RNAOMNI: AN AUTONOMOUS MODELING AGENT FOR RNA DOWNSTREAM TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

As a foundational molecule of life, RNA property prediction and functional identification represent a vital frontier at the intersection of AI and biology. However, developing high-performance models for downstream RNA tasks still relies heavily on labor-intensive iterations of model design, validation, and refinement. Given the growing demand for task-specific modeling, domain experts possess rich data but lack engineering bandwidth, while AI researchers have strong modeling capabilities yet struggle to address the diverse domain-specific needs. Existing AutoML methods partially alleviate this burden but remain restricted to hyperparameter tuning within fixed search spaces, and LLM-based agents depend on human-provided blueprints, neither of which achieves full end-to-end automation from data to model. To address these limitations, we introduce RNAOmni, an end-to-end autonomous modeling agent for RNA prediction tasks that mimics expert decision-making by jointly exploring data processing strategies, feature representations, model backbones, and hyperparameters, starting solely from raw data. Furthermore, RNAOmni supports both *Direct* and *Thinking* modes, incorporating an attribution probe within the latter to systematically identify modeling weaknesses and guide efficient exploration. Across 17 RNA downstream tasks, RNAOmni achieves average relative improvements of **59.50%** over AutoML methods and **18.4%** over AI research agents. Its generated solutions consistently surpass domain-specific RNA models and further enhance the utility of RNA foundation models. These results demonstrate that RNAOmni enables a new paradigm for RNA downstream modeling by shifting the field from expert-dependent engineering toward autonomous, data-driven model development. Code will be available upon acceptance.

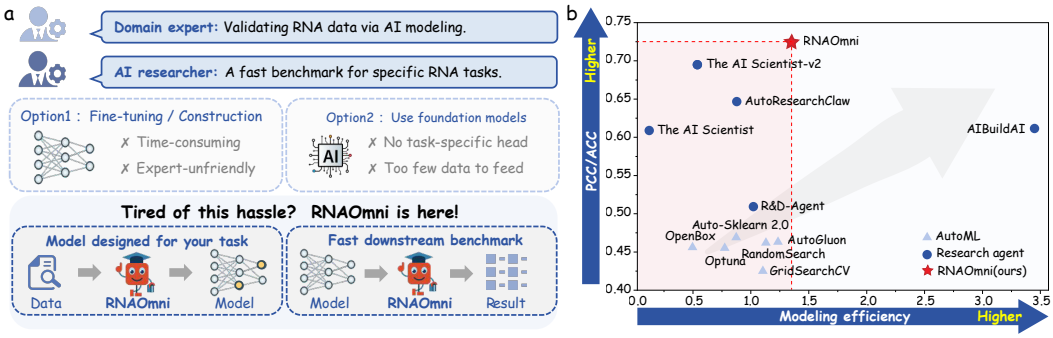


Figure 1: **a: Motivation Statement.** RNAOmni bridges domain experts (data-rich, dev-poor) and AI researchers (model-ready, benchmark-slow) around the shared task of rapid model building, offering an end-to-end pipeline from data input to adapted model and result output. **b: Comparison between RNAOmni and 11 baseline methods across 17 RNA downstream tasks.** The horizontal axis denotes training efficiency (reciprocal of training time in hours), while the vertical axis represents test-set performance, evaluated by Pearson Correlation Coefficient (PCC) or Accuracy (ACC) ranging from 0 to 1.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Given RNA's role as a core molecule of life, its property prediction and functional identification have become a vital interdisciplinary direction at the intersection of AI and biology (Zhang et al., 2025; Degenhardt et al., 2025; Feng et al., 2025). Customizing high-performing AI models is a frequent yet labor-intensive task, requiring iterative feature exploration, model design and refinement, and hyperparameter tuning (Zhang et al., 2026b). However, a gap exists between demand and development capacity: *domain experts often have rich data but limited engineering bandwidth, while AI researchers possess strong modeling capabilities yet cannot cover all domain-specific needs*. This disconnect is particularly glaring in RNA foundation models (Shen et al., 2024; Penić et al., 2025). Despite immense pre-training costs, these high-capacity models remain severely underutilized, largely because downstream adaptation demands specialized engineering expertise and prohibitive fine-tuning budgets. These tensions point to a core need for a rapid downstream task adaptation bridge that connects domain-specific demands with general-purpose technologies, ultimately enabling powerful domain models to deliver strong performance on real-world tasks.

Although existing AI-driven modeling methods can partially automate model construction, they remain constrained by limited accuracy on RNA tasks and incomplete end-to-end automation (Tie et al., 2026). AutoML methods such as Auto-Sklearn optimize hyperparameters within predefined spaces, requiring manually specified search spaces and achieving limited accuracy, with an average metric of only 0.445 on a scale from 0 to 1 (Fig. 1b) (Feurer et al., 2022; Li et al., 2021). LLM-based coding agents such as OpenHands construct models via code generation and editing from user instructions and feedback, while autonomous model-building agents further automate the pipeline from task descriptions to evaluation (Wang et al., 2025; Zhang et al., 2026b). However, these approaches limited by insufficient autonomous task understanding and model adaptation capabilities, often requiring explicit task descriptions and initial modeling strategies beyond the provided data files. These limitations burden domain experts with limited coding experience and require AI researchers to manually adapt general agents to diverse RNA downstream tasks (Appendix A).

In practice, effective modeling requires a series of informed decisions: *what data characteristics matter, which feature representations suit them, and which backbone and hyperparameters fit the task*. Yet current agents typically lack the ability to make these judgments autonomously and diagnose modeling bottlenecks. These two gaps reinforce each other: without understanding task-specific data characteristics, exploration lacks direction, while without diagnostic feedback, model refinement lacks guidance. As a result, their search remains inaccurate and inefficient.

To address these limitations, we present RNAOmni, an autonomous modeling agent tailored for RNA prediction tasks (Fig. 1a). RNAOmni enables complete end-to-end automation from raw data to final predictions, addressing the needs of both domain experts and AI researchers for rapid and cost-effective model development. By mimicking the iterative workflow of human model developers, RNAOmni autonomously explores data characteristics, feature representations, backbone designs, and hyperparameter spaces to construct task-specific models with improved predictive performance, thereby identifying the best configuration suited to each task. Crucially, RNAOmni incorporates an attribution probe that diagnoses modeling bottlenecks and provides actionable feedback to guide efficient search and decision-making. We validate RNAOmni through benchmarking against state-of-the-art autonomous modeling agents and applying RNAOmni to refit task heads for RNA foundation models. Our main contributions are summarized as follows:

- We introduce RNAOmni, an autonomous modeling agent that enables end-to-end RNA model development from raw data by exploring and optimizing data sampling, feature representations, model backbones, and hyperparameters.
- We propose an attribution-guided decision mechanism with an attribution probe that mimics human modeling workflows by diagnosing bottlenecks and guiding targeted refinements, improving the balance between performance and search efficiency.
- We benchmark RNAOmni on 17 diverse RNA downstream tasks (Fig. 1b), demonstrating average improvements of 59.50% and 18.4% in accuracy-efficiency trade-offs over AutoML methods and AI research agents, respectively. RNAOmni also outperforms RNA-specific predictors and optimizes downstream heads for RNA foundation models.

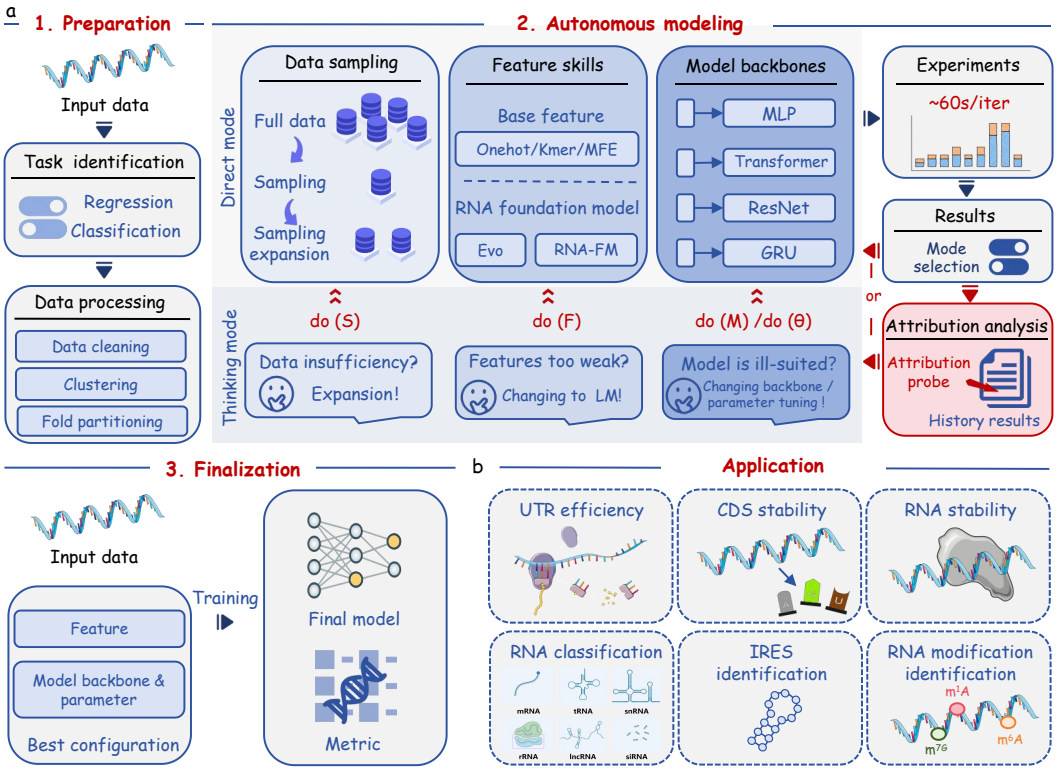


Figure 2: **a: Overall framework of RNAOmni.** Input RNA data undergo preparation, followed by autonomous modeling that optimizes sampling, features, backbones, and parameters through iterative experiments. Exploration is independent of previous trials in *Direct* mode, while *Thinking* mode is trial-guided and equipped with an attribution probe. In the finalization stage, the model is retrained and evaluated using the best configuration to produce the final result. **b:** RNAOmni supports diverse downstream RNA applications across regression and classification tasks.

2 RNAOMNI

2.1 RNAOMNI FRAMEWORK

The overall RNAOmni pipeline consists of three stages (Fig. 2a): the *Data preparation stage*, which infers task types and constructs task-specific datasets; the *Autonomous modeling stage*, which iteratively explores data sampling, feature skills, model backbones, and parameters; and the *Finalization stage*, which retrains the optimal configuration and reports test-set performance (parameter settings provided in Appendix B). RNAOmni supports diverse downstream RNA tasks (Fig. 2b).

The autonomous modeling problem addressed by RNAOmni is formulated as follows. Given a downstream task dataset X, RNAOmni searches over a configuration space defined by data sampling scale S, feature skills F, model backbones M, and parameters θ. An iterative search operator ITR performs multiple rounds of "evaluate–decide–revert" to identify the optimal configuration R_{bestc}:

$$R_{\text{bestc}} = \text{ITR}\big(X; S, F, M, \theta\big). \tag{1}$$

At each iteration, the current configuration is represented as (s, f, m, ϑ), where $s \in S$, $f \in F$, $m \in M$, and $\vartheta \in \theta$ denote the sampling scale, feature recipe, model backbone, and parameter set, respectively. Given this configuration, the forward model ϕ maps sampled and encoded features $f(s(X))$ to predictions Y, which are evaluated using the decision metric D:

$$Y = \phi_{\vartheta}^{m}\big(f(s(X))\big). \tag{2}$$

After iterative exploration, the finalization stage takes the full dataset X and the optimized configuration R_{bestc} as inputs, retraining the final model R_{finalm} and reporting its test-set metric R_{metric}:

$$(R_{\text{finalm}}, R_{\text{metric}}) = \text{FIN}\big(X; R_{\text{bestc}}\big). \tag{3}$$

2.2 AUTONOMOUS MODELING ENGINE

The core of RNAOmni is an autonomous modeling mechanism designed to overcome the limited ability of existing agents to identify task-appropriate data, feature representations, and model configurations while diagnosing bottlenecks during refinement. RNAOmni treats these components as dynamically adjustable actions and iteratively evolves the modeling configuration (Fig. 2a):

Definition 1 (Autonomous modeling mechanism). RNAOmni formulates model development as an iterative action-selection process over four dimensions:

$$\mathcal{A} = \{\mathrm{do}(S), \mathrm{do}(F), \mathrm{do}(M), \mathrm{do}(\theta)\},$$

where $\mathcal{A}$ denotes the action space, and S, F, M, and θ denote data sampling, feature construction, model backbone, and parameters, respectively.

The action selection follows two operational modes. In *Direct* mode, actions are randomly sampled to explore possible configurations, whereas in *Thinking* mode, actions are selected using accumulated attribution-probe information to enable experience-guided exploration. Each candidate action is retrained under a fixed 60-second budget and retained only when it yields sufficient validation improvement; otherwise, the previous configuration is restored.

The four modeling dimensions provide complementary exploration capabilities. For $\mathrm{do}(S)$, RNAOmni dynamically adjusts the sample size under a fixed encoding-time budget, expanding the sampled subset when model performance indicates insufficient data. For $\mathrm{do}(F)$, RNAOmni explores extensible feature skills (Xu & Yan, 2026), including sequence, structural, and thermodynamic descriptors (Li et al., 2018; Moeckel et al., 2024; Lorenz et al., 2011), as well as pretrained representations that capture contextual and evolutionary information (Shen et al., 2024; Nguyen et al., 2024). It further optimizes feature selection and fusion to construct task-specific feature recipes. For $\mathrm{do}(M)$ and $\mathrm{do}(\theta)$, RNAOmni jointly explores model backbones and parameters, covering representative architectures such as pooling heads and Transformers (Zhang et al., 2026a; Han et al., 2021). Within each backbone, $\mathrm{do}(\theta)$ updates model parameters, whereas $\mathrm{do}(M)$ triggers backbone replacement when optimization stagnates. Specifically, replacement occurs when the current backbone is not suppressed and consecutive non-improving rounds reach a predefined threshold:

$$\mathrm{switch}(m) \triangleq \mathbb{I}[c_{\mathrm{supp}}(m) = 0 \wedge c_{\mathrm{stall}} \geq \tau_{\mathrm{stall}}], \tag{4}$$

where $c_{\mathrm{supp}}(m)$ indicates whether backbone m is suppressed, c_{stall} counts consecutive non-improving rounds, and τ_{stall} denotes the stagnation threshold. A complete list of feature skills and backbones is provided in Appendix C.

2.3 ATTRIBUTION-GUIDED EXPLORATION

Efficient exploration requires not only generating new modeling configurations but also understanding why previous trials succeed or fail. In *Thinking* mode, RNAOmni incorporates an attribution probe to analyze historical trial outcomes, identify ineffective exploration directions, and guide subsequent modeling actions.

