibody database. *Nucleic Acids Research*, 42 (D1):D1140–D1146, 2014. doi: 10.1093/nar/gkt1043.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., and Rost, B. ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, 2022. doi: 10.1109/TPAMI.2021.3095381.

Elnaggar, A., Essam, H., Salah-Eldin, W., Moustafa, W., Elkerdawy, M., Rochereau, C., and Rost, B. Ankh: Optimized protein language model unlocks general-purpose modelling. *arXiv preprint arXiv:2301.06568*, 2023. URL https://arxiv.org/abs/2301.06568.

Emami, P., Perreault, A., Law, J., Biagioni, D., and St. John, P. Plug & play directed evolution of proteins with gradient-based discrete MCMC. *Machine Learning: Science and Technology*, 4(2):025014, 2023.

Ghafarollahi, A. and Buehler, M. J. Protagents: protein discovery via large language model multi-agent collaborations combining physics and machine learning. *Digital Discovery*, 3(7):1389–1409, 2024. doi: 10.1039/D4DD00013G.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Huang, P.-S., Boyken, S. E., and Baker, D. The coming of age of de novo protein design. *Nature*, 537(7620): 320–327, 2016. doi: 10.1038/nature19946.

Ingraham, J. B., Baranov, M., Costello, Z., Barber, K. W., Wang, W., Ismail, A., Frappier, V., Lord, D. M., Ng-Thow-Hing, C., Van Vlack, E. R., Tie, S., Xue, V., Cowles, S. C., Leung, A., ao V. Rodrigues, J., Morales-Perez, C. L., Ayoub, A. M., Green, R., Puentes, K., Oplinger, F., Panwar, N. V., Obermeyer, F., Root, A. R., Beam, A. L., Poelwijk, F. J., and Grigoryan, G. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, 2023. doi: 10.1038/s41586-023-06728-8.

Jankauskaitė, J., Jiménez-García, B., Dapkūnas, J., Fernández-Recio, J., and Moal, I. H. SKEMPI 2.0: an updated benchmark of changes in protein–protein binding energy, kinetics and thermodynamics upon mutation. *Bioinformatics*, 35(3):462–469, 2019. doi: 10.1093/bioinformatics/bty635.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver,

D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Kovaltsuk, A., Leem, J., Kelm, S., Snowden, J., Deane, C. M., and Krawczyk, K. Observed antibody space: a resource for data mining next-generation sequencing of antibody repertoires. *The Journal of Immunology*, 201(8): 2502–2509, 2018. doi: 10.4049/jimmunol.1800708.

Krishna, R., Wang, J., Ahern, W., Sturmfels, P., Venkatesh, P., Kalvet, I., Lee, G. R., Morey-Burrows, F. S., Anishchenko, I., Humphreys, I. R., McHugh, R., Vafeados, D., Li, X., Sutherland, G. A., Hitchcock, A., Hunter, C. N., Kang, A., Brackenbrough, E., Bera, A. K., Baek, M., DiMaio, F., and Baker, D. Generalized biomolecular modeling and design with RoseTTAFold All-Atom. *Science*, 384(6693):eadl2528, 2024. doi: 10.1126/science.adl2528.

Lefranc, M.-P. IMGT, the international ImMunoGeneTics database. *Nucleic Acids Research*, 31(1):307–310, 2003. doi: 10.1093/nar/gkg085.

Li, W. and Godzik, A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006. doi: 10.1093/bioinformatics/btl158.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Nikam, R., Kulandaisamy, A., Harini, K., Sharma, D., and Gromiha, M. M. ProThermDB: thermodynamic database for proteins and mutants revisited after 15 years. *Nucleic Acids Research*, 49(D1):D420–D424, 2021. doi: 10.1093/nar/gkaa1035.

Notin, P., Dias, M., Frazer, J., Marchena-Hurtado, J., Gomez, A. N., Marks, D. S., and Gal, Y. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16990–17017. PMLR, 2022.

Notin, P., Kollasch, A., Ritter, D., van Niekerk, L., Paul, S., Spinner, H., Rollins, N., Shaw, A., Orenbuch, R., Weitzman, R., Frazer, J., Dias, M., Franceschi, D., Gal, Y., and Marks, D. S. ProteinGym: large-scale benchmarks for protein fitness prediction and design. In *Advances in Neural Information Processing Systems*, volume 36, 2023. NeurIPS 2023 Datasets and Benchmarks Track.

