PIGRAM: An Interpretable Patch–Motif Interaction Grammar for Protein–Nucleic-Acid Recognition

Yaru Han
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
yhan459@gatech.edu

Abstract

Protein–nucleic-acid recognition underlies aptamer discovery and nucleic-acid therapeutics, yet computational models often trade mechanistic interpretability for predictive flexibility. Hybrid models risk fallback takeover, where the latent branch dominates prediction and sidelines the explicit pathway. We introduce PIGRAM, which represents proteins as local residue patches and nucleic-acid partners as secondary-structure motifs. It learns a sign-separated grammar over explicit physicochemical and geometric attributes to preserve rare favorable interactions, selectively refining heterogeneous rules into a coarse-to-fine atlas. The frozen grammar remains the primary predictor, while a residual Transformer applies sparse, bounded corrections in regions poorly captured by the grammar. On an exact-sequence-pair-disjoint benchmark, PIGRAM achieves Pearson 0.494 versus 0.511 for an unconstrained Transformer, with lower local Spearman correlation but numerically higher Top10 precision and AUPRC in downstream candidate screening. The atlas exposes signed patch–motif compatibility regimes; motif patterns aligned with supervision priors are interpreted as prior-informed. Wet-lab experiments show that suppressive-rule relief improves GFP binding, whereas favorable-rule disruption weakens NELF binding. These results support interpretable grammar as a predictive scaffold and an experimentally actionable guide for optimizing synthetic nucleic acids.

1 Introduction

Protein–nucleic-acid recognition underlies aptamer discovery [1], targeted biosensing [2], and therapeutic molecular design [3–9]. Single-stranded DNA and RNA aptamers are attractive because they can be selected against diverse protein targets and chemically optimized for downstream applications [10–12].

However, aptamer discovery still relies heavily on iterative screening procedures such as SELEX [13–15], followed by post-selection optimization through truncation, mutagenesis, and affinity tuning [12]. Computational models that predict local protein–nucleic-acid compatibility while exposing the underlying interaction logic could accelerate candidate prioritization and rational molecular refinement.

Recent structure-aware neural models offer a promising route toward this goal [16–23]. Flexible deep interaction models can learn complex biomolecular patterns [24–32], yet their predictions are often difficult to use for experimental design: a high score does not reveal which local protein environment interacts with which nucleic-acid structural motif [33–38], or why. Conversely, interpretable models based on prototypes, discrete templates, or shallow rule systems can expose reusable patterns [39–42], but often lack the capacity to capture the heterogeneous and sparse interaction landscape of protein–nucleic-acid recognition.

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

A natural strategy is to combine an explicit interpretable mechanism with latent neural capacity [43–45]. However, such hybrids introduce a critical failure mode: the latent branch can become the de facto predictor while the explicit pathway remains visible but functionally secondary. We refer to this phenomenon as *fallback takeover*. In sparse biomolecular settings, fallback takeover is especially problematic because common non-interacting pairs can dominate explicit rules, while rare favorable mechanisms may suffer favorable-rule starvation and be delegated to an opaque black-box component [46, 47]. The resulting model appears interpretable, but its explanations are not necessarily responsible for its predictions [48–50].

We propose **PIGRAM**, an interpretable patch–motif grammar model for protein–nucleic-acid recognition. PIGRAM reformulates interaction prediction at an intermediate structural resolution: proteins are decomposed into local residue patches, and nucleic-acid partners are decomposed into secondary-structure motifs. It learns a patch–motif interaction field through an explicit sign-separated grammar over protein-patch and nucleic-acid motif attributes. By separating suppressive and favorable rule families, PIGRAM prevents rare positive compatibility modes from being absorbed by dominant non-binding regimes. It further imposes asymmetric rule geometry, using sharper suppressive kernels for local exclusion and broader favorable kernels for tolerant binding basins. Selective fine refinement then subdivides only internally heterogeneous coarse rule families, producing a compact coarse-to-fine grammar atlas.

To prevent latent capacity from replacing the grammar, PIGRAM trains the coarse-to-fine grammar foundation first and freezes it before introducing a Transformer residual branch [51–54]. The residual branch can add only bounded corrections through a restricted, pair-specific routing mechanism, acting as local rescue for grammar-hard patch–motif pairs rather than as a competing predictor.

In summary, this work introduces a patch–motif formulation for interpretable local protein–nucleic-acid compatibility; a sign-separated coarse-to-fine grammar with asymmetric rule geometry; a frozen-grammar residual rescue mechanism that prevents fallback takeover; and evaluations characterizing the tradeoff between local structural accuracy and an auditable grammar atlas. The contribution is intrinsic interpretability combined with candidate prioritization and experimentally supported directional edits, rather than structural-accuracy superiority over an unconstrained Transformer.

Together, these results position PIGRAM as an interpretable framework for data-driven aptamer modeling, SELEX candidate prioritization, false-positive filtering, and rational molecular refinement [12, 55–60].