Definition 2 (Attribution probe). The attribution probe assigns each trial to a single action $a_t \in \{\mathrm{do}(S), \mathrm{do}(F), \mathrm{do}(M), \mathrm{do}(\theta)\}$ through $c_t = \mathrm{attr}(a_t, s_t)$, where c_t denotes the attributed action of trial t, and $s_t \in \{\mathrm{keep}, \mathrm{discard}, \mathrm{crash}\}$ represents the trial outcome.

By accumulating attribution information across iterations, *Thinking* mode infers the current modeling bottleneck from c_t and selects the next action through the bottleneck-to-action mapping:

$$a_{t+1} = \begin{cases} \mathrm{do}(S), & \text{data are insufficient,} \\ \mathrm{do}(F), & \text{features are insufficient,} \\ \mathrm{do}(M), & \text{the backbone is unsuitable,} \\ \mathrm{do}(\theta), & \text{parameters require adjustment.} \end{cases} \tag{5}$$

This attribution-guided mechanism enables *Thinking* mode to transform trial outcomes into actionable feedback, improving exploration efficiency through experience-guided decision making.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Table 1: Summary of RNA tasks and dataset sizes

Index	RNA Task	Samples	Index	RNA Task	Samples	Index	RNA Task	Samples
T1	MusTE	849	T7	MRL	91,519	T13	NC88CLS	23,200
T2	PC3TE	8,508	T8	Deg50C	2,400	T14	NC88withnoiseCLS	23,200
T3	HEKTE	9,619	T9	DegMg50C	2,400	T15	IRES_SPECLS	1,244
T4	MusEL	849	T10	DegMgpH10	2,400	T16	IRES_Identification	2,408
T5	PC3EL	8,508	T11	DegpH10	2,400	T17	ModificationCLS	292,528
T6	HEKEL	9,619	T12	DegR	2,400			

We evaluate RNAOmni on 17 RNA downstream tasks covering both regression and classification scenarios (Table 1). The regression tasks include UTR translation efficiency prediction (T1–T6) and RNA stability prediction (T7–T12), while the classification tasks involve non-coding RNA classification (T13–T14), internal ribosome entry site (IRES) identification (T15–T16), and RNA modification classification (T17) (Penić et al., 2025; Das et al., 2020; Griffiths-Jones et al., 2005; Zhao et al., 2020; Liang et al., 2024). The detailed data sources and descriptions are provided in Appendix D. The dataset sizes range from 849 to 292,528 samples, forming a diverse benchmark for evaluating the generalization capability of RNAOmni. For baselines from original publications, we used results obtained with the same data processing protocols; all other experiments were conducted using RNAOmni's default cross-validation or clustering-based splitting strategies to minimize potential data leakage and ensure fair comparisons. LLM-based methods were powered by the DeepSeek-V4-Flash (Xu et al., 2026), while all experiments were performed on an NVIDIA A100 GPU.

3.2 MAIN RESULTS

We first systematically compare RNAOmni with representative agent-based methods (AIBuildAI (Zhang et al., 2026b), The AI Scientist (Lu et al., 2026) and its v2 version (Yamada et al., 2025), AutoResearchClaw (Liu et al., 2026), and R&D-Agent (Yang et al., 2025)) and conventional AutoML methods (Auto-Sklearn 2.0 (Feurer et al., 2022), AutoGluon (Erickson et al., 2020), GridSearchCV (Alemerien et al., 2024), Openbox (Li et al., 2021), Optuna (Akiba et al., 2019), and Random-search (Bergstra & Bengio, 2012)) across 17 RNA downstream tasks (Table 2).

RNAOmni achieves consistently strong performance across both regression and classification tasks. On the 12 regression benchmarks, RNAOmni obtains an average PCC of 0.622, compared with 0.536 for agent-based methods and 0.347 for conventional AutoML methods. On the 5 classification benchmarks, RNAOmni achieves an average accuracy of 0.974, exceeding agent-based methods (0.854) and AutoML methods (0.782) by 14.0% and 24.6%, respectively. Here and below, percentage improvements are reported as relative improvements over the corresponding baselines. Across all 17 tasks, RNAOmni reaches an overall average score of 0.726, compared with 0.613 for agent-based methods and 0.455 for AutoML methods, yielding improvements of 18.4% and 59.5%, respectively. These results indicate that RNAOmni can effectively adapt its modeling strategy across diverse RNA prediction scenarios, while general-purpose agents exhibit less stable performance across tasks and conventional AutoML methods remain constrained by limited search spaces and modeling flexibility. Furthermore, computational analysis (Table 3) shows that RNAOmni completes the modeling pipeline, excluding feature encoding time, in approximately 44 minutes without manual task specification or architecture template design. Together, these results demonstrate that RNAOmni provides an effective trade-off between predictive performance, computational cost, and end-to-end automation for diverse RNA downstream modeling tasks.

3.3 ABLATION STUDY

RNAOmni introduces a developer-centric paradigm for RNA model construction and refinement, where model development follows a human-like iterative process of diagnosing performance bottle-

Table 2: Comparison with agent-based and automated machine learning methods across RNA downstream tasks. Regression and classification tasks are evaluated by PCC and ACC, respectively. "TX" denotes the task index. Best and second-best scores are shown in **bold** and underlined, respectively.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
Method	PCC	PCC	PCC	PCC	PCC	PCC	PCC	PCC	PCC	PCC	PCC	PCC	ACC	ACC	ACC	ACC	ACC
AIBuildAI	0.35	0.44	0.38	0.32	0.33	0.29	0.86	0.44	0.68	0.71	0.22	0.61	0.98	0.96	0.82	**1.00**	**0.99**
The AI Scientist	0.53	0.42	0.42	0.44	0.37	0.48	0.84	0.34	0.65	0.66	0.36	0.55	0.97	0.52	0.81	**1.00**	**0.99**
The AI Scientist-v2	0.54	0.47	**0.66**	0.61	0.43	0.41	**0.95**	0.56	0.62	0.66	0.64	**0.75**	0.98	0.82	0.78	0.99	0.91
AutoResearchClaw	0.56	0.50	0.44	0.57	0.40	**0.52**	0.77	0.50	0.46	0.67	0.42	**0.75**	0.97	0.80	0.81	**1.00**	0.85
R&D-Agent	0.25	0.34	0.23	**0.84**	0.43	0.16	0.89	0.50	0.38	0.59	0.09	0.52	0.97	0.56	0.82	**1.00**	0.06
Auto-Sklearn 2.0	0.12	0.15	0.15	0.14	0.08	0.07	0.60	0.48	0.57	0.66	0.33	0.58	0.79	0.62	0.73	0.99	0.91
AutoGluon	0.15	0.12	0.04	0.05	0.09	0.01	0.64	0.46	0.66	0.62	0.39	0.63	0.77	0.65	0.68	0.99	0.91
GridSearchCV	0.21	0.11	0.08	0.08	0.11	0.00	0.22	0.43	0.65	0.68	0.38	0.59	0.69	0.37	0.66	0.99	**0.99**
OpenBox	0.11	0.11	0.07	0.16	0.08	0.02	0.59	0.47	0.66	0.65	0.36	0.60	0.74	0.52	0.65	0.99	**0.99**
Optuna	0.14	0.13	0.07	0.11	0.08	0.04	0.55	0.48	0.64	0.69	0.31	0.62	0.72	0.50	0.66	0.99	**0.99**
RandomSearch	0.26	0.10	0.07	0.11	0.06	-0.01	0.53	0.44	0.66	0.67	0.38	0.62	0.74	0.53	0.71	0.99	**0.99**
RNAOmni	**0.57**	**0.52**	0.45	0.59	**0.45**	0.50	0.87	**0.67**	**0.70**	**0.72**	**0.68**	**0.75**	**0.99**	**0.99**	**0.90**	**1.00**	**0.99**

Table 3: Time cost and extra preparation (Extra *prepar.*) of the compared methods. "Yes" means that manual inputs (e.g., task descriptions or model templates) are required, while "No" means no extra input beyond data is needed.

Method	**Time cost (min)**	**Extra *prepar.***	**Method**	**Time cost (min)**	**Extra *prepar.***
AIBuildAI	17.31	Yes	AutoGluon	48.60	No
The AI Scientist	487.64	Yes	GridSearchCV	54.36	No
The AI Scientist-v2	112.67	Yes	OpenBox	120.90	No
AutoResearchClaw	68.28	Yes	Optuna	77.42	No
R&D-Agent	58.76	Yes	RandomSearch	53.08	No
Auto-Sklearn 2.0	68.66	No	RNAOmni	44.34	No

necks and applying targeted refinements to improve predictive performance. The attribution probe (AP) serves as a key component of this framework by providing feedback-driven guidance for model refinement. To evaluate its operational capability and assess the contribution of AP, we conduct ablation and robustness experiments on the T6 task by comparing two operational modes: *Direct* mode without AP and *Thinking* mode with AP-enabled attribution feedback (Fig. 3).

As shown in Fig. 3a–b, *Direct* mode relies on random sampling, resulting in scattered search trajectories, whereas *Thinking* mode leverages AP for targeted, feedback-driven refinement. In Case 1 ("Fix the judgment"), AP identifies performance degradation and improves the model by expanding data sampling and strengthening feature representations. In Case 2 ("Break the plateau"), AP detects tail-dominated outliers during stagnation and dynamically replaces MSE with Huber loss. This attribution-guided feedback enables RNAOmni to escape local optima, while *Direct* mode lacks adaptive responses to such failure cases. Quantitative results (Fig. 3c–e) show that AP improves average PCC from 0.446 to 0.482, increases the effective modification rate (28.57% vs. 31.27%), and reduces exploration trials (58.00 vs. 35.33). All values are averaged over three independent runs. Evaluation across diverse LLM APIs (Fig. 3f) further demonstrates consistent improvements without requiring expensive high-tier models. Appendix E provides a demo of *Direct* and *Thinking* mode on the HEKEL task (T6), including the complete autonomous modeling process and the final test set metrics. In summary, these results show that AP converts blind exploration into an efficient, evidence-driven optimization loop, enabling accurate RNA model development with fewer exploration trials.

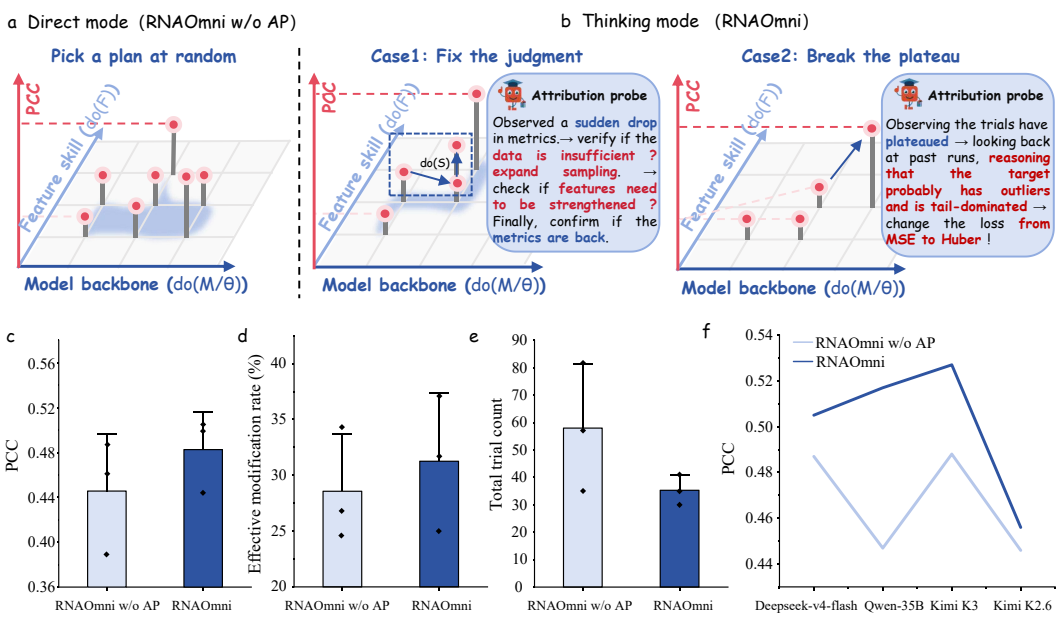


Figure 3: **Ablation study and robustness on the T6 task. a–b**: Comparison of the two operational modes: *Direct* mode without the attribution probe (RNAOmni w/o AP) and *Thinking* mode with attribution feedback (RNAOmni). Shaded regions indicate the search footprints, revealing random exploration in *Direct* mode (**a**) versus more directional and efficient exploration in *Thinking* mode (**b**). **b** illustrates two representative cases in which RNAOmni converges to the correct solution through iterative attribution-guided refinement. **c–f**: Ablation studies assessing the contribution of the attribution probe, with three independent runs performed for each configuration to ensure reproducibility. **c**: Comparison of predictive performance measured by PCC. **d**: Number of effective model modifications retained during the modeling process. **e**: Total number of exploration trials performed during optimization. **f**: Robustness evaluation across different underlying LLM APIs.

3.4 Comparison of the generated models

We comprehensively benchmark RNAOmni against state-of-the-art and representative RNA prediction models across both regression and classification tasks, covering three major categories of approaches: general-purpose biological foundation models, such as the EVO series (Brixi et al., 2026; Nguyen et al., 2024); RNA-specialized pretrained models (Penić et al., 2025; Chu et al., 2024; Shen et al., 2024; Cao et al., 2021; Sample et al., 2019; Li et al., 2018; Wang et al., 2023); and conventional sequence-based baselines (Testagrose & Boucher, 2025). EVO and EVO2 use ordinary least squares (OLS) and Softmax task heads under the same data configuration (Brixi et al., 2026; Nguyen et al., 2024), while other results are reported from the original papers (Penić et al., 2025).

In regression benchmarks (Table 4), RNAOmni achieves the best or second-best performance across all six tasks, obtaining the highest Spearman correlations on MusTE, HEKTE, and MusEL (0.74, 0.67, and 0.72, respectively). On PC3TE, PC3EL, and HEKEL, RNAOmni performs comparably to or slightly surpasses RiNALMo (Penić et al., 2025). In contrast, domain-specific models such as UTR-LM (Chu et al., 2024) and RNA-FM (Shen et al., 2024) show lower overall accuracy, while feature-frozen variants exhibit substantial degradation, highlighting the importance of task-specific adaptation. In classification benchmarks (Table 5), RNAOmni achieves 100% accuracy on NC88CLS and 99% on the noisy NC88withnoiseCLS, matching the best-performing baselines. Notably, several competing architectures, including the EVO model family (Brixi et al., 2026; Nguyen et al., 2024), experience pronounced performance drops under noisy conditions. Together, these results demonstrate that RNAOmni matches or exceeds specialized RNA models across diverse downstream tasks, providing a scalable and adaptable framework for complex RNA modeling.

Table 4: **Comparison with SOTA RNA prediction models on regression tasks using identical data.** Spearman correlation is reported for all prediction tasks. "FT" indicates fine-tuning status. Best and second-best scores are shown in **bold** and underlined, respectively.