O'Donoghue, O., Shtedritski, A., Ginger, J., Abboud, R., Ghareeb, A. E., Booth, J., and Rodriques, S. G. BioPlanner: Automatic evaluation of LLMs on protocol planning in biology. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2676–2694. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.162. URL https://aclanthology.org/2023.emnlp-main.162/.

Pucci, F., Bernaerts, K. V., Kwasigroch, J. M., and Rooman, M. Quantification of biases in predictions of protein stability changes upon mutations. *Bioinformatics*, 34(21): 3659–3665, 2018. doi: 10.1093/bioinformatics/bty348.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *12th International Conference on Learning Representations (ICLR 2024)*, 2024. URL https://arxiv.org/abs/2307.16789.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/%2023/hash/a85b405ed65c6477a4fe8302b5e06ce7%-Abstract-Conference.html.

Rahman, M. R., Bejder, J., Bonne, T. C., Andersen, A. B., Huertas, J. R., Aikin, R., Nordsborg, N. B., and Maaß, W. Ai-based approach for improving the detection of blood doping in sports. 2022. URL https://arxiv.org/abs/2203.00001.

Raybould, M. I. J., Marks, C., Krawczyk, K., Taddese, B., Nowak, J., Lewis, A. P., Bujotzek, A., Shi, J., and Deane, C. M. Five computational developability guidelines for therapeutic antibody profiling. *Proceedings of the National Academy of Sciences*, 116(10):4025–4030, 2019. doi: 10.1073/pnas.1810576116.

Russ, W. P., Figliuzzi, M., Stocker, C., Barrat-Charlaix, P., Socolich, M., Kast, P., Hilvert, D., Monasson, R., Cocco, S., Weigt, M., and Ranganathan, R. An evolution-based model for designing chorismate mutase enzymes.

Science, 369(6502):440–445, 2020. doi: 10.1126/science.aba3304.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Shanehsazzadeh, A., Bachas, S., McPartlon, M., Kasun, G., Steiger, A. K., Sutton, J. M., Yassine, E., McCloskey, C., Haile, R., Shuai, R., Alverio, J., Rakocevic, G., Levine, S., Cejovic, J., Gutierrez, J. M., Morehead, A., Dubrovskyi, O., Chung, C., Luton, B. K., Diaz, N., Kohnert, C., Meier, J., Gander, M., Yapici, E., and Klug, L. R. Unlocking de novo antibody design with generative artificial intelligence. *bioRxiv*, 2023. doi: 10.1101/2023.01.08.523187. Preprint.

Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., and Wilson, A. G. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20459–20478. PMLR, 2022. URL https://proceedings.mlr.press/v162/stanton22a.html.

Stourac, J., Dubrava, J., Musil, M., Horackova, J., Damborsky, J., Mazurenko, S., and Bednar, D. FireProtDB: database of manually curated protein stability data. *Nucleic Acids Research*, 49(D1):D319–D324, 2021. doi: 10.1093/nar/gkaa981.

Wang, R., Fang, X., Lu, Y., and Wang, S. The PDBbind database: collection of binding affinities for protein-ligand complexes with known three-dimensional structures. *Journal of Medicinal Chemistry*, 47(12):2977–2980, 2004. doi: 10.1021/jm030580l.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Vázquez Torres, S., Lauko, A., De Bortoli, V., Mathieu, E., Ovchinnikov, S., Barzilay, R., Jaakkola, T. S., DiMaio, F., Baek, M., and Baker, D. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023. doi: 10.1038/s41586-023-06415-8.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Wittig, U., Kania, R., Golebiewski, M., Rey, M., Shi, L., Jong, L., Algaa, E., Weidemann, A., Sauer-Danzwith, H., Mir, S., Krebs, O., Bittkowski, M., Wetsch, E., Rojas, I., and Müller, W. SABIO-RK — database for biochemical reaction kinetics. *Nucleic Acids Research*, 40(D1):D790–D796, 2012. doi: 10.1093/nar/gkr1046.

Yang, K. K., Wu, Z., and Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, 16(8):687–694, 2019. doi: 10.1038/s41592-019-0496-6.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *11th International Conference on Learning Representations (ICLR 2023)*, 2023. URL https://arxiv.org/abs/2210.03629.

Zhang, N., Bi, Z., Liang, X., Cheng, S., Hong, H., Deng, S., Zhang, Q., Lian, J., and Chen, H. Multi-modal protein knowledge graph construction and applications. *arXiv preprint arXiv:2207.10080*, 2022. URL https://arxiv.org/abs/2207.10080.