2 Related Work

Protein–nucleic-acid interaction modeling. Protein–nucleic-acid recognition has been modeled through residue–nucleotide contact prediction [61–63], interface classification [61, 64, 65], docking-inspired scoring [66–68], and complex-level affinity prediction [69, 70]. These formulations usually operate either at fine residue/contact resolution or at coarse complex-level resolution. Contact-level predictions can be difficult to summarize into reusable design principles, whereas global scores obscure which local protein environments interact with which nucleic-acid structural states. PIGRAM instead models recognition at an intermediate patch–motif resolution, exposing reusable local compatibility patterns rather than only individual contacts or aggregate scores.

Structure-aware neural interaction models. Recent neural interaction models combine learned protein and nucleic-acid representations with MLP heads, bilinear scoring, graph architectures, cross-attention, or Transformer-style modules. Transformer-style models are widely used for protein, molecular, and protein–nucleic-acid representation learning [25–28, 51–54], while graph-based and interface-aware models have been applied to protein binding, drug–protein interactions, and protein–ligand or protein–nucleic-acid interfaces [24, 29–32, 71]. Although these models absorb sequence, structural, and geometric signals, their flexible latent modules are usually not aligned with explicit mechanism discovery, making it difficult to distinguish stable local compatibility principles from shortcut-prone distributed decisions. This limits their use for aptamer optimization, where local explanations are needed.

Interpretable biomolecular models. Interpretable alternatives include prototype-based models, discrete templates, symbolic or semi-symbolic rules, attention explanations, and concept-bottleneck-

style representations. Related work has explored intrinsically interpretable sequence-to-function models, interpretable graph embeddings, motif-based binding-site predictors, and interpretable affinity or drug–target models [39–45]. Other approaches use residue–nucleotide specificity or chemically informed descriptors to expose binding-relevant structure [72]. These methods expose inspectable intermediate structure, but flat prototypes or rigid rule systems may be too weak for sparse and heterogeneous biomolecular interfaces. In protein–nucleic-acid recognition, local compatibility can depend jointly on protein-side chemistry, geometry, topology, and nucleic-acid motif context. PIGRAM addresses this limitation by making the explicit mechanism itself expressive: signed coarse rules define the primary interaction field, while selective fine refinement resolves heterogeneous rule families without uniformly fragmenting the rule space.

Hierarchical and residual specialization. Our work is also related to hierarchical prediction, mixture-of-experts, selective computation, and residual modeling, where submodules specialize to different input regions [73–76]. Generic specialization, however, does not guarantee interpretability. In hybrid interpretable–latent architectures, a flexible latent pathway can become the effective decision function while the explicit pathway remains visible but secondary. PIGRAM avoids this failure mode by first training and freezing an explicit coarse-to-fine grammar foundation, then training a bounded residual Transformer restricted to sparse local correction of grammar-hard patch–motif pairs.

Positioning of PIGRAM. PIGRAM occupies a middle ground between flexible black-box predictors and rigid interpretable systems. Its distinction is the combination of sign-separated patch–motif grammar, asymmetric rule geometry, selective fine refinement, and frozen bounded residual rescue, which allows heterogeneous protein–nucleic-acid interactions to be modeled while keeping the grammar field as the primary interpretable predictive object.

3 Method

3.1 Patch–Motif Representation and Prediction Target

For each protein–nucleic-acid sample, we represent the protein as an ordered set of local residue patches $\mathcal{P} = \{P_i\}_{i=1}^{N_p}$ and the nucleic-acid partner as an ordered set of secondary-structure motifs $\mathcal{M} = \{M_j\}_{j=1}^{N_m}$. PIGRAM predicts a patch–motif interaction matrix $S \in \mathbb{R}^{N_p \times N_m}$, where S_{ij} denotes the predicted compatibility between protein patch P_i and nucleic-acid motif M_j.

The primary supervision signal is a continuous structure-aware patch–motif target $Y^{\mathrm{struct}} \in [0,1]^{N_p \times N_m}$. Binary labels $Y^{\mathrm{bin}} \in \{0,1\}^{N_p \times N_m}$ are used for threshold-based evaluation, and a patch-level target $Y^{\mathrm{patch}} \in \{0,1\}^{N_p}$ provides auxiliary protein-side relevance supervision. Details of patch construction, motif segmentation, and label construction are provided in Appendix A.

3.2 PIGRAM Architecture

PIGRAM models protein–nucleic-acid recognition as a grammar-centered patch–motif interaction problem. As illustrated in Figure 1, the model learns a sign-separated coarse grammar over explicit patch and motif attributes, selectively refines heterogeneous coarse rules into fine subrules, and then applies a bounded residual Transformer as sparse local rescue for grammar-hard pairs.