Model	FT	MusTE	PC3TE	HEKTE	MusEL	PC3EL	HEKEL
EVO2-OLS	×	0.49	0.35	0.32	0.52	0.30	0.31
EVO-OLS	×	0.45	0.32	0.29	0.50	0.27	0.28
RiNALMo	✓	0.72	**0.70**	0.66	0.70	**0.67**	**0.69**
RiNALMo-frozen	✓	0.38	0.35	0.25	0.36	0.18	0.20
RiNALMo-150M	✓	0.70	0.66	0.62	0.68	0.63	0.66
RiNALMo-33M	✓	0.69	0.67	0.63	0.70	0.63	0.65
UTR-LM	×	0.67	0.65	0.60	0.66	0.63	0.65
RNA-FM	✓	0.62	0.60	0.55	0.59	0.49	0.51
Cao-RF	×	0.63	0.63	0.55	0.64	0.59	0.57
Optimus 5-Prime	×	0.41	0.38	0.36	0.15	0.19	0.18
One-Hot	×	0.34	0.34	0.25	0.32	0.12	0.14
RNAOmni	×	**0.74**	0.69	**0.67**	**0.72**	**0.67**	**0.69**

Table 5: **Comparison with SOTA RNA classification models.** Accuracy is reported for noiseless and noisy RNA family classification. "FT" indicates fine-tuning. Best and second-best scores are marked in **bold** and underlined, respectively.

Model	FT	NC88CLS	NC88withnoiseCLS
EVO2-Softmax	×	0.42	0.32
EVO-Softmax	×	0.32	0.22
RiNALMo	✓	0.99	**0.99**
RiNALMo-frozen	✓	0.97	0.68
RiNALMo-150M	✓	0.99	**0.99**
RiNALMo-33M	✓	0.98	0.98
Uni-RNA	×	0.99	0.98
RNA-FM	✓	0.92	0.96
One-Hot	×	0.31	0.24
3-mer	×	0.89	0.84
2-mer	×	0.88	0.81
1-mer	×	0.87	0.81
RNAOmni	×	**1.00**	**0.99**

3.5 Refitting Task Heads for RNA Foundation Models

The experiments in this section demonstrate that RNAOmni enhances the downstream adaptation of existing RNA foundation models. We equipped four representative models (RiNALMo-33M (Penić et al., 2025), RNA-FM (Shen et al., 2024), Evo (Nguyen et al., 2024), and Evo2 (Brixi et al., 2026)) with automatically constructed task heads and evaluated their performance across regression task (measured by Spearman correlation or RMSE) and classification task (measured by accuracy).

The task heads were automatically constructed and optimized while keeping the foundation model backbone frozen, allowing us to evaluate their task-specific adaptation quality without manual head engineering or inconsistent training protocols. As shown in Fig. 4, RNAOmni-derived heads consistently improved downstream performance across all evaluated models. For regression tasks, RNAOmni achieved average relative Spearman correlation coefficient (SCC) gains of 1.96% and 13.50% over RiNALMo-33M (Penić et al., 2025) and RNA-FM (Shen et al., 2024), respectively, and reduced RMSE by 6.64%–57.91% compared with Evo (Nguyen et al., 2024) and Evo2 (Brixi et al., 2026). For classification tasks, RNAOmni substantially improved Evo (Nguyen et al., 2024) and

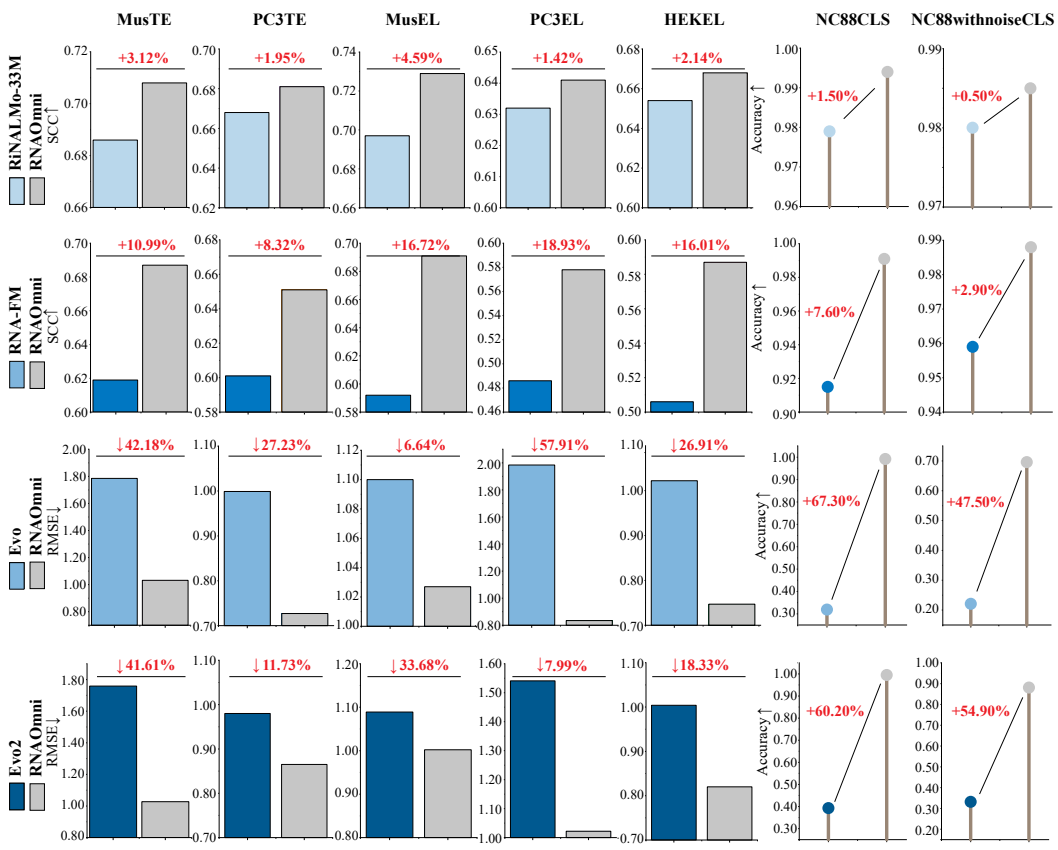


Figure 4: **RNAOmni enhances the downstream application capabilities of RNA language models.** Results for RiNALMo-33M and RNA-FM are cited from the literature, while Evo and Evo2 are evaluated with a two-layer MLP as the task head for regression tasks and a softmax classifier as the task head for classification tasks. "SCC" denotes Spearman correlation coefficient. In each panel, the left and right entries denote the baseline and RNAOmni, respectively.

Evo2 (Brixi et al., 2026) from near-random baseline accuracy (ACC = 31.9%–39.3%) to 99.1%–99.5% in noise-free settings, while maintaining robust performance (69.6%–88.1%) under noisy conditions. These results demonstrate that RNAOmni effectively unlocks the downstream potential of diverse RNA foundation models. By eliminating manual head design, RNAOmni provides a standardized and efficient framework for rapidly benchmarking and adapting RNA foundation models across diverse downstream tasks, enabling systematic evaluation of their transferable representations.

4 Conclusion

In this work, we present RNAOmni, an autonomous modeling agent for RNA prediction tasks. RNAOmni enables end-to-end model development by autonomously exploring data, features, model backbones, and hyperparameters, with an attribution probe guiding refinement. Across 17 RNA downstream tasks, RNAOmni achieves a favorable accuracy-efficiency trade-off, outperforming AutoML methods and AI research agent baselines by 59.50% and 18.4% on average, respectively. Its generated solutions improve RNA prediction models and downstream adaptation of RNA foundation models. **Limitations:** RNAOmni is currently evaluated primarily on single-label prediction tasks; future work will extend it to multi-label prediction and RNA design and generation. **Broader Impacts:** RNAOmni reduces the expertise and engineering effort required for RNA model development, potentially accelerating AI applications in RNA biology.

AI USE STATEMENT

In this work, generative AI tools were used for manuscript editing and language polishing, limited to improving grammar, clarity, and presentation. These tools were not used for study design, data analysis, result interpretation, or generation of scientific content. The analysis, framing, and conclusions reflect the authors' expertise in bioinformatics and independent scholarly judgment. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we provide a comprehensive account of our code, model design, experimental setup, and datasets, all of which are referenced in the relevant parts of the paper and appendix as detailed below. The complete source code will be publicly released upon acceptance of this paper. Details of the proposed model and algorithm, together with the full experimental configurations (including hyperparameters, training procedures, and evaluation protocols), can be found in Appendix B and Appendix C. All datasets used in our experiments are described in Appendix D, where we detail the data sources, preprocessing steps, and the train/validation/test splits adopted throughout this work. For the experimental results, the ablation studies presented in Section 3.3 report outcomes from three independent runs as well as results obtained with different underlying models, which together demonstrate the stability and reproducibility of our approach across varying settings. We hope that these materials, taken together, will allow the community to reproduce and build upon our findings with ease.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Khalid Alemerien, Saleel Alsarayreh, and Enshirah Altarawneh. Diagnosing cardiovascular diseases using optimized machine learning algorithms with GridSearchCV. *Journal of Applied Data Sciences*, 5(4):1539–1552, 2024.

James Bergstra and Yann Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

Garyk Brixi, Matthew G. Durrant, Jerome Ku, Mohsen Naghipourfar, Michael Poli, Gwanggyu Sun, Greg Brockman, Daniel Chang, Alison Fanton, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Jonathan C. Schmok, Ali Taghibakhshi, Anton Vorontsov, Brandon Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, and Brian L. Hie. Genome modelling and design across all domains of life with Evo 2. *Nature*, 652:1349–1361, 2026.

Jicong Cao, Eva Maria Novoa, Zhizhuo Zhang, William C. W. Chen, Dianbo Liu, Gigi C. G. Choi, Alan S. L. Wong, Claudia Wehrspaun, Manolis Kellis, and Timothy K. Lu. High-throughput 5′ UTR engineering for enhanced protein production in non-viral gene therapies. *Nature Communications*, 12(4138), 2021.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *Proceedings of the International Conference on Learning Representations*, 2025.

Yanyi Chu, Dan Yu, Yupeng Li, Kaixuan Huang, Yue Shen, Le Cong, Jason Zhang, and Mengdi Wang. A 5′ UTR language model for decoding untranslated regions of mRNA and function predictions. *Nature Machine Intelligence*, 6:449–460, 2024.

Rhiju Das, Hannah Wayment-Steele, Christian Do Soon Kim, Christian Choe, Bojan Tunguz, Walter Reade, and Maggie Demkin. OpenVaccine: COVID-19 mRNA vaccine degradation prediction, 2020. URL `https://kaggle.com/competitions/stanford-covid-vaccine`.

Maximilia F. S. Degenhardt, Hermann F. Degenhardt, Yuba R. Bhandari, Yun-Tzai Lee, Jienyu Ding, Ping Yu, William F. Heinz, Jason R. Stagno, Charles D. Schwieters, Norman R. Watts, Paul T. Wingfield, Alan Rein, Jinwei Zhang, and Yun-Xing Wang. Determining structures of RNA conformers using AFM and deep neural networks. *Nature*, 637:1234–1243, 2025.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-Tabular: Robust and accurate AutoML for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Ziyan Feng, Xuelian Pang, Qian Xu, Zijie Gu, Shiliang Li, Lili Zhu, and Honglin Li. Design of a multi-valent SARS-CoV-2 peptide vaccine for broad immune protection via deep learning. *Engineering*, 52:142–159, 2025.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free AutoML via meta-learning. *Journal of Machine Learning Research*, 23 (261):1–61, 2022.

Sam Griffiths-Jones, Simon Moxon, Mhairi Marshall, Ajay Khanna, Sean R. Eddy, and Alex Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*, 33 (suppl_1):D121–D124, 2005.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15908–15919, 2021.

Jianwei Li, Yan Huang, Xiaoyue Yang, Yiran Zhou, and Yuan Zhou. RNAm5Cfinder: A web-server for predicting RNA 5-methylcytosine (m5c) sites based on random forest. *Scientific Reports*, 8 (17299), 2018.

Yang Li, Yu Shen, Wentao Zhang, Yuanwei Chen, Huaijun Jiang, Mingchao Liu, Jiawei Jiang, Jinyang Gao, Wentao Wu, Zhi Yang, Ce Zhang, and Bin Cui. OpenBox: A generalized black-box optimization service. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3209–3219, 2021.

Zhanmin Liang, Haokai Ye, Jiongming Ma, Zhen Wei, Yue Wang, Yuxin Zhang, Daiyun Huang, Bowen Song, Jia Meng, Daniel J. Rigden, and Kunqi Chen. m6A-Atlas v2.0: updated resources for unraveling the N6-methyladenosine (m6A) epitranscriptome among multiple species. *Nucleic Acids Research*, 52(D1):D194–D202, 2024.

Jiaqi Liu, Shi Qiu, Mairui Li, Bingzhou Li, Haonian Ji, Siwei Han, Xinyu Ye, Peng Xia, Zihan Dong, Meng Chen, Congyu Zhang, Letian Zhang, Guiming Chen, Haoqin Tu, Xinyu Yang, Lu Feng, Xujiang Zhao, Haifeng Chen, Jiawei Zhou, Xiao Wang, Weitong Zhang, Hongtu Zhu, Yun Li, Jieru Mei, Hongliang Fei, Jiaheng Zhang, Linjie Li, Linjun Zhang, Yuyin Zhou, Sheng Wang, Caiming Xiong, James Zou, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. AutoResearchClaw: Self-reinforcing autonomous research with human-AI collaboration. *arXiv preprint arXiv:2605.20025*, 2026.

Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F. Stadler, and Ivo L. Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(26), 2011.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651:914–919, 2026.

Camille Moeckel, Manvita Mareboina, Maxwell A. Konnaris, Candace S. Y. Chan, Ioannis Mouratidis, Austin Montgomery, Nikol Chantzi, Georgios A. Pavlopoulos, and Ilias Georgakopoulos-Soares. A survey of k-mer methods and applications in bioinformatics. *Computational and Structural Biotechnology Journal*, 23:2289–2303, 2024.

Eric Nguyen, Michael Poli, Matthew G. Durrant, Brian Kang, Dhruva Katrekar, David B. Li, Liam J. Bartie, Armin W. Thomas, Samuel H. King, and Brian L. Hie. Sequence modeling and design from molecular to genome scale with Evo. *Science*, 386(6723):eado9336, 2024.

Rafael Josip Penić, Tin Vlašić, Roland G. Huber, Yue Wan, and Mile Šikić. RiNALMo: general-purpose RNA language models can generalize well on structure prediction tasks. *Nature Communications*, 16(5671), 2025.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys*, 54(4):1–34, 2021.

Paul J. Sample, Ban Wang, David W. Reid, Vlad Presnyak, Iain J. McFadyen, David R. Morris, and Georg Seelig. Human 5′ UTR design and variant effect prediction from a massively parallel translation assay. *Nature Biotechnology*, 37:803–809, 2019.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.