The grammar foundation produces

$$H_{ij}^{A} = G_{ij}^{\mathrm{coarse}} + G_{ij}^{\mathrm{fine}}, \tag{1}$$

where G_{ij}^{coarse} captures broad suppressive or favorable compatibility rules and G_{ij}^{fine} resolves selected within-rule heterogeneity. After this foundation is trained and frozen, the final prediction is

$$S_{ij}^{\mathrm{final}} = H_{ij}^{A} + r_{ij}\Delta_{ij}^{B}, \tag{2}$$

where Δ_{ij}^{B} is a residual Transformer correction and r_{ij} is a bounded rescue gate. Thus, the Transformer is not a standalone interaction head; it can only provide gated local corrections to the frozen grammar field.

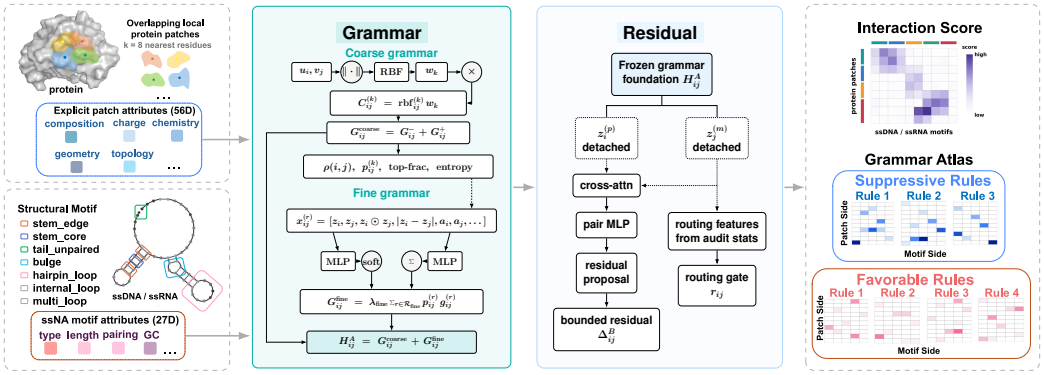


Figure 1: **Overview of PIGRAM.** PIGRAM represents protein–nucleic-acid recognition as an interpretable patch–motif grammar. An explicit grammar foundation provides the primary interaction field, while a bounded residual Transformer supplies constrained local corrections for grammar-hard pairs.

3.3 Patch and Motif Encoding

Each protein is decomposed into overlapping local residue patches, and each nucleic-acid partner is segmented into secondary-structure motifs. For each patch P_i and motif M_j, PIGRAM constructs latent embeddings $\mathbf{z}_i^{(p)}, \mathbf{z}_j^{(m)} \in \mathbb{R}^d$ and explicit attribute vectors $\mathbf{a}_i^{(p)} \in \mathbb{R}^{56}$, $\mathbf{a}_j^{(m)} \in \mathbb{R}^{27}$. The patch encoder summarizes residue identity, local geometry, sequence topology, and center-residue context; the motif encoder summarizes sequence information and secondary-structure context, including motif type, pairing state, edge status, and nucleotide composition. Latent embeddings are used by the fine refiners and residual rescue branch, whereas explicit attributes define the interpretable grammar vocabulary. Detailed encoder parameterization and attribute definitions are provided in Appendix B.

3.4 Sign-Separated Coarse Grammar

The coarse grammar is an attribute-based interaction module that decomposes patch–motif compatibility into suppressive and favorable rule families:

$$G_{ij}^{\text{coarse}} = G_{ij}^{\text{neg}} + G_{ij}^{\text{pos}}. \tag{3}$$

Both branches operate only on explicit patch and motif attributes. For each sign branch $s \in \{\text{neg}, \text{pos}\}$, the attributes are projected into rule-specific vectors $\mathbf{u}_i^{(s,k)}$ and $\mathbf{v}_j^{(s,k)}$. After adding learnable rule anchors and normalizing the rule vectors, each rule computes an RBF compatibility:

$$\text{RBF}_{ij}^{(s,k)} = \exp\left(-\frac{\left\|\mathbf{u}_i^{(s,k)} - \mathbf{v}_j^{(s,k)}\right\|_2^2}{(\sigma_{s,k}^{\text{eff}})^2}\right). \tag{4}$$

Rule magnitudes are sign-constrained so that suppressive rules can only contribute negatively and favorable rules can only contribute positively:

$$w_k^{\text{neg}} = -\,\text{softplus}(\theta_k^{\text{neg}}), \qquad w_k^{\text{pos}} = \text{softplus}(\theta_k^{\text{pos}}). \tag{5}$$

Each rule contribution is

$$C_{ij}^{(s,k)} = \text{RBF}_{ij}^{(s,k)}\, w_k^s, \tag{6}$$

and the total coarse grammar score is

$$G_{ij}^{\text{coarse}} = \sum_{k=1}^{K_{\text{neg}}} C_{ij}^{(\text{neg},k)} + \sum_{k=1}^{K_{\text{pos}}} C_{ij}^{(\text{pos},k)} + \ell_p(\mathbf{a}_i^{(p)}) + \ell_m(\mathbf{a}_j^{(m)}) + b. \tag{7}$$

A key design choice is the use of asymmetric bandwidths. Suppressive rules are assigned narrower admissible bandwidths, encouraging sharp local exclusion zones for unfavorable steric or electrostatic

regimes. Favorable rules are assigned broader bandwidths, allowing compatible patch–motif configurations to tolerate modest geometric or sequence-level variation. This asymmetric rule geometry helps the grammar preserve rare favorable modes without allowing broad non-binding regimes to dominate the interaction field. The exact bandwidth parameterization and schedule are provided in Appendix C.