Tao Shen, Zhihang Hu, Siqi Sun, Di Liu, Felix Wong, Jiuming Wang, Jiayang Chen, Yixuan Wang, Liang Hong, Jin Xiao, Liangzhen Zheng, Tejas Krishnamoorthi, Irwin King, Sheng Wang, Peng Yin, James J. Collins, and Yu Li. Accurate RNA 3D structure prediction using a language model-based deep learning approach. *Nature Methods*, 21:2287–2298, 2024.

Conrad Testagrose and Christina Boucher. Tokenization and deep learning architectures in genomics: A comprehensive review. *Computational and Structural Biotechnology Journal*, 27: 3547–3555, 2025.

Guiyao Tie, Jiawen Shi, Dingjie Song, Yixiao Huang, Ziji Sheng, Xueyang Zhou, Daizong Liu, Pan Zhou, Yongchao Chen, Ran Xu, Lifang He, Qingsong Wen, Manling Li, Cong Lu, Shuai Li, Pengtao Xie, Yixuan Yuan, Rui Meng, Lei Xing, Lichao Sun, Caiming Xiong, Philip S. Yu, and Jianfeng Gao. AutoResearch AI: Towards AI-powered research automation for scientific discovery. *arXiv preprint arXiv:2605.23204*, 2026.

Xi Wang, Ruichu Gu, Zhiyuan Chen, Yongge Li, Xiaohong Ji, Guolin Ke, and Han Wen. UNI-RNA: Universal pre-trained models revolutionize RNA research. *bioRxiv*, 2023. doi: https://doi.org/10.1101/2023.07.11.548588.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *Proceedings of the International Conference on Learning Representations*, pp. 65882–65919, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, Haoran Pan, Yuge Zhang, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. R&D-Agent: An LLM-agent framework towards autonomous data science. *arXiv preprint arXiv:2505.14738*, 2025.

Di Zhang, Zhangpeng Gong, Xiaobo Pang, Jiashuai Liu, Junbo Lu, Hao Cui, Jiusong Ge, Zhi Zeng, Kai Yi, Yinghua Li, Si Liu, Tingsong Yu, Haoran Wang, Mireia Crispin-Ortuzar, Weimiao Yu, Chen Li, and Zeyu Gao. CARE: A molecular-guided foundation model with adaptive region modeling for whole slide image analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

He Zhang, Hailong Liu, Yushan Xu, Haoran Huang, Yiming Liu, Jia Wang, Yan Qin, Haiyan Wang, Lili Ma, and Jicong Cao. Deep generative models design mRNA sequences with enhanced translational capacity and stability. *Science*, 390(6773):eadr8470, 2025.

Ruiyi Zhang, Peijia Qin, Qi Cao, Li Zhang, and Pengtao Xie. AIBuildAI: An AI agent for automatically building AI models. arXiv preprint arXiv:2604.14455, 2026b.

Jian Zhao, Yan Li, Cong Wang, Haotian Zhang, Hao Zhang, Bin Jiang, Xuejiang Guo, and Xiaofeng Song. IRESbase: A comprehensive database of experimentally validated internal ribosome entry sites. *Genomics, Proteomics & Bioinformatics*, 18(2):129–139, 2020.

A Related Work

In this section, we review existing studies closely related to autonomous modeling, which can be broadly categorized into three paradigms: traditional Automated Machine Learning (AutoML) and neural architecture search (NAS), LLM-based coding agents, and autonomous model-building agents.

Traditional AutoML and Neural Architecture Search (NAS). Developing AI models remains labor-intensive and requires substantial domain expertise (Zhang et al., 2026b). Traditional AutoML automates parts of the modeling workflow through hyperparameter optimization (HPO) and model selection within predefined search spaces. Early methods such as grid and random search (Li et al., 2018) established the basis for automated optimization, while modern frameworks including Optuna (Akiba et al., 2019), Auto-Sklearn 2.0, and AutoGluon combine Bayesian optimization, model selection, HPO, and ensembling in end-to-end pipelines (Shahriari et al., 2016; Feurer et al., 2022; Erickson et al., 2020). Neural Architecture Search (NAS) further extends automation to architecture design (Ren et al., 2021). However, these approaches rely on manually defined search spaces and predefined components, leaving model implementation largely to human experts. In specialized domains such as biology, they also lack domain-specific tools and knowledge representations for incorporating physical or chemical constraints and handling complex biomedical data.

LLM-based Coding Agents. With the rapid advancement of LLM-based code intelligence, recent studies have reformulated model development as a code-centric optimization problem, where agents iteratively generate, modify, and refine model implementations based on task descriptions, execution feedback, and intermediate results. Representative systems, such as OpenHands (Wang et al., 2025), employ reasoning and tool-use capabilities for multi-step software engineering task. Nevertheless, these approaches remain constrained by the limited ability of individual LLM calls to maintain long-term objectives, accumulate architectural insights, and perform sustained exploration over complex model design spaces.

Autonomous Model Building Agents. To further reduce human involvement in model development, emerging autonomous agents aim to execute the entire modeling lifecycle from raw task specifications and datasets, including strategy formulation, code generation, optimization, and evaluation (Tie et al., 2026). By integrating multi-agent collaboration, external tools, iterative experimentation, and self-debugging mechanisms, these systems have demonstrated strong performance on general machine learning benchmarks such as MLE-Bench (Chan et al., 2025). Compared with conventional coding agents, they formulate model development as a multi-stage, evidence-driven closed-loop process. However, several fundamental challenges remain. Existing evaluations primarily focus on general machine learning tasks or Kaggle-style competitions, providing limited assessment of scientific domains that require domain knowledge, structured representations, and physical constraints, such as biology, chemistry, and physics. Furthermore, exploration over open-ended model design spaces remains challenging, often leading to inefficient search processes and unstable performance across iterative experiments.

Overall, existing approaches have progressively advanced autonomous modeling from automated optimization of predefined components, to code-level model generation, and finally to end-to-end agentic development. However, current solutions remain limited in their ability to autonomously discover domain-aware modeling strategies, incorporate scientific knowledge and constraints, and efficiently navigate open-ended design spaces. These limitations highlight the need for a new generation of autonomous modeling systems that can integrate domain-specific reasoning, flexible model construction, and iterative self-improvement to address complex scientific problems.

B RNAOMNI FRAMEWORK SETUP DETAIL

Table 6: **Data split.** Data partitioning parameters used for training, validation, and testing across regression and classification tasks.

Parameter	Default Value	Description
N_FOLDS	5	Number of cross-validation folds for regression
TEST_FRACTION	0.1	Regression test hold-out fraction
CLS_TRAIN_FRAC	0.7	Classification training set fraction
CLS_VAL_FRAC	0.2	Classification validation set fraction
CLS_TEST_FRAC	0.1	Classification test set fraction

Table 7: **Autonomous modeling parameters.** Parameters governing search termination and backbone replacement.

Parameter	Default Value	Description
MIN_DELTA_CLASSIFICATION	3×10^{-4}	Minimum validation accuracy improvement
MIN_DELTA_REGRESSION	1×10^{-3}	Minimum validation PCC improvement
τ_{stall}	15	Stagnation threshold for backbone replacement
MIN_ITERATIONS	10	Minimum number of search iterations

Table 8: **Finalization stage parameters.** Parameters governing the finalization stage of the iterative modeling process.

Parameter	Default	Description
Commit	auto	Best commit to finalize (hash or auto)
Early-Stop / No-Early-Stop	0	Early-stopping switch
Patience	5	Number of evals without validation improvement before early stopping
Min-Delta-Ratio	0.002	Threshold relative to the search validation PCC
Seed	42	Random seed

RNAOmni consists of three stages: a *Data preparation stage* that infers the task type and constructs the task data, an *Autonomous modeling stage* that automatically explores the data, feature, model backbone, and parameters while validating each trial by experiment, and a *Finalization stage* that consolidates the best configuration and reports its test-set metric. We evaluate the RNAOmni framework under a unified data setup, with the main data partitioning parameters summarized in Appendix Table 6. For regression tasks, we adopt 5-fold cross-validation with a 0.1 test hold-out fraction, while for classification tasks the data are split into 0.7/0.2/0.1 for training, validation, and testing, respectively. Beyond data partitioning, the iterative modeling process is governed by two sets of scheduling parameters that control when to keep, discard, or stop a candidate configuration. As listed in Appendix Table 7, the search termination criteria include retention thresholds on validation accuracy for classification (3×10^{-4}) and validation Pearson correlation for regression (1×10^{-3}), meaning that a newly explored configuration is kept only if it improves the corresponding validation metric by more than the prescribed margin. In addition, the search stops after 15 consecutive non-significant rounds, switches the current head after 10 consecutive model discards for a single named head, and always runs for a minimum of 10 iterations to ensure sufficient exploration. Once the search terminates, the finalization stage is controlled by the parameters in Appendix Table 8, which specify the best commit to finalize, the early-stopping switch and its patience of 5 evaluations, a minimum delta ratio of 0.002 relative to the search validation PCC, and a random seed of 42 to ensure reproducibility.

Table 9: **Feature skills.** Overview of feature skills and their tunable parameters.

Feature name	Dimension	Description	Tunable Parameters
Onehot	4	Nucleotide one-hot encoding	—
Length	1	Normalized sequence length	—
Kmer	64 (default k=3)	K-mer frequency features	k∈1–4
Struct_thermo	3	Thermodynamic structural features	—
Struct_dotbracket	3	Dot-bracket structure encoding	—
Struct_bpp	2	Base-pair probability features	—
RNA_FM	640	RNA-FM embeddings	layer k∈0–12 (default 12)
RiNALMo	480	RiNALMo embeddings	33M, 150M, 650M
Evo	4096	Evo embeddings	—
Evo2	4096	Evo2 embeddings	—

Table 10: **Model backbones.** Candidate backbone architectures for the prediction head.

Type	Description
Pool_MLP	Mean/max pooling with a residual convolutional stem and MLP head.
Attn_Pool_MLP	Attention pooling over sequence positions followed by MLP prediction.
ResNet1D_Pool	1D ResNet blocks with pooling-based aggregation.
CLS_MLP	Learned CLS token aggregation followed by MLP, without Transformer layers.
TF_Mean	Transformer encoder with mean pooling aggregation.
TF_CLS	Transformer encoder with learned CLS token aggregation.
TF_Attn	Transformer encoder with attention pooling aggregation.
TF_Conv	Convolutional residual block followed by Transformer encoding and pooling.
Max_Pool_MLP	Max pooling with MLP head and no additional encoder.
Gated_Pool_MLP	Learnable gated pooling followed by MLP prediction.
GRU_Pool	Bidirectional GRU with mean pooling aggregation.
Mean_Deep_MLP	Mean pooling followed by a deeper MLP head.

C AUTONOMOUS MODELING ENGINE DETAIL

The RNAOmni framework is built upon a extensible design, in which feature skills and model backbones can be flexibly combined. As summarized in Appendix Table 9, we consider ten feature extraction skills with different dimensions and tunable parameters, including simple sequence-based features such as Onehot, Length, and Kmer, structural features such as Struct_thermo, Struct_dotbracket, and Struct_bpp, as well as embeddings from pretrained RNA foundation models including RNA_FM, RiNALMo, Evo, and Evo2. These features capture complementary aspects of RNA sequences, ranging from raw nucleotide composition and secondary structure to evolutionary and functional representations, with the RNA foundation model embeddings in particular providing rich, pretrained contextual representations that can be readily transferred to a wide range of downstream tasks.

For the prediction head, as listed in Appendix Table 10, we evaluate a diverse set of candidate backbone architectures, ranging from simple multilayer perceptrons (MLPs) (e.g., Pool_MLP, Max_Pool_MLP, and Mean_Deep_MLP) and attention-based variants (e.g., Attn_Pool_MLP and Gated_Pool_MLP) to more expressive designs incorporating residual convolutions, Transformer encoders, and recurrent modules (e.g., ResNet1D_Pool, CLS_MLP, TF_Mean, TF_CLS, TF_Attn, TF_Conv, and GRU_Pool). Together, these two tables define the search space of the RNAOmni framework, allowing different feature skills and backbone architectures to be paired for each downstream task.

D DATA SOURCE

Table 11: **Data source.** Summary of the benchmark datasets used for the RNA downstream tasks considered in this work.

Index	RNA Task	Samples	Data source
T1	MusTE	849	
T2	PC3TE	8,508	
T3	HEKTE	9,619	https://drive.google.com/drive/folders/190oihtrwCxWjtDCK9kJzyhXPKxbr5xoR
T4	MusEL	849	
T5	PC3EL	8,508	
T6	HEKEL	9,619	
T7	MRL	91,519	https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE114002&format=file
T8	Deg50C	2,400	
T9	DegMg50C	2,400	
T10	DegMgpH10	2,400	https://www.kaggle.com/competitions/stanford-covid-vaccine/data?select=test.json
T11	DegpH10	2,400	
T12	DegR	2,400	
T13	NC88CLS	23,200	https://raw.githubusercontent.com/bioinformatics-sannio/ncrna-deep/master/datasets/Rfam-novel/train_test_val_sets_177_families.RDa
T14	NC88withnoiseCLS	23,200	
T15	IRES_SPECLS	1,244	http://reprod.njmu.edu.cn/cgi-bin/iresbase/index.php
T16	IRES_Identification	2,408	
T17	ModificationCLS	292,528	http://180.208.58.19/m6A-Atlas/download.html

We evaluated our model on 17 RNA downstream tasks covering both regression and classification scenarios, including translation efficiency, expression level, ribosome load, RNA degradation, non-coding RNA classification, internal ribosome entry site (IRES) prediction, and RNA modification classification. These datasets span diverse biological tasks and vary substantially in scale, ranging from 849 to 292,528 samples, providing a diverse benchmark for evaluating model performance across different RNA prediction settings. To ensure consistent evaluation, we applied task-specific data processing and splitting strategies. For regression tasks, we adopted 5-fold cross-validation with a 0.1 test hold-out fraction, while for classification tasks, the data were split into 0.7/0.2/0.1 for training, validation, and testing, respectively. All datasets are publicly available, with their data sources, preprocessing details, and download links provided in Appendix Table 11.

E A DEMO OF DIRECT AND THINKING MODE ON THE HEKEL TASK

Direct mode on T6 task

Autonomous Modeling Stage: 82 experiments attempted, keep rate 26.8%.
Finalization Stage: Test Pearson = 0.461

Trial	val pearson	status	RNAOmni Improvement
1	0.241365	keep	[feature] kmer k=4 (C=256) for richer k-mer context (input_dim 74->266, pool_mlp)
2	0.009952	discard	[model] pool_mlp: rebuild to stem->WIDTH + conv residual at width (framework=pool_mlp)
3	0.164637	discard	[model] pool_mlp: raise dropout 0.1->0.3 to curb overfit on 266-dim kmer4 (framework=pool_mlp)
4	0.244657	keep	[model] pool_mlp: LayerNorm stem residual + pooled 2C vector before MLP to stabilize high-dim 266 features (framework=pool_mlp)
5	0.023246	discard	[model] pool_mlp: stem C->WIDTH then conv residual at WIDTH (Layernorm + mean/max pool -> 2*WIDTH) to widen head to framework baseline (framework=pool_mlp)
6	0.050302	discard	[model] pool_mlp: canonical rebuild stem C->WIDTH(384), conv residual at WIDTH, pool 2*WIDTH (framework=pool_mlp)
7	0.012633	discard	[model] pool_mlp: second conv-residual stem block at C before LayerNorm 2C pooling to deepen high-dim 266 motif extraction (framework=pool_mlp)
8	0.224273	discard	[model] pool_mlp: shallow MLP head 3->2 hidden layers (pool 2C=532, dropout 0.1) to reduce over-param on 266-dim regression (framework=pool_mlp)
9	0.266079	keep	[model] pool_mlp: pre-activation residual MLP head (LayerNorm->Linear->ReLU->Dropout + skip) to ease high-dim 266 regression fit (framework=pool_mlp)
10	0.323057	keep	[model] pool_mlp: deepen pre-activation residual MLP head 2->3 blocks (LayerNorm->Linear->ReLU->Dropout + skip) for more capacity on 266-dim regression (framework=pool_mlp)
11	0.359982	keep	[model] pool_mlp: deepen pre-activation residual MLP head 3->4 blocks (LayerNorm->Linear->ReLU->Dropout + skip) for more capacity on 266-dim regression (framework=pool_mlp)
12	0.320491	discard	[model] pool_mlp: deepen pre-activation residual MLP head 4->5 blocks (LayerNorm->Linear->ReLU->Dropout + skip) for more capacity on 266-dim regression (framework=pool_mlp)
13	0.302508	discard	[model] pool_mlp: self-gated (learnable per-channel sigmoid) conv residual on the stem so the net can scale down motif injection for high-dim 266 kmer4 features (framework=pool_mlp)
14	0.292608	discard	[model] pool_mlp: widen conv stem receptive field via dilated middle conv (k7/d1 -> k5/d2 -> k3 proxying ~25nt) for longer-range motif context on 266-dim kmer4 (framework=pool_mlp)
15	0.471957	keep	[model] pool_mlp: widen pre-activation residual MLP head 384->512 (deeper-width head has more capacity on 266-dim regression) (framework=pool_mlp)
16	0.538223	keep	[model] pool_mlp: widen pre-activation residual MLP head 512->768 (WIDTH=HIDDEN_DIM; head capacity continues to drive 266-dim regression) (framework=pool_mlp)
17	0.557148	keep	[model] pool_mlp: widen pre-activation residual MLP head 768->1024 (WIDTH=HIDDEN_DIM; head capacity keeps driving 266-dim regression) (framework=pool_mlp)
18	0.608147	keep	[model] pool_mlp: widen pre-activation residual MLP head 1024->1536 (WIDTH=HIDDEN_DIM; head capacity keeps driving 266-dim regression) (framework=pool_mlp)
19	0.47839	discard	[model] pool_mlp: widen pre-activation residual MLP head 1536->2048 (WIDTH=HIDDEN_DIM; head capacity keeps driving 266-dim regression) (framework=pool_mlp)
20	0.542072	discard	[model] pool_mlp: raise head dropout 0.1->0.15 to regularize the large 1536 residual head (2048 width failed; high-capacity head likely overfits) (framework=pool_mlp)
21	0	crash	[model] pool_mlp: add feed-forward expansion (FF_EXPAND=4) inside each residual MLP block — more head capacity than 1536-wide embedding without the 2048 overfit (framework=pool_mlp) || crash: fold 1 step 00030 | loss: 376.583921 | epoch: 1 | fold_remaining: 54s fold 1 step 00031 | loss: 868.941897 | epoch: 1 | fold_remaining: 54s fold 1 step 00032 | loss: 849.059814 | epoch: 1 | fold_remaining: 54s fold 1 step 00033 | loss: 1097.598654 | epoch: 1 | fold_remaining: 54s fold 1 step 00034 | loss: 1000.360022 | epoch: 1 | fold_remaining: 53s FAIL
22	0.491701	discard	[model] pool_mlp: add global-norm grad clipping (max_grad_norm=1.0) to stabilize the 1536 residual head and curb exploding-loss failures (framework=pool_mlp)
23	0.635291	keep	[model] pool_mlp: intermediate head width 1536->1792 (2048 overfit; 1792 sits between best 1536 and failed 2048 for the 266-dim residual head) (framework=pool_mlp)
24	0.632112	discard	[model] pool_mlp: intermediate head width 1792->1920 (2048 overfit; 1920 sits between best 1792 and failed 2048 for the 266-dim residual head) (framework=pool_mlp)
25	0.383508	discard	[model] pool_mlp: thin bottleneck expansion (FF_EXPAND=2) in each residual block at the proven-stable 1792 width — gentler capacity than 4x (crashed) or 1920/2048 (overfit) (framework=pool_mlp)
26	0.578525	discard	[model] pool_mlp: add std pooling channel feeding 1792 residual head (mean+max+std -> pool_dim 532->798) to enrich the pooled summary for 266-dim regression (framework=pool_mlp)
27	0.567092	discard	[model] pool_mlp: deepen residual MLP head 4->5 blocks at the winning 1792 width (width curve overfits beyond 1792/1920/2048; add capacity via depth instead of more width) (framework=pool_mlp)
28	0.613194	discard	[feature] re-enable struct_dotbracket L3 ( ) . one-hot alongside bpp pairing probability (input_dim 266->269) — the recipe's 'bpp subsumes dotbracket' assumption was never tested
29	0.592122	discard	[model] pool_mlp: bisect head width 1792->1856, midpoint between best 1792 and first-overfit 1920 (0.632) for the 266-dim residual head (framework=pool_mlp)
30	0.608586	discard	[model] pool_mlp: raise AdamW weight decay 1e-4->3e-4 to regularize the large 1792 residual head (2048/1920/1856 width curve overfits) (framework=pool_mlp)
31	0.632497	discard	[model] halve LR 2e-3->1e-3 at the winning 1792 residual head (4 blocks): capacity-sweet-spot head likely overshoots/unstable at default LR (framework=pool_mlp)
32	0.609304	discard	[model] pool_mlp: LayerNorm before the final head Linear so the huge 1792 residual-block activations feed a normalized final projection (LR/weight-decay/dropout/grad-clip and width/depth all exhausted; normalization is the untried stabilizer) (framework=pool_mlp)

33	0.632403	discard	[feature] kmer k=4->3 (input_dim 266->74): the pool_mlp head-width axis (winning 1792, framework=pool_mlp) is 8 consecutive [model] discards deep and every continuation (1920/1856/2048 widths, FF_EXPAND, depth-5, WD/LR/LayerNorm) fails with an overfit signature against the ~266-dim concat recipe, so shrink the encoder instead — k=4 over-parameterizes the 256-channel kmer vocabulary for this small EL regression (only ~2 training-hours of data, 60s/fold budget) and forces the huge residual head to fight noisy sequence-level identity; k=3 cuts kmer 256->64 channels (4^3), giving the 1792-width head a cleaner lower-dim signal to fit and ~64x fewer kmer positions to memorize under the short fixed TIME_BUDGET; single-lever feature change, input_dim read from data, same concat fusion, no head/hyperparam edit (framework=pool_mlp)
34	0.670219	keep	[model] pool_mlp: halve batch size 256->128 to add gradient noise / regularize the huge 1792 residual head against small-EL-data memorization (dropout 0.15, WD 3e-4, LR 1e-3, LayerNorm head, grad-clip, std-pool all failed individually; batch size is the untried regularization lever on the ~266-dim kmer4 recipe) (framework=pool_mlp)
35	0.673402	keep	[model] pool_mlp: halve batch size 128->64 to push the winning small-batch gradient-noise regularization further on the huge 1792 residual head vs small-EL kmer4 data (batch 128 beat 256 0.635->0.670; cross all other reg levers — dropout/WD/LR/LN/grad-clip — failed individually, batch is the only axis that improved) (framework=pool_mlp)
36	0.661909	discard	[model] pool_mlp: halve batch size 64->32 to continue the only winning regularization axis on the huge 1792 residual head (batch is the sole lever that has monotonically improved small-EL kmer4 regression: 256->128->64 gave 0.635->0.670->0.673; dropout/WD/LR/LN/grad-clip and all structural variants failed individually) (framework=pool_mlp)
37	0.673933	discard	[model] pool_mlp: lower EMA decay 0.999->0.99 so the EMA tracks the noisy batch-64 weights responsively (eval materializes from ema.shadow; batch was the sole winning axis, 256->128->64 gave 0.635->0.670->0.673, 32 over-regularized, so tune the eval-EMA instead of adding more gradient noise) (framework=pool_mlp) (below min_delta=0.001 delta=0.000531)
38	0.661854	discard	[model] pool_mlp: lower AdamW beta2 0.999->0.95 (betas=(0.9,0.95)) so the optimizer keeps a short EMA over gradient-variance history and takes noisier, more responsive steps on the huge 1792 residual head — same regularization spirit as the winning tiny-batch axis (256->128->64 gave 0.635->0.670->0.673) but without batch-32's over-regularization collapse (0.661); batch is the only lever that has improved and beta2 is the untried optimizer-side proxy for that gradient noise (framework=pool_mlp)
39	0.670609	discard	[model] pool_mlp: bisect eval-EMA decay 0.999->0.995 at the winning batch-64 setting — EMA 0.99 already moved the right way (0.6739 vs best 0.6734, just under min_delta) and EMA 0.999 (best) lags the noisy batch-64 train; 0.995 is the clean untried midpoint between the too-stale 0.999 and over-responsive 0.99 for the huge 1792 residual head on small-EL kmer4 (framework=pool_mlp)
40	0.670803	discard	[model] pool_mlp: raise head dropout 0.1->0.15 at the winning 1792 wide head + batch-64 config — the small-EL-data overfitting theme that made small batch (256->128->64) the SOLE winning lever predicts more regularization still helps, while the only prior 0.15 test was at the far weaker 1536-wide head with batch 256 (much less gradient noise), so this is a fresh test on the now-stronger head rather than a repeat of that failure (framework=pool_mlp)
41	0.661837	discard	[model] pool_mlp: widen stem conv channels 256->512 on the 266-dim kmer4 recipe — every conv experiment so far (self-gated residual, dilated kernel, canonical stem->WIDTH rebuild x3, FF_EXPAND) modified the residual/gated path or let conv stay at the kmer-vocab-sized 256 bottleneck fixed since the 69-dim stage; 256 is the same magnitude as the kmer4 vocab (4^4=256) so the motif extractor shrinks back to a near-identity bottleneck that must squeeze 266 channels through 256 before pooling 2C=532 into the huge 1792 head; doubling gives the ReLU layer room to represent richer kmer4 motifs before the mean/max pool, aligned with the head-capacity axis that has been the only structural winner here (framework=pool_mlp)
42	0.676318	keep	[model] pool_mlp: lengthen warmup 0.1->0.25 at the winning batch-64 config — every structural and hyperparameter axis (width/depth/EMA/beta2/dropout/LR/WD/grad-clip/stem-conv/FF-expand/std-pool/LN-head) is exhausted around this config, and warmup is the one untried regularizer; batch-64 is noise-limited at 0.673 and a longer warmup holds LR low during the noisy early small-batch steps so the huge 1792 residual head settles before cosine decay, complementing the gradient-noise theme that finally broke 0.635 (framework=pool_mlp)
43	0.675648	discard	[model] pool_mlp: lengthen warmup 0.25->0.4 at the winning batch-64 config — warmup is the one fresh axis that just kept (0.1->0.25, 0.6763) after every structural/EMA/beta2/dropout/LR/stem-conv lever failed around batch-64; 0.4 holds LR low through even more of the noisy early small-batch steps so the huge 1792 residual head settles under full LR only when the gradients stabilize, extending the only winning regularization theme (gradient-noise + warmup) (framework=pool_mlp)
44	0.670408	discard	[model] pool_mlp: bisect warmup 0.25->0.3 at the winning batch-64 config — warmup is the sole axis that recently kept (0.1->0.25 broke 0.6763), and its upper neighbor 0.4 (0.6756) is only ~0.0007 below best while 0.25 sits just above, so the noisy batch-64 curve is flat over warmup and 0.3 is the clean untried midpoint bisection between the two nearly-tied neighbors, holding LR low slightly longer during the noisy early small-batch steps so the huge 1792 residual head settles before cosine decay (framework=pool_mlp)
45	0.635317	discard	[feature] kmer k=4->3 (input_dim 266->74) at the winning batch-64/warmup-0.25 pool_mlp config — the one structural overfit source the huge 1792 head still fights is the dominant kmer4 identity channel (256 of the 266 input); every micro-regularizer (EMA/beta2/dropout/warmup bisection) around batch-64 discards, and the sole prior k=3 test (c558271) ran at the far weaker batch-256/no-warmup config so is unrepeated here; cutting the kmer vocabulary 256->64 channels (4^3) gives the high-capacity 1792 residual head a much cleaner lower-dim signal to fit under the small EL data, single-lever feature change with input_dim read from data and concat fusion kept (framework=pool_mlp)
46	0.678661	keep	[model] pool_mlp: halve LR 2e-3->1e-3 at the winning batch-64/warmup-0.25 config — the only prior LR-1e-3 test (e9d80d0) ran on the pre-batch-wave batch-256 config, before batch 256->128->64 established small-batch gradient-noise as the sole winning axis; batches 64/32, warmup 0.1/0.25/0.3/0.4, EMA 0.99/0.995, beta2 0.95, dropout 0.15, stem-conv 512 have all been tried HERE, but LR itself never moved under the batch-64+warmup-0.25 regime, and lowering LR is the optimizer-side complement to the gradient-noise theme so the huge 1792 residual head settles with smaller, quieter steps rather than more noise (framework=pool_mlp)
47	0.679415	discard	[model] pool_mlp: raise weight decay 1e-4->3e-4 at the winning batch-64/warmup-0.25/LR-1e-3 config — the only prior WD-3e-4 test (5396250) ran at batch-256, before batch 256->128->64 established small-batch gradient noise as the sole winning overfit axis, and LR followed the identical pattern (failed at batch-256 as e9d80d0, then kept here as 54a10a6); weight decay is the one regularizer never re-tested under the now-noisier batch-64 regime, so raise it to pull the 1792-wide residual head harder toward small weights under the extra gradient variance (framework=pool_mlp) (below min_delta=0.001 delta=0.000754)