For downstream fine refinement and residual routing, we summarize rule usage by normalized absolute rule masses:

$$p_{ij}^{(s,k)} = \frac{|C_{ij}^{(s,k)}|}{\sum_{s',k'} |C_{ij}^{(s',k')}| + \varepsilon}. \tag{8}$$

These masses identify which coarse families remain mechanistically mixed and provide grammar-side confidence features for selective fine refinement and bounded residual rescue.

3.5 Selective Fine Grammar

The sign-separated coarse grammar defines compact suppressive and favorable compatibility basins, but individual coarse rules can still mix distinct local mechanisms, a form of within-rule heterogeneity that we call *mechanism masking*. In this case, a coarse rule identifies an interaction family, while different patch geometries, charge environments, or motif contexts remain unresolved within that family.

PIGRAM addresses mechanism masking with a selective fine grammar. Instead of refining every coarse rule uniformly, we audit the trained coarse grammar and refine only rule families that are sufficiently supported and internally heterogeneous. For each selected rule $r \in \mathcal{R}_{\text{fine}}$, a rule-specific fine refiner predicts local fine-subrule refinements from patch–motif latent features, explicit attributes, and coarse-rule statistics. The teacher-derived pseudo-labels are used only as refinement scaffolds, not as fixed biological rule identities; their construction is detailed in Appendix D.

Let $g_{ij}^{(r)}$ denote the rule-local fine contribution for selected coarse rule r. The selective fine grammar aggregates these refinements using the corresponding coarse-rule mass:

$$G_{ij}^{\text{fine}} = \lambda_{\text{fine}} \sum_{r \in \mathcal{R}_{\text{fine}}} p_{ij}^{(r)} g_{ij}^{(r)}. \tag{9}$$

Together with Eq. 1, this preserves coarse-rule interpretability while allowing selected rule families to express finer suppressive, amplifying, or rescue-like submechanisms.

3.6 Budgeted Residual Transformer Rescue

After the coarse-to-fine grammar foundation is trained, all grammar-foundation parameters are frozen. PIGRAM then trains a residual Transformer branch on detached patch and motif embeddings. The branch uses bidirectional cross-attention to propose a raw local correction $\tilde{\Delta}_{ij}^{B}$ for each patch–motif pair.

To prevent unbounded correction magnitude, the residual proposal is clipped as

$$\Delta_{ij}^{B} = \Delta_{\max} \tanh\left(\frac{\tilde{\Delta}_{ij}^{B}}{\Delta_{\max}}\right). \tag{10}$$

The correction is further controlled by a bounded pair-level rescue gate:

$$r_{ij} = r_{\max} \,\text{sigmoid}\left(\text{MLP}_{\text{route}}\left(\mathbf{g}_{ij}\right)\right), \qquad r_{\max} < 1, \tag{11}$$

where $\mathbf{g}_{ij}$ contains grammar-side confidence features.

The final score follows Eq. 2. Unlike a mixture-of-experts fallback model, the Transformer is not a parallel interaction predictor; it can only make bounded, gated corrections to the frozen grammar field. Thus, predictions remain grammar-driven when the route is closed and receive local residual rescue only on grammar-hard patch–motif pairs.

PIGRAM is trained in three stages to enforce this division of labor. First, a pure sign-separated coarse grammar teacher learns stable suppressive and favorable rule families. Second, the coarse-to-fine grammar foundation is trained using teacher-derived fine pseudo-labels, with an auxiliary

Table 1: Baseline comparison on the exact-sequence-pair-disjoint benchmark. P/S denote test Pearson/Spearman correlations. Subscripts denote SD over five seeds.

Model	P	S	AUROC	AUPRC
Sequence-only MLP	$.117_{.002}$	$.105_{.009}$	$.707_{.001}$	$.031_{.003}$
Concat-MLP	$.415_{.007}$	$.280_{.002}$	$.777_{.002}$	$.118_{.005}$
Bilinear	$.396_{.003}$	$.255_{.002}$	$.771_{.007}$	$.102_{.004}$
XGBoost attributes	$.395_{.003}$	$.274_{.002}$	$.745_{.014}$	$.106_{.005}$
ESM-2 + MLP	$.405_{.006}$	$.265_{.014}$	$.723_{.002}$	$.113_{.001}$
Cross-attention	$.448_{.005}$	$.301_{.017}$	$.775_{.006}$	$.113_{.014}$
Transformer	$\mathbf{.511}_{.014}$	$\mathbf{.482}_{.009}$	$\mathbf{.821}_{.004}$	$\mathbf{.157}_{.004}$
Prototype	$.347_{.022}$	$.221_{.020}$	$.695_{.019}$	$.043_{.015}$
Prototype + residual	$.458_{.015}$	$.316_{.007}$	$.789_{.010}$	$.110_{.014}$
PIGRAM	$.494_{.005}$	$.383_{.004}$	$.794_{.002}$	$.150_{.006}$

coarse-grammar loss to ensure that fine refinement does not replace the coarse rule field. Third, the grammar foundation is frozen, and only the residual Transformer and rescue gate are trained with sparsity, correction-size, and grammar-easy no-harm penalties. Full loss definitions are provided in Appendix F; residual route features and route-atlas definitions are provided in Appendix E.

4 Experiments

We evaluate PIGRAM on an exact-sequence-pair-disjoint patch–motif benchmark. RNA and DNA complexes are split independently at the sequence-pair level and then merged into train/validation/test partitions, yielding 3,680/289/259 samples and 3,373/252/253 unique protein–partner pairs. Targets are derived from the structure-aware labeling pipeline in Appendix A.

We assess predictive performance, mechanism resolution, and grammar dominance under sparse pair-disjoint generalization. Pearson and Spearman correlations on the continuous target are primary metrics; AUROC and AUPRC are auxiliary threshold-based metrics. Unless otherwise stated, model comparisons are averaged over five training seeds, with standard deviations shown as subscripts. We additionally retrain Transformer, PIGRAM, and Grammar Foundation over five seeds on a full-protein-sequence-cluster-disjoint split defined by 30% identity and 80% bilateral coverage. PIGRAM achieves Pearson 0.479, versus 0.495 for Transformer and 0.447 for Grammar Foundation. A post-split audit finds no cross-split match at this threshold. This establishes transfer across the stated full-protein sequence clusters, not independence of local domains or three-dimensional interfaces. Split details and pair-clustered bootstrap CIs are given in Appendix A.7.

4.1 Predictive Performance Under Sparsity

We first compare PIGRAM against flexible black-box baselines and simpler interpretable alternatives in Table 1. PIGRAM remains slightly below the unconstrained Transformer on Test Pearson (0.494 vs. 0.511), leaving a gap of 0.017, but consistently outperforms simpler interpretable baselines. The Prototype + residual comparison shows that the gain is not explained merely by adding neural residual capacity to an interpretable scaffold; the explicit patch–motif grammar provides a stronger predictive foundation. The paired difference (PIGRAM minus Transformer) is -0.0171 for Pearson (95% CI $[-0.042, 0.009]$) and -0.0987 for Spearman ($[-0.129, -0.067]$). The Pearson difference is unresolved, which does not establish equivalence; the Spearman interval supports a Transformer advantage in local ranking. PIGRAM's contribution instead combines intrinsic interpretability with candidate prioritization and experimentally supported directional edits. The residual branch is deliberately sparse (mean route 0.038; Table 4), so it cannot freely replace the grammar foundation. Additional spatial and domain-baseline comparisons, together with residual-contribution analyses, are reported in Appendix L.

4.2 Designing the Sign-Separated Coarse Scaffold

Before training the final coarse-to-fine grammar, we audit how signed coarse-rule capacity should be allocated. We compare a sign-separated coarse teacher with an attribute-only coarse teacher, and use a relaxed diagnostic probe only for scaffold-capacity auditing. This probe tests whether dormant rules reflect true redundancy or optimization starvation before the final PIGRAM grammar is trained. Table 2 summarizes these diagnostics.

The attribute-only teacher collapses into suppressive rules, whereas sign separation preserves active favorable pathways. The relaxed probe further shows an asymmetry in rule utility: the favorable probe captures a small but highly concentrated high-response family, while the suppressive probe

Table 2: **Design diagnostics for the coarse grammar scaffold.** Panel A compares dominant-rule assignments between sign-separated and attribute-only coarse teachers. Panels B–C summarize a relaxed diagnostic probe used only for scaffold-capacity auditing, not as the final PIGRAM rule atlas. S and F denote suppressive and favorable rules; high-response pairs satisfy $Y_{ij}^{\mathrm{struct}} \geq 0.5$.