48	0.680682	keep	[model] pool_mlp: halve LR 1e-3->5e-4 at the winning batch-64/warmup-0.25 config — LR is the freshest winning axis (1e-3 just kept, 0.6787) after batches 256->128->64 established small-batch gradient noise as the only structural winner; the single prior LR-2e-3 test and its 1e-3 keep both landed the huge 1792 residual head at the noise floor, and every other optimizer side-knob (EMA/beta2/dropout/WD/grad-clip) has already failed under this regime, so 5e-4 is the natural lower bisection that pushes the quieter-smaller-steps complement to gradient noise further without the over-regularization that killed batch-32 (framework=pool_mlp)
49	0.679651	discard	[model] pool_mlp: halve LR 5e-4->2.5e-4 at the winning batch-64/warmup-0.25 config — LR is the only axis with consecutive keeps (2e-3->1e-3 kept 0.6787, 1e-3->5e-4 kept 0.6807) after batches 256->128->64 established small-batch gradient noise as the sole structural winner; every other optimizer side-knob (EMA/beta2/dropout/WD/grad-clip) has already failed under this regime, and both LR halvings landed mid-to-high, so 2.5e-4 continues the quieter-smaller-steps complement to gradient noise without batch-32 over-regularization or the freshly-discarded WD-3e-4 (0.6794, below min_delta) (framework=pool_mlp)
50	0.679583	discard	[model] pool_mlp: bisect LR 5e-4->7.5e-4, midpoint between the two consecutive keeps on the winning LR axis (1e-3 kept 0.6787, 5e-4 kept 0.6807) after batches 256->128->64 established small-batch gradient noise as the only structural winner; the lower neighbor 2.5e-4 (0.6797) already discarded just under min_delta so the LR curve peaks around 5e-4 with 1e-3 only 0.002 below, making 7.5e-4 the clean untried upper bisection that asks whether the peak actually sits slightly above 5e-4 — the freshest axis (LR) is the only one with consecutive keeps and is the natural point to probe before abandoning it (framework=pool_mlp)
51	0.672445	discard	[model] pool_mlp: raise eval-EMA decay 0.999->0.9995 at the winning batch-64/warmup-0.25/LR-5e-4 config — the two prior EMA probes both moved DOWN (0.999->0.99 discard, 0.999->0.995 discard) under the 'track the noisy batch-64 weights responsively' theory, the wrong direction against the winning quiet/smoother/smaller-steps theme that small-batch gradient-noise + LR-5e-4 just established; a HIGHER decay is the untested upper complement, giving a longer-horizon smoother weight average to materialize at eval from ema.shadow, and it extends the only two winning axes (batch, LR) whose lower bisections are now both exhausted (2.5e-4 and 7.5e-4 discarded) without touching the over-regularization that killed batch-32 (framework=pool_mlp)
52	0.681682	keep	[model] pool_mlp: raise AdamW beta1 (momentum) 0.9->0.95 at the winning batch-64/warmup-0.25/LR-5e-4 config — LR is the only axis with consecutive keeps (1e-3->5e-4, 0.6807) and its lower bisection 2.5e-4 (0.6797) discarded just under min_delta, confirming the quieter-smoother-smaller-steps theme on the huge 1792 residual head; EMA (0.99/0.995/0.9995), beta2 (0.95), dropout, WD, grad-clip all already failed here, and momentum (beta1) is the one Adam hyperparameter never moved — a higher beta1 integrates a longer first-moment history, smoothing the gradient direction that drives each step into the same quiet-steps regime that just kept, pushing LR-5e-4's aims further than the two discarded LR neighbors without batch-32 over-regularization (framework=pool_mlp)
53	0.682314	discard	[model] pool_mlp: raise AdamW beta1 (momentum) 0.95->0.98 at the winning batch-64/warmup-0.25/LR-5e-4 config — beta1 is the freshest winning axis (0.9->0.95 just kept, 0.6817) after LR's consecutive keeps (1e-3->5e-4) exhausted its lower/upper bisections (2.5e-4, 7.5e-4 both discarded); every other optimizer knob (EMA/beta2/dropout/WD/grad-clip/warmup) has failed here, so momentum is the only untried continuation of the quieter/smoother/smaller-steps theme — a higher beta1 integrates an even longer first-moment history, further smoothing the gradient direction that drives each small-LR step, pushing beta1-0.95's keep further without the over-regularization that killed batch-32 (framework=pool_mlp) (below min_delta=0.001 delta=0.000632)
54	0.666298	discard	[model] add global-norm grad clipping MAX_GRAD_NORM=1.0 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95 config — the theme that just kept is quieter/smoother/smaller-steps (LR 2e-3->1e-3->5e-4 consecutive keeps, then beta1 0.9->0.95 kept 0.6817 while its upper bisection 0.98 narrowly missed min_delta at 0.6823); every other optimizer knob (EMA 0.99/0.995/0.9995, beta2 0.95, dropout, WD, warmup 0.3/0.4, LR 2.5e-4/7.5e-4) has already failed under this regime, and the only prior grad-clip (64f40a9, 0.49) ran at the weak 1536-head pre-batch-wave config so was never tested against the small-batch gradient-noise that batch-64 provides; a mild max_norm=1.0 bounds the extreme tail fluctuations of the noisy batch-64 steps so each update stays confined to the same quiet regime that LR-5e-4/beta1-0.95 established, without removing the batch-64 gradient-noise regularization that is the structural winner (framework=pool_mlp)
55	0.685222	keep	[model] raise AdamW beta2 (variance EMA) 0.999->0.9995 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95 config — the "quieter, smoother, smaller-steps" theme is the only one that has kept consecutively (LR 2e-3->1e-3->5e-4 then beta1 0.9->0.95, 0.6817): its lone beta2 probe (0.95, f271684) went the NOISY direction and failed, exactly the wrong way for this theme; the untried upper complement — a LONGER gradient-variance horizon — damps the variance estimate that divides each Adam step, pushing each update into the same smoother regime that LR-5e-4/beta1-0.95 established, while every other side-knob (EMA 0.99/0.995/0.9995, dropout, WD, warmup 0.3/0.4, LR 2.5e-4/7.5e-4, beta1 0.98, grad-clip) has already failed here (framework=pool_mlp)
56	0.683846	discard	[model] pool_mlp: raise AdamW epsilon 1e-8->1e-4 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95/beta2-0.9995 config — the 'quieter, smoother, smaller-steps' theme keeps holding (LR 2e-3->1e-3->5e-4, beta1 0.9->0.95, beta2 0.999->0.9995, 0.6852), and every knob that would make steps NOISIER or BIGGER has failed while the untried continuation is the denominator floor itself: eps sits under sqrt(v_hat) in Adam's step denominator, so a LARGER eps is a lower bound on the normalized step magnitude — it caps the extreme tail spikes of the noisy batch-64 gradients while leaving the quiet median steps essentially unchanged, pushing the same damp-the-update regime further than beta2-0.9995 kept without the over-regularization that killed batch-32; the only prior grad-clip (fd07b3d, 0.666) failed, but a soft eps floor bounds the tail differently from a hard max-norm cut and eps is the one scalar Adam hyperparameter (alongside betas/LR/WD) never moved here (framework=pool_mlp)
57	0.681792	discard	[model] pool_mlp: bisect batch 64->48, midpoint between the winning batch-64 (0.6852 after LR/beta keeps) and the over-regularized batch-32 (0.6619) — small-batch gradient-noise was the sole structural winner (256->128->64, 0.635->0.670->0.673) and 48 is the untried midpoint that asks whether the batch curve still rises a hair before the 32 collapse; the optimizer-side 'quieter/smoother/smaller-steps' bisections (LR 2.5e-4/7.5e-4, beta1 0.98, epsilon 1e-4) are all exhausted just under min_delta on the 1792 residual head, so the batch axis is the one fresh lever left on this plateau (framework=pool_mlp)
58	0.682474	discard	[model] pool_mlp: lower AdamW weight decay 1e-4->1e-5 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95/beta2-0.9995 config — the 'quieter/smoother/smaller-steps' theme has kept consecutively (LR 5e-4, beta1 0.95, beta2 0.9995) while every Adam scalar neighbor (LR 2.5e-4/7.5e-4, beta1 0.98, epsilon 1e-4, EMA 0.99/0.995/0.9995, warmup 0.3/0.4, grad-clip) has failed around this plateau; the only weight-decay probes ever run were RAISES (3e-4 at batch-256 5396250, 3e-4 at batch-64 93e3e62, both discards) — and the campaign has repeatedly rewarded the UNTRIED COMPLEMENT of a failed direction (beta2's upper bisection 0.999->0.9995 kept right after the noisy lower 0.95 failed), so the opposite of the failed WD-raise is the untested lever: a lower decay keeps the L2 pull near-zero so the small-LR quiet steps move freely without the decay term dragging the 1792-wide residual head toward tiny weights under the batch-64 gradient variance (framework=pool_mlp)

59	0.68092	discard	[model] pool_mlp: raise AdamW beta2 (variance EMA) 0.9995->0.9997 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95 config — the 'quieter, smoother, smaller-steps' theme is the only axis with consecutive keeps (LR 2e-3->1e-3->5e-4, beta1 0.9->0.95, beta2 0.999->0.9995, 0.6852), and beta2's lone upper probe (0.999->0.9995) just kept while its noisy lower 0.95 and every other Adam scalar neighbor (epsilon 1e-4, WD 1e-5, EMA 0.99/0.995/0.9995, warmup 0.3/0.4, LR 2.5e-4/7.5e-4, beta1 0.98) has failed; the untried continuation is to push beta2's longer gradient-variance horizon one notch further — an even longer EMA over v damps the normalized step denominator further, pushing each update into the same smoother regime that beta2-0.9995 just established, while every knob that makes steps NOISIER or BIGGER has already failed here and a 0.9997 sits safely above the two tested keeps without drifting so close to 1.0 that v becomes stale (framework=pool_mlp)
60	0.685687	crash	[model] pool_mlp: enable AMSGrad (amsgrad=True) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95/beta2-0.9995 config — the 'quieter, smoother, smaller-steps' theme has kept consecutively (LR 5e-4, beta1 0.95, beta2 0.999->0.9995, 0.6852) and every Adam scalar (LR 2.5e-4/7.5e-4, beta1 0.98, beta2 0.9997/0.95, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, warmup 0.3/0.4, grad-clip, batch-48) now fails just under min_delta on this plateau; the one Adam knob never moved is the optimizer variant itself — AMSGrad replaces AdamW's EMA of v with a running MAX of the squared-gradient EMA, so the normalized step denominator v_hat can never decrease and each effective step only ever shrinks, damping the extreme tail spikes of the noisy batch-64 gradients in exactly the same damp-the-update direction that beta2-0.9995 kept while leaving the quiet median steps at the proven LR-5e-4 magnitude (framework=pool_mlp) (below min_delta=0.001 delta=0.000465) (train_exit_nonzero_or_bad_metric) || crash: NameError: name 'CONV_CHANNELS' is not defined | File "/gpool/home/yanyilin/BioClaw/RNAClaw_v1/worktrees/20260821_004207_Reg_EL_HEK/Reg_EL_HEK/train.py", line 54, in PooledMLP
61	0.684914	discard	[model] pool_mlp: enable AMSGrad (amsgrad=True, AMSGRAD const) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95/beta2-0.9995 config — the 'quieter, smoother, smaller-steps' theme has kept consecutively (LR 5e-4, beta1 0.95, beta2 0.999->0.9995, 0.6852) and every Adam scalar (LR 2.5e-4/7.5e-4, beta1 0.98, beta2 0.9997/0.95, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, warmup 0.3/0.4, grad-clip, batch-48) fails just under min_delta on this plateau; the prior AMSGrad probe (8f9a84e) crashed on a NameError (undefined CONV_CHANNELS) before ever training, so the variant itself remains untested — AMSGrad replaces AdamW's EMA of v with a running MAX of the squared-gradient EMA so the normalized step denominator v_hat can never decrease and each effective step only ever shrinks, damping the extreme tail spikes of the noisy batch-64 gradients in the same damp-the-update direction that beta2-0.9995 kept, via a clean edit introducing a single AMSGRAD constant with no undefined symbol references (framework=pool_mlp)
62	0.656761	discard	[model] pool_mlp: ReZero/LayerScale residual — learnable per-channel scale (init 0) on each _ResBlock's residual branch at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.95/beta2-0.9995 config (framework=pool_mlp)
63	0.687623	keep	[model] pool_mlp: raise AdamW beta1 (momentum) 0.95->0.99 at the winning batch-64/warmup-0.25/LR-5e-4/beta2-0.9995 config — beta1 is the freshest optimizer axis: 0.9->0.95 kept (0.6817) and its upper bisection 0.95->0.98 was still RISING (0.6823, discarded only ~min_delta short) when the beta2-0.9995 keep (0.6852) jumped the bar, so the momentum curve never got its continued upper probe; a longer first-moment history integrates the gradient direction over more steps and drives each small-LR update into the same smoother/quieter regime that has kept consecutively (LR 2e-3->1e-3->5e-4, beta1 0.95, beta2 0.9995), pushing the theme one notch further than 0.98 (whose trend was still up) while every other Adam scalar neighbor (beta2 0.9997, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, batch 48, AMSGrad, ReZero) has already failed just under min_delta (framework=pool_mlp)
64	0.678204	discard	[model] pool_mlp: raise AdamW beta1 (momentum) 0.99->0.995 at the winning batch-64/warmup-0.25/LR-5e-4/beta2-0.9995 config — momentum is the freshest winning axis (0.9->0.95 kept 0.6817; 0.95->0.98 was still RISING 0.6823 when discarded just under min_delta, then 0.95->0.99 kept 0.6876), so its upper curve has never been exhausted and 0.995 is the clean untried upper bisection; a longer first-moment history integrates the gradient direction over even more steps, driving each small-LR update into the same smoother/quieter regime that has kept consecutively (LR 2e-3->1e-3->5e-4, beta1 0.95, beta2 0.9995) while every other Adam scalar neighbor (beta2 0.9997, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, batch 48, AMSGrad, ReZero) has already failed just under min_delta (framework=pool_mlp)
65	0.688701	keep	[model] pool_mlp: swap activation ReLU->GELU across stem conv residual and residual MLP head at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995 config — the 'quieter, smoother, smaller-steps' theme has kept consecutively (LR 2e-3->1e-3->5e-4, beta1 0.9->0.95->0.99, beta2 0.999->0.9995, 0.6876) and every Adam scalar/optimizer knob around this plateau (beta2 0.9997, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, batch 48/32, AMSGrad, ReZero, grad-clip) has already failed just under min_delta, while the one element the campaign has NEVER varied is the activation itself — the model's only nonlinearity is ReLU (stem conv residual + 4 residual MLP blocks), which hard-zeroes gradient on ~half the units at every step and adds a sharp kink whose derivative discontinuity injects noise exactly opposite to the quiet-steps regime every keep chased; GELU is smooth (no dead units, continuous derivative) so each small-LR step moves every channel into the same smoother/quieter gradient regime the optimizer scalars just established, ~0 extra FLOPs (O(W) elementwise vs the O(W^2) Linears) so the 60s/fold step count is unchanged; single fresh lever, zero capacity/LR/batch/schedule change, skeleton unchanged, recipe unchanged (framework=pool_mlp)
66	0.688579	discard	[model] lower block/head dropout 0.1->0.05 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995 + GELU config — the campaign has only EVER RAISED dropout (0.1->0.15 at the 1536 head 5a06410 and at the batch-64 1792 head 694dd18, both discards with the noisy/overfit signature of the pre-GELU ReLU regime), so the untried COMPLEMENT (lower) is the fresh lever: the pattern that keeps is the failed direction's complement (beta2 0.999->0.9995 kept right after the noisy lower 0.95 failed; GELU just kept by smoothing the only nonlinearity), and a lower dropout lets the now-smoother GELU-saturated activations and quiet small-LR/beta1-0.99/beta2-0.9995 gradients propagate into the huge 1792 residual head without random zeroing under the tiny 60s/fold budget, ~0 FLOPs, single-lever, skeleton and recipe unchanged (framework=pool_mlp)