A. Coarse-teacher dominant-rule diagnostics

	Sign-separated teacher			Attribute-only teacher		
Rule	Weight	All	High	Weight	All	High
S0	-1.248	791.6k (91.1%)	7.3k (65.6%)	-1.075	631.4k (91.8%)	8.4k (85.4%)
S1	-1.066	9.1k (1.05%)	292 (2.62%)	-0.790	56.2k (8.17%)	1.4k (14.6%)
S2	-0.820	57 (<0.01%)	0 (0%)	-0.510	0 (0%)	0 (0%)
S3	-0.550	44 (<0.01%)	0 (0%)	-0.230	0 (0%)	0 (0%)
F4	0.239	0 (0%)	0 (0%)	0.101	0 (0%)	0 (0%)
F5	0.481	0 (0%)	0 (0%)	0.390	0 (0%)	0 (0%)
F6	0.711	801 (0.09%)	20 (0.18%)	0.667	0 (0%)	0 (0%)
F7	0.938	67.2k (7.74%)	3.5k (31.6%)	0.953	0 (0%)	0 (0%)

B. Probe support audit

Probe	Support	High	Conc.	Pattern
Suppr.	15.8k	559	3.54%	RNA loop
Fav.	2.6k	1.3k	49.75%	DNA tail

C. Probe ablation

Variant	Test P	AUROC
Full probe	0.4668	0.7878
Abl. Suppr.	0.4640	0.7783
Abl. Fav.	0.4649	0.7874

Table 3: Representative coarse rules and fine subtypes from the final grammar atlas. Coarse-rule counts are measured on high-response pairs. Fine subtypes are interpreted relative to their parent rule. The complete atlas is provided in Appendix G. Motif associations aligned with the target-construction multiplier are prior-informed.

Unit	Sign	Support	Motif context	Patch context	Interpretation
R2	–	62,812	Stem-edge; DNA+	High-offset topology	Topology-driven suppression
2.1		–	Stem-core; DNA-like	Shift from stem-edge penalty	Stem-core rescue subtype
R3	+	32,184	Stem-core; DNA-dom.	Arg-rich; extended geometry	High-confidence favorable expert
3.1		–	Stem-core; DNA-strong	High offset; extended geometry	Strong favorable subtype
R4	+	272,792	Stem-core; canonical DNA	Moderate offset; broad geometry	Canonical favorable basin
4.1		–	Stem-core; DNA-shifted	Extended; distance-enriched	Geometry-resolved favorable subtype
R5	+	21,783	Tail-unpaired; RNA+	Local topology; low pairing	Rare tail family
5.2		–	Rare stem-core/DNA mode	Distance; cation–π enriched	Rare stem-core rescue subtype

behaves more like a broad residual constraint. We therefore use an asymmetric final scaffold with compressed suppressive capacity and retained favorable capacity. The relaxed probe is used only for capacity allocation; final rule identities are re-audited after coarse-to-fine training and may reorganize.

4.3 Coarse-to-Fine Grammar Atlas

We next inspect the signed, attribute-linked patch–motif regimes represented by the learned grammar. After coarse-to-fine training, each patch–motif pair is assigned to its dominant final coarse rule, and representative coarse rules and fine subtypes are summarized in Table 3. Coarse-rule indices refer to the final PIGRAM grammar, independent of the diagnostic probes in Section 4.2; Rules 0–2 are suppressive and Rules 3–6 are favorable, and fine subtypes are interpreted relative to their parent coarse rule.

A literature-grounded biological rationale linking these rules to known protein–nucleic-acid recognition principles is provided in Appendix G.3.

Rule contributions directly generate predictions, while biological names are assigned after training. Because motif type, pairing, GC content, and position also enter target construction, atlas components aligned with those priors are described as *prior-informed*, rather than independently discovered mechanisms. A prior-only baseline and within-motif analysis demonstrate additional protein-conditioned structure (Appendix E.5).

Table 3 shows that the final grammar contains both coarse compatibility basins and finer local submechanisms. At the coarse level, Rule 2 captures a stem-edge/DNA suppressive regime with high-offset patch topology, while Rules 3 and 4 separate a compact arginine-rich favorable expert from a broader canonical stem-core/DNA basin. Rule 5 preserves a rare tail-unpaired parent family rather than smoothing it into background.

The representative fine subtypes show how selective refinement resolves within-rule heterogeneity. Fine 2.1 identifies a stem-core rescue subtype within a suppressive parent rule, Fine 3.1 isolates a strong favorable stem-core/DNA mode, Fine 4.1 refines the broad canonical basin into a geometry-resolved favorable subtype, and Fine 5.2 exposes a rare stem-core/DNA minority mode within a tail-unpaired parent family. These examples describe patch- and motif-associated compatibility regimes; motif components aligned with target construction are prior-informed, while evidence for additional protein-conditioned learning comes from the within-motif control in Appendix E.5.

Appendix Figure 6 further shows that fine refinement is selective rather than uniform. Refined parent rules split into local subtypes to resolve hidden heterogeneity, while Rule 6 remains coarse-only. The complete coarse-rule atlas, fine-subtype atlas, biological rationale, and rule–attribute alignment heatmaps are provided in Appendix G. Together, these diagnostics support an attribute-linked patch–motif grammar. We next examine whether bounded residual rescue improves grammar-hard cases without becoming an unconstrained fallback predictor.

4.4 Bounded Residual Rescue Targets Grammar-Hard Regions

A key risk in hybrid interpretable–latent models is that the latent branch may silently become the true predictor. PIGRAM addresses this risk by freezing the coarse-to-fine grammar foundation before training the residual Transformer, so the residual branch can only apply the bounded local correction in Eq. 2.

Table 4 summarizes the progression from coarse grammar to final PIGRAM and isolates fine refinement under the same residual-rescue protocol. Coarse-to-fine refinement improves the frozen grammar foundation. Removing fine refinement still improves over the coarse teacher, but requires a larger mean route (0.0756) and remains below the final model. Thus, fine refinement improves the explicit grammar and reduces the burden on latent correction rather than merely shifting prediction to the residual branch.

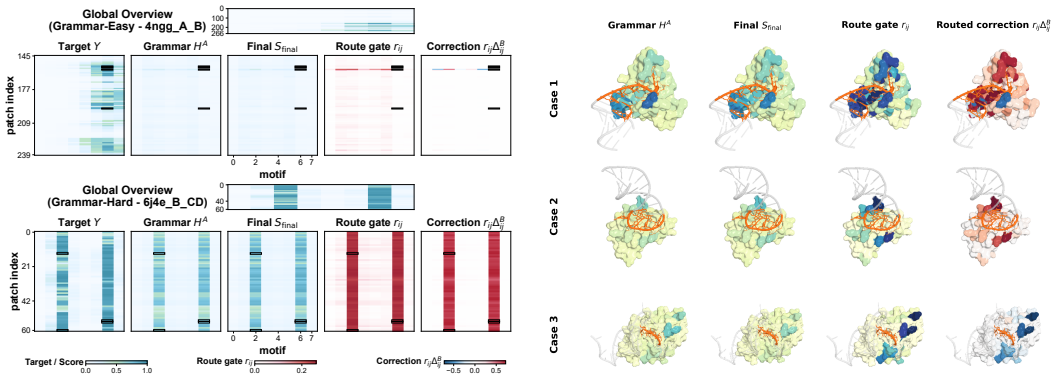
Table 4: Progression from coarse grammar to final PIGRAM, with a fine-refinement ablation. Values are means over five training seeds; subscripts denote standard deviations.

Model / score	Test P	Test S	Mean route
Coarse teacher G^{coarse}	$0.4619_{.0012}$	$0.3437_{.0036}$	0
Grammar foundation H^A	$0.4770_{.0062}$	$0.3820_{.0017}$	0
Coarse-only grammar + residual	$0.4836_{.0014}$	$0.3826_{.0076}$	$0.0756_{.0054}$
Final PIGRAM	$\mathbf{0.4937}_{.0049}$	$\mathbf{0.3833}_{.0038}$	$\mathbf{0.0377}_{.0064}$

The low global route does not mean the residual branch is inactive. Instead, routing concentrates on grammar-hard regions. We define grammar-hard pairs as the top quartile of frozen-grammar error and grammar-easy pairs as the bottom quartile. Appendix Figure 3 shows that grammar-hard pairs receive larger routes than grammar-easy pairs, high-route pairs are enriched for hard cases, and rescued-hard pairs obtain the largest error reduction.

Figure 2(A–B) provides complementary local views. Dense matrix cases show that routing remains nearly closed for a grammar-easy pair but opens locally for a grammar-hard rescued pair. Structure-level projections show the same behavior on protein surfaces: the final field remains aligned with the frozen grammar, while routing and routed correction concentrate near selected ssNA motifs.

Together, the route atlas, dense case studies, and structure projections show that the residual branch acts as a sparse local correction module rather than a hidden replacement for the explicit grammar. Additional fine-subtype route analyses, attribute-level route signatures, and attribute-shuffle ablations are provided in Appendix E.



(A) Dense case studies **(B) Structure-level projection**

Figure 2: **Local views of bounded residual rescue.** (A) Dense cases compare grammar-easy and grammar-hard rescued complexes (`4ngg_A_B`, `6j4e_B_CD`), showing that routing remains nearly closed in the grammar-easy case but opens locally in the rescued case. (B) Structure projections show motif-proximal routing and correction in representative complexes (`6GBM_B_A`, `2ESE_A_B`, `4ATO_A_G`), while the final field remains anchored to the grammar foundation.

4.5 Downstream aptamer screening and experimental validation

We evaluate downstream utility in two settings: candidate prioritization and grammar-guided experimental design. For screening, candidate-level readouts and affinity directions were selected on validation splits and fixed for five full protein–aptamer pools: GFP, HNRNPC, NELF, CHK2, and UBLCP1. The final score S_{final} achieved the best Top10, AUPRC, and AUROC, while the grammar foundation H^A achieved the best Top50 enrichment (Table 5). This cutoff-dependent behavior supports the intended division of labor: H^A acts as a stable broad screening prior, whereas residual rescue sharpens high-confidence retrieval and improves overall ranking rather than uniformly improving every enrichment cutoff.