67	0.684701	discard	[feature] re-enable struct_dotbracket L3 dot-bracket one-hot alongside struct_bpp (input_dim 266->269, concat, kmer4) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU pool_mlp config (framework=pool_mlp) — every recent [model] lever on the pool_mlp plateau (activation GELU kept 0.6887, then dropout 0.1->0.05 discard 0.688579, beta1 0.995, beta2 0.9997, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, warmup 0.3/0.4, LR 2.5e-4/7.5e-4, batch 48, AMSGrad, ReZero, grad-clip) is exhausted just under min_delta on the huge 1792 residual head, and the recipe has been frozen as concat+kmer4 since the campaign moved off the early weak configs — struct_dotbracket was the only feature ever toggled, and its sole prior test (01cbae9, 0.613) ran at the pre-batch-wave 1536-head era before batch-64 gradient-noise and the optimizer/quiet theme established the current strong regime; bpp (pairing probability) is position-agnostic while dot-bracket gives the per-position MFE structural label a supervised column, so concatenating both gives the 1792 head a slightly richer (269-dim) structural cue with a single whitelist toggle, input_dim read from data, same concat fusion, no head/hyperparam/skeleton change (framework=pool_mlp)
68	0.681288	discard	[model] pool_mlp: smooth the max-pool branch to a softmax-weighted (soft-max) pool at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — the 'quieter/smoother/smaller-steps' theme just kept by swapping the model's only nonlinearity ReLU->GELU (b04c8f1, 0.6887) precisely because GELU is smooth (continuous derivative, no hard zeroing) while every optimizer scalar/EMA/batch/dropout/activation-else knob is now exhausted just under min_delta on this plateau; the ONE remaining non-smooth, hard-selecting operator in the mean+max head is the max-pool itself, which argmax-selects a single position per channel and carries the exact same piecewise derivative-kink/discontinuity that ReLU did, injecting gradient noise opposite to the quiet steps every keep chased; a softmax-weighted pool is the differentiable 'soft max' — it smoothly weights every position (temperature-0 limit recovers max) so each small-LR step flows into every channel with a continuous gradient, the pooling analog of GELU, with pool_dim still 2*C (mean+smooth_max) so the Linear(2*C,1792) projection is untouched, ~0 added FLOPs (O(L*C) vs the O(W^2) head) keeping the 60s/fold step count, and skeleton/recipe unchanged; single fresh lever, zero capacity/LR/batch/schedule change (framework=pool_mlp)
69	0.678583	discard	[model] pool_mlp: push the winning smooth-activation axis one notch further, GELU->Mish (x*tanh(softplus(x))) across stem conv residual and residual MLP head at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995 config — GELU was the FIRST activation change the campaign ever made and it KEPT (b04c8f1, 0.6887), landing the 'quieter, smoother, smaller-steps' theme on the one model element never varied before, after every optimizer scalar/EMA/batch/depth/width/pool lever around it had failed just under min_delta; GELU is smooth (continuous derivative, no dead units) but its positive tail GROWS ~linearly unbounded with the input, which still lets large activations inject the exact noise the quiet regime avoids; Mish is the smooth-activation sibling never tried (SiLU/Mish family untouched) and is strictly smoother — same continuous derivative, no hard zeroing, but its positive tail SATURATES (bounded) so each small-LR step keeps every channel in the same quiet/small-magnitude gradient regime that GELU established while dampening the extreme positive activations the head's huge 1792 width can produce; ~0 extra FLOPs (O(W) elementwise vs the O(W^2) Linears) so the 60s/fold step count is unchanged; single fresh lever on the one axis that just kept, zero capacity/LR/batch/schedule/skeleton change, recipe + framework=pool_mlp unchanged
70	0.688347	discard	[model] pool_mlp: raise block/head dropout 0.1->0.15 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — complement of the just-tested LOWER (52e0d47, 0.1->0.05) whose 0.688579 basically TIED the GELU best (0.688701) instead of beating it, showing the dropout curve around 0.1 is at its ceiling and the looser direction is a dead end; the campaign's most productive keep-pattern is 'the untried complement of a failed direction' (beta2 0.999->0.9995 kept right after the noisy lower 0.95 failed; GELU kept as the never-varied element), and the only prior 0.15 test (694dd18) ran in the pre-GELU noisy-ReLU regime where more regularization fought dead-unit noise — under the now-smoother GELU + quiet small-LR/beta1-0.99/beta2-0.9995 gradients, slightly MORE dropout lets the huge 1792 residual head and 266-dim stem shed the near-tied loose endpoint and nudge the 'quieter/smoother/smaller-steps' theme one notch further; ~0 FLOPs, single fresh lever, skeleton and recipe unchanged (framework=pool_mlp)
71	0.640173	discard	[model] pool_mlp: switch AdamW->NAdam (Nesterov momentum) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995 config - the 'quieter, smoother, smaller-steps' theme has kept consecutively (LR 2e-3->1e-3->5e-4, beta1 0.9->0.95->0.99, beta2 0.999->0.9995, 0.6876, then GELU 0.6887) and every Adam scalar around this plateau is now exhausted (beta2 0.9997, eps 1e-4, WD 1e-5/3e-4, EMA 0.99/0.995/0.9995, batch 48/32, LR 2.5e-4/7.5e-4, beta1 0.995 all failed just under min_delta), and the only Adam optimizer-VARIANT ever tried is AMSGrad (0.684914, discard): NAdam is the untried variant that directly extends the exact axis that has kept winning most persistently - first-moment history (beta1 0.9->0.95->0.99 consecutive keeps) - by adding Nesterov lookahead, which integrates an even longer effective gradient-direction history and smooths each small-LR update into the same quieter regime the scalar bisections established, holding LR/betas/WD/EMA/schedule/batch/activation(GELU)/skeleton/recipe all fixed; a single optimizer constructor swap (NADAM const guards the AdamW path), ~0 FLOPs, no capacity or hyperparameter change (framework=pool_mlp
72	0.689043	discard	[model] pool_mlp: swap regression MSE->SmoothL1 (Huber) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — the 'quieter, smoother, smaller-steps' theme has kept consecutively (LR 2e-3->1e-3->5e-4, beta1 0.9->0.95->0.99, beta2 0.999->0.9995, then activation GELU, 0.6887) and every optimizer/structural/activation/feature lever since (dropout 0.05/0.15, soft-max pool, Mish, NAdam, beta1 0.995, ReZero, AMSGrad, kmer k=3, struct_dotbracket, rna_fm-class L4) has failed just under min_delta; the regression LOSS is the one element this branch has NEVER varied — MSE is the only term that grows the step quadratically: its gradient is proportional to the residual, so a large-residual outlier example injects an out-sized squared-gradient spike into the noisy batch-64 update, exactly the noise the quiet regime every keep chased avoids; SmoothL1/Huber caps the gradient magnitude at 1 for |residual|>1 (beta=1), so the huge 1792 residual head-optimal small-LR steps are damped on outlier tails while staying identical (MSE-like) for small inlier residuals, a soft tail-regularizer on the loss gradient that complements the now-smooth GELU activations and quiet Adam scalars, single fresh lever, ~0 extra FLOPs, LR/betas/WD/EMA/schedule/batch/skeleton/recipe all unchanged (framework=pool_mlp) (below min_delta=0.001 delta=0.000342)

73	0.68789	discard	[model] pool_mlp: tighten SmoothL1 (Huber) beta 1.0->0.5 at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — the 'quieter, smoother, smaller-gradients' theme has kept consecutively (LR/beta1/beta2 scalars then activation GELU, 0.6887) and every lever since on this plateau (dropout 0.05/0.15, soft-max pool, Mish, NAdam, beta1 0.995, ReZero, AMSGrad, struct_dotbracket) has failed just under min_delta, with the lone exception being the regression LOSS swap MSE->SmoothL1 (9ea3fb8) which registered the campaign's CLOSEST miss to date (+0.000342, final 0.689043): the only element ever varied on the loss axis, and it transitioned the quadratic outlier-tail gradient of MSE into a constant-magnitude (|grad|=1) linear tail past |residual|>1 — exactly the damp-the-noise direction every keep chased; the untried continuation on that same freshest axis is the Huber threshold beta itself, and LOWERING beta 1.0->0.5 shrinks the quadratic inlier region so the wide-gradually-growing gradient kicks in over a narrower residual band and a larger share of examples feed the capped |grad|=1 linear tail, pushing the same damp-gradient-spike theme one notch further under the now-smooth GELU activations; single fresh lever on the closest-miss axis, ~0 extra FLOPs, LR/betas/WD/EMA/schedule/batch/skeleton/recipe all unchanged (framework=pool_mlp)
74	0.683087	discard	[model] pool_mlp: widen stem conv intermediate channels 256->384 (CONV_CHANNELS const) under the winning smooth-GELU/quiet regime — the motif-extractor conv is the one structural bridge the campaign has NEVER moved since the 266-dim kmer4 era: its intermediate is hardcoded at 256, EXACTLY the kmer4 vocabulary size (4^4), so the two GELU conv layers form a near-identity bottleneck that must squeeze the 266-dim sequence down to 256 channels before mean/max-pooling 532 into the huge 1792 residual head; the lone prior probe (f51491f, 256->512, 0.6618) ran in the pre-GELU noisy-ReLU era before the 'quieter/smoother/smaller-steps' theme (LR 2e-3->5e-4, beta1->0.99, beta2->0.9995, then activation GELU, 0.6887) established the current strong regime — under smooth continuous GELU throughout, more conv room lets the motif extractor represent richer kmer4 patterns before pooling, extending the head-capacity axis that has been the only structural winner; a clean bisection to 384 (above the near-identity 256, below the failed 512), ~0 extra FLOPs at the O(L) conv scale vs the O(W^2) 1792 head so the 60s/fold step count is unchanged, single fresh structural lever, LR/betas/WD/EMA/schedule/batch/depth/recipe all fixed (framework=pool_mlp)
75	0.687731	discard	[model] pool_mlp: switch regression loss MSE->log-cosh (smooth log(cosh(r))) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — the 'quieter, smoother, smaller-steps' theme has kept consecutively (LR 2e-3->5e-4, beta1->0.99, beta2->0.9995, then activation GELU, 0.6887) and the campaign's CLOSEST miss to date was the loss swap MSE->SmoothL1/Huber (9ea3fb8, +0.000342, final 0.689043): Huber capped the outlier-tail gradient at |grad|=1 but introduced a hard kink at |r|=beta (derivative discontinuity) — the exact class of non-smoothness the campaign just learned to avoid when ReLU->GELU kept precisely because GELU is smooth/continuous-derivative; log-cosh is the smooth ANALOG of that closest miss, with gradient tanh(r) that is MSE-like for small residuals yet saturates continuously toward +-1 on large residuals — same damp-the-outlier-noise tail-cap as the Huber keep-miss but no kink anywhere, so every small-LR quiet step flows into a fully continuous gradient (the 'smoother' continuation of the single closest-miss axis), ~0 FLOPs (elementwise O(B) vs the O(W^2) 1792 head) keeping the 60s/fold step count, single fresh lever, LR/betas/WD/EMA/schedule/batch/skeleton/recipe all unchanged (framework=pool_mlp)
76	0.687622	discard	[model] pool_mlp: raise Huber/SmoothL1 loss beta 1.0->2.0 (LOSS_BETA bisection toward MSE) at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — the 'quieter/smoother/smaller-gradients' theme has kept consecutively (LR scalars, beta1->0.99, beta2->0.9995, then activation GELU, 0.6887) and the single closest miss of the campaign was the regression LOSS swap MSE->SmoothL1 beta=1 (9ea3fb8, +0.000342, final 0.689043): Huber caps the quadratic outlier-tail gradient at |grad|=1 past |r|>beta, damping the noisy batch-64 steps exactly as every keep chased; the only lever ever varied on the loss axis after that closest miss was LOWERING beta (2be0223, 1.0->0.5, discard down to 0.6879), so the untried COMPLEMENT is RAISING beta — beta larger than 1 widens the MSE-like quadratic inlier region and defers the capped linear tail, interpolating smoothly between the closest-miss SmoothL1(beta=1) and plain MSE (the running best), a clean bisection on the coldest-miss axis rather than a repeated failure; single fresh lever, ~0 extra FLOPs (elementwise O(B) vs the O(W^2) 1792 head) keeping the 60s/fold step count, LR/betas/WD/EMA/schedule/batch/activation/skeleton/recipe all unchanged (framework=pool_mlp)
77	-0.005133	keep	[model] framework=cls_mlp framework_base: swap the sequence-level head from pool_mlp to cls_mlp — replace the mean+max pooled 2*WIDTH summary with ONE learned (CLS-style) query vector that attends over length (masked cross-attention) to produce a single WIDTH summary fed through the MLP head that inherits the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config (HIDDEN_DIM=1792, loss=MSE, EMA, dropout). The pool_mlp plateau (0.6887) has exhausted every scalar/activation/loss/dropout lever just under min_delta (GELU kept 0.6887, then log-cosh/Huber/softmax-pool/Mish/NAdam/beta1-0.995/dropout 0.05/0.15/CONV-CHANNELS-384 all discarded below threshold), so the orchestrator-injected next_framework drops the mean+max pooling that summarizes length into a fixed 2-vector summary — a CLS-style head is the untried summary mechanism that lets the model learn which positions matter end-to-end (a differentiable learned weighting over all valid positions) rather than hard-pooling every channel equally, giving the huge 1792 head a single learned context vector to fit under the small-EL data; stem maps straight C->WIDTH (conv+LayerNorm), four inherited _ResBlocks run over the sequence, single masked cross-attention over length to pool, head Linear(WIDTH,WIDTH)->GELU->out (pool_dim=WIDTH), no CONV_CHANNELS knob per red lines, features/active.yaml untouched, framework_base first shot
78	0.13897	discard	[model] framework=cls_mlp: run the 4 residual MLP blocks on the pooled WIDTH summary instead of over every sequence position — the cls_mlp base shot (80768fa) scored -0.005 with 3.2GB because it ran four 1792-wide _ResBlocks over (B=64, L<=1024) at HIDDEN_DIM, an O(L*WIDTH^2) per-forward burden that collapses the 60s/fold budget to a handful of steps; the CLS masked cross-attention already squeezes length into ONE learned WIDTH summary, so applying the inherited residual capacity to that (B, WIDTH) summary (the same MLP-on-summary shape that drove pool_mlp's 0.6887 wins) keeps the framework's learned-summary pooling while cutting sequence-path FLOPs to O(L*WIDTH) so training actually progresses under the fixed budget; pool_dim=WIDTH contract and cls_mlp learned-query pooling (q/k/v + -inf masked softmax) both unchanged, zero hyperparam/recipe change, framework=cls_mlp

79	0	crash	[model] framework=cls_mlp: multi-head (8) learned-query cross-attention CLS pooling at the winning batch-64/warmup-0.25/LR-5e-4/beta1-0.99/beta2-0.9995/GELU config — the cls_mlp framework was just injected after pool_mlp's whole plateau (0.6887) exhausted; its base shot (80768fa) ran the four inherited 1792-wide _ResBlocks over every sequence position, an O(L*WIDTH^2) burden that collapsed the 60s/fold budget (3.2GB, -0.005), and the follow-up (34e8c2d) moved those blocks to the pooled WIDTH summary (0.139 discard). The defining, never-varied element of this fresh head is its single-head learned-query cross-attention: ONE query runs one softmax over up to ~1024 noisy small-EL positions, so the single WIDTH summary inherits a near-uniform/degenerate position-weighting and massively underfits next to pool_mlp's 2*WIDTH mean+max summary (0.6887); 8 heads give 8 independent learned position-weightings averaged into the one WIDTH vector, enriching the CLS summary the 1792 head fits with ~0 added FLOPs at the O(L*WIDTH^2)-dominant head scale, keeping pool_dim=WIDTH and the cls_mlp learned-summary pooling contract intact, restoring the blocks-on-summary path for tractability, single_lever architecture change, recipe unchanged, framework_base of the multi-head variant (framework=cls_mlp) || crash: RuntimeError: Subtraction, the `-` operator, with a bool tensor is not supported. If you are trying to invert a mask, use the `~` or `logical_not()` operator instead. | File "/gpool/home/yanyilin/anaconda3/envs/myenv3.11/lib/python3.11/site-packages/torch/_tensor.py", line 1098, in __rsub__
80	0	crash	[model] framework=cls_mlp: multi-head (8) learned-query cross-attention CLS pooling with correct bool masking — the base cls_mlp shot (80768fa) ran four inherited 1792-wide _ResBlocks over every sequence position, an O(L*WIDTH^2) per-forward burden that collapsed the 60s/fold budget (3.2GB, -0.005), and the prior multi-head attempt (1899ebd) crashed before training on a bool-tensor subtraction in the mask (`1 - mask` on a bool tensor); this clean re-edit uses `~pos_mask` (logical_not) for the pad inversion plus `masked_fill` so the -inf pad flows into softmax with no bool arithmetic, and 8 heads give 8 independent learned position-weightings over up to ~1024 noisy small-EL positions averaged into the one WIDTH summary the 1792 head fits (enriching the single-query degenerate near-uniform weighting that underfit pool's 2*WIDTH mean+max), keeping pool_dim=WIDTH, blocks-on-sequence path, and the cls_mlp learned-summary contract (framework=cls_mlp) || crash: RuntimeError: mat1 and mat2 shapes cannot be multiplied (512x224 and 1792x1792) | File "/gpool/home/yanyilin/anaconda3/envs/myenv3.11/lib/python3.11/site-packages/torch/nn/modules/linear.py", line 125, in forward
81	0.14973	keep	[model] framework=cls_mlp: run the 4 inherited WIDTH residual blocks on the pooled CLS summary instead of over every sequence position, single-head (not multi-head) — the checked-out cls_mlp base shot (80768fa) applies four 1792-wide _ResBlocks to every one of up to ~1024 positions, an O(L*WIDTH^2) per-forward burden at the winning HIDDEN_DIM=1792 that collapses the 60s/fold budget to a handful of steps (-0.005, massive underfit) and matches the O(L*W^2) shape-mismatch crash signature the last multi-head attempt hit (head-dim 224 reshaped blocks feeding Linear(1792)); the CLS masked cross-attention (single query, ~pos_mask -> -inf masked_fill -> softmax, correct bool masking, no head_split reshape of q/k/v so no 224/1792 mismatch) already squeezes length into ONE learned WIDTH summary, so applying the inherited residual capacity to that (B, WIDTH) vector (the exact MLP-on-summary shape that drove pool_mlp's 0.6887 wins) keeps the cls_mlp learned-query pooling contract (pool_dim=WIDTH) while cutting the sequence path to O(L*WIDTH) so training actually progresses under the fixed budget; single fresh tail on the framework_base, no multi-head (which was the crash source), stem C->WIDTH (conv+LayerNorm) unchanged, hyperparams/EMA/loss/recipe unchanged, features/active.yaml untouched (framework=cls_mlp)
82	0.098533	keep	[model] framework=cls_mlp: sharpen the CLS cross-attention by raising the logit scale from the standard d^-0.5 (0.0236 at HIDDEN_DIM=1792) to d^+0.25 (~6.5), ~280x sharper position weighting — the cls_mlp head's defining, never-varied element is its single learned query, and at the inherited 1792 width the textbook 1/sqrt(d) scaling flattens the softmax over up to ~1024 small-EL positions into a near-uniform weighting (the exact "degenerate/near-uniform summary" the framework_base commit 80768fa itself flagged), so the one WIDTH summary degenerates toward a plain mean of the shallow conv-stem keys with no position selectivity — the very signal pool_mlp's max-branch captured; score on the current blocks-on-summary build (a1c9b1d) collapsed to 0.150 vs pool_mlp's mean+max 0.6887; sharpening lets the learned query actually pick the informative positions end-to-end at ~0 extra FLOPs (O(L*W) elementwise vs the O(W^2) head) so the 60s/fold step count is unchanged, single fresh lever on the framework's core mechanism, single-head kept (multi-head was the crash source), pool_dim=WIDTH contract and stem C->WIDTH (conv+LayerNorm) unchanged, hyperparams/EMA/loss/recipe untouched, features/active.yaml untouched (framework=cls_mlp)

Direct mode on T6 task

Autonomous Modeling Stage: 37 experiments attempted, keep rate 32.4%.
Finalization Stage: Test Pearson = 0.505

Trial	val pearson	status	RNAOmni Improvement
1	0.600919	keep	[feature] struct_thermo: add MFE/ensemble thermodynamic structure signal to evo2 concat (input_dim 4096->4099), cause=feature
2	0.604103	keep	[feature] struct_dotbracket: add MFE dot-bracket per-pos one-hot to evo2+thermo concat (input_dim 4099->4102), cause=feature
3	0.667387	keep	[feature] struct_bpp: enable base-pair probability + positional entropy from PF in evo2+thermo+dotbracket concat (input_dim 4102->4104), cause=feature
4	0.745481	keep	[feature] rna_fm: enable RNA-FM reps (dim 640) in evo2+thermo+dotbracket+bpp concat (input_dim 4104->4744), cause=feature
5	0.686113	discard	[model] baseline: raise DROPOUT 0.1->0.2 to regularize 4744-dim head from 5 stacked channels (val test-overfit), cause=knob
6	0.782194	keep	[feature] rinalmo: enable RiNALMo micro reps (dim 480) in evo2+thermo+dotbracket+bpp+rna_fm concat (input_dim 4744->5224), cause=feature
7	0.734923	discard	[feature] kmer: enable position-wise k-mer one-hot (k=3, C=64) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo concat (input_dim 5224->5288), cause=feature
8	0.794432	keep	[model] framework=attn_pool_mlp framework_base cause=head: swap Predictor PooledMLP->AttnPoolMLP (stem->attn-sum+max pool, concat 2*WIDTH->MLP), inherit HIDDEN_DIM/LR/batch/dropout/EMA
9	0.774744	keep	[model] framework=gated_pool_mlp framework_base cause=head: swap Predictor AttnPoolMLP->GatedPoolMLP (stem->per-position softmax gate sum, pool_dim WIDTH->MLP), inherit HIDDEN_DIM/LR/batch/dropout/EMA
10	0.673758	discard	[feature] rna_fm: tune repr_layer 12->8 (mid-layer RNA-FM reps, dim 640 unchanged) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo concat, cause=feature
11	0.786544	discard	[feature] onehot: enable L0 position-wise one-hot (dim 4) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo concat (input_dim 5224->5228), cause=feature
12	0.797256	keep	[feature] length_frac: enable position-normalized sequence length feature (dim 1) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo concat (input_dim 5224->5225), cause=feature
13	0.800515	keep	[model] attn_pool_mlp: shrink HIDDEN_DIM 384->192 to cut the 5225-dim concat stem+net cost and fit more steps in the fixed TIME_BUDGET (5-fold), cause=knob
14	0.792515	discard	[model] framework=attn_pool_mlp: raise DROPOUT 0.1->0.2 to regularize the high-dim 5225 concat stem when many steps fit in the fixed TIME_BUDGET (5-fold), cause=knob
15	0.800499	discard	[feature] rna_fm: tune repr_layer 12->4 (mid/lower-layer RNA-FM reps, dim 640 unchanged; layer 8 already discarded, probe distinct layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature
16	0.791656	discard	[model] framework=attn_pool_mlp: halve BATCH_SIZE 256->128 to get more weight updates in the fixed TIME_BUDGET with the smaller 192-dim head (more steps just kept via HIDDEN_DIM shrink), cause=knob
17	0.741047	keep	[model] framework=pool_mlp framework_base cause=head: swap Predictor AttnPoolMLP->PooledMLP (Conv1d motif extractor + residual, mean+max pool -> MLP), inherit HIDDEN_DIM=192/LR/batch/dropout/EMA
18	0.667617	discard	[feature] rna_fm: tune repr_layer 12->11 (final transformer block just below default; layers 8 and 4 already discarded, probe untried high layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat (dim 640 unchanged), cause=feature
19	0.791898	discard	[model] framework=attn_pool_mlp: shrink HIDDEN_DIM 192->128 to fit even more steps in the fixed TIME_BUDGET for the 5225-dim concat (prior 384->192 kept), cause=knob
20	0.804466	keep	[model] framework=attn_pool_mlp: halve LR 2e-3->1e-3 for more stable optimization of the 192-dim head on the 5225-dim concat stem (untried knob for this head), cause=knob
21	0.781381	discard	[feature] rna_fm: tune repr_layer 12->2 (lower-layer RNA-FM reps, dim 640 unchanged; layers 8,4,11 already discarded, probe distinct untried lower layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature
22	0.815041	keep	[model] framework=attn_pool_mlp: halve LR 1e-3->5e-4 for even more stable optimization of the 192-dim head on the 5225-dim concat stem (prior 2e-3->1e-3 kept; probe next lower step), cause=knob
23	0.807112	discard	[model] framework=attn_pool_mlp: raise DROPOUT 0.1->0.15 to lightly regularize the 5225-dim concat stem+192-dim head at current LR 5e-4 (dropout untried on this exact config; prior 0.1->0.2 discard was on the old HIDDEN_DIM=384+LR 2e-3 config, so probe a modest step), cause=knob
24	0.754113	discard	[feature] rna_fm: tune repr_layer 12->6 (lower-mid-layer RNA-FM reps, dim 640 unchanged; layers 8,4,11,2 already discarded, probe distinct untried lower-mid layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature
25	0.795612	discard	[model] framework=attn_pool_mlp: raise EMA_DECAY 0.999->0.9995 for smoother eval weights on the 5225-dim concat stem+192-dim head (untried knob for this head; LR 5e-4 kept), cause=knob
26	0.74875	discard	[feature] rna_fm: tune repr_layer 12->0 (raw input embedding layer, dim 640 unchanged; blocks 8,4,11,2,6 already discarded, probe the untried layer-0 nucleotide-identity embedding before any transformer mixing) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature
27	0.807643	discard	[model] framework=attn_pool_mlp: halve LR 5e-4->2.5e-4 for even more stable optimization of the 192-dim head on the 5225-dim concat stem (prior 2e-3->1e-3 and 1e-3->5e-4 both kept; best commit flagged probe next lower step), cause=knob
28	0.811582	discard	[model] framework=attn_pool_mlp: lower EMA_DECAY 0.999->0.99 for less-smoothed eval weights on the 5225-dim concat stem+192-dim head at LR 5e-4 (prior 0.999->0.9995 increase discarded, so probe the untried less-smoothed direction), cause=knob
29	0.697848	discard	[feature] rna_fm: tune repr_layer 12->10 (upper-mid layer RNA-FM reps, dim 640 unchanged; layers 8,4,11,2,6,0 already discarded, probe distinct untried high layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature

30	0.808751	discard	[model] framework=attn_pool_mlp: raise WEIGHT_DECAY 1e-4->1e-3 to L2-regularize the 5225-dim concat stem+192-dim head (distinct untried regularizer; dropout/EMA/batch/LR/HIDDEN alternates all tried and discarded), cause=knob
31	0.725518	discard	[feature] rna_fm: tune repr_layer 12->9 (untried mid-high layer; dim 640 unchanged) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat (layers 8,4,11,2,6,0,10 already discarded), cause=feature
32	0.815554	discard	[model] framework=attn_pool_mlp: switch regression MSE->Huber (smooth_l1 beta=1) to be robust to noisy/outlier reactivity targets on the 5225-dim stem+192-dim head at LR 5e-4 (loss is the only untried actionable knob; feature/other-knob tuning exhausted), cause=knob (below min_delta=0.001 delta=0.000513)
33	0.762664	discard	[feature] rna_fm: tune repr_layer 12->5 (untried lower-mid layer, dim 640 unchanged; layers 8,4,11,2,6,0,10,9 all discarded, probe distinct untried layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature
34	0.792479	discard	[model] framework=attn_pool_mlp: lower WEIGHT_DECAY 1e-4->1e-5 to let the 192-dim head fit the 5225-dim structure-heavy concat with less L2 pull (prior WD probe was up 1e-4->1e-3 and discarded, so probe the untried down direction at LR 5e-4), cause=knob
35	0.792684	discard	[model] framework=attn_pool_mlp: lower DROPOUT 0.1->0.05 on the 192-dim head+5225-dim stem at LR 5e-4 (prior +0.1->0.15 raise was discarded, so probe the untried less-regularized direction for the small head / fixed TIME_BUDGET), cause=knob
36	0.77171	discard	[feature] rna_fm: tune repr_layer 12->7 (untried mid-transformer block, dim 640 unchanged; layers 8,4,11,2,6,0,10,9,5 already discarded, probe distinct untried middle layer) in evo2+thermo+dotbracket+bpp+rna_fm+rinalmo+length_frac concat, cause=feature
37	0.7948	discard	[model] framework=attn_pool_mlp: double BATCH_SIZE 256->512 for larger more stable gradient estimates on the 192-dim head + 5225-dim concat at LR 5e-4 (prior batch knob was only the halve 256->128 discard, so probe the untried increase direction), cause=knob

Figure 5: A demo of the *Direct* and *Thinking* modes of RNAOmni for autonomous modeling of RNA translation efficiency prediction on the HEKEL task, along with the intermediate model performance.