We then used the signed grammar to design directional mutations. GFP served as a suppressive-rule case, where relief of a recurrent Rule 0 stem-core context should lower apparent K_D. NELF served as a favorable-rule case, where disruption of Rule 4 stem-core contexts should increase apparent K_D. For each backbone, we tested WT, wild-type reference; SP, structure-preserving mutant; SD, structure-disrupting mutant; R0, Rule 0-relief mutant for GFP; R4, Rule 4-conservative mutant for NELF; and RC, composition-matched random control. Apparent K_D was measured by fluorescence ELISA with five replicates per sequence; design details are provided in Appendix I. The affinity shifts matched the predicted rule directions (Table 5). In GFP, R0 produced the largest K_D decrease across both backbones, while SP and RC stayed much closer to WT. In NELF, SD produced the largest K_D increases, whereas R4 retained binding closest to WT on average across the two backbones and separated from the composition-matched RC. These results support a bidirectional grammar interpretation: suppressive-rule relief improves affinity, favorable-rule disruption weakens affinity, and favorable-rule-preserving edits better maintain WT-like binding than composition-matched perturbations.

Together, the downstream studies show that the learned grammar supports both prioritization and design. Its signed rules provide experimentally validated perturbation directions, while the composition-matched controls show that sequence identity changes alone can perturb binding and that grammar-guided edits can better preserve or improve binding relative to generic sequence perturbations.

Table 5: **Downstream screening and experimental validation.** (A) Candidate-level screening on five downstream protein–aptamer pools using validation-selected readouts; values are macro-averaged over targets and subscripts denote standard deviations over five seeds. (B) Grammar-guided affinity validation; apparent K_D values (nM) are mean $\pm$ SD from five fluorescence-ELISA replicates, rounded to the nearest integer. Lower K_D indicates stronger binding. Significance is computed against matched WT using a two-sided Welch's t-test: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$, ns = not significant.

A. Downstream screening

Score	Top10	Top50	AUPRC	AUROC
Final S_{final}	$\mathbf{0.430}_{.089}$	$0.245_{.076}$	$\mathbf{0.314}_{.042}$	$\mathbf{0.641}_{.063}$
Grammar H^A	$0.421_{.079}$	$\mathbf{0.261}_{.036}$	$0.284_{.054}$	$0.632_{.048}$
Transformer	$0.360_{.097}$	$0.258_{.054}$	$0.257_{.065}$	$0.640_{.074}$
Concat	$0.340_{.053}$	$0.238_{.038}$	$0.252_{.064}$	$0.620_{.095}$
Bilinear	$0.260_{.054}$	$0.190_{.073}$	$0.220_{.085}$	$0.600_{.054}$
Proto+res.	$0.248_{.084}$	$0.193_{.013}$	$0.187_{.018}$	$0.571_{.022}$
Prototype	$0.240_{.085}$	$0.157_{.035}$	$0.171_{.046}$	$0.482_{.065}$

B. Grammar-guided affinity validation

	GFP		*NELF*	
Var.	603	1434	6260	4894
WT	676 ± 13	659 ± 13	287 ± 9	439 ± 5
SP	$652\pm7^{*}$	$661\pm16^{\text{ns}}$	$352\pm8^{***}$	$491\pm6^{***}$
SD	$545\pm10^{***}$	$586\pm12^{***}$	$564\pm19^{***}$	$688\pm12^{***}$
R0	$490\pm13^{***}$	$516\pm8^{***}$		
R4			$330\pm6^{***}$	$479\pm7^{***}$
RC	$657\pm8^{*}$	$654\pm12^{\text{ns}}$	$387\pm7^{***}$	$519\pm13^{***}$

5 Conclusion and Limitations

PIGRAM learns a sign-separated coarse-to-fine grammar over explicit patch and motif attributes, then freezes it before adding sparse, bounded residual Transformer corrections. On the exact-sequence-pair-disjoint benchmark, it achieves Pearson 0.494 with mean route 0.038, concentrating corrections in grammar-hard regions. The grammar provides a broad screening prior, while the final score improves top-ranked retrieval. Wet-lab validation shows that Rule 0 relief improves GFP binding, Rule 4 disruption weakens NELF binding, and Rule 4-conservative edits better preserve WT-like binding than composition-matched controls. These results support interpretable grammar as a predictive scaffold and an actionable guide for optimizing synthetic nucleic acids.

PIGRAM requires a target protein structure and predicted aptamer secondary structure, but no candidate-specific 3D prediction or docking for fixed-target screening. The Transformer has better local Spearman correlation; the paired Pearson difference remains unresolved and does not establish equivalence. PIGRAM therefore contributes intrinsic interpretability, candidate prioritization, and directionally supported edits rather than structural-accuracy superiority.

The additional split establishes generalization only across full-protein sequence clusters at 30% identity and 80% bilateral coverage. Local-domain and 3D-interface similarity were not audited, and domain, fold, interface, and nucleic-acid-family independence is not established. Supervision-aligned motif patterns are prior-informed, and biological rule names are assigned after training. The atlas is a compatibility map that does not model atomic physics, solvent effects, induced fit, or long-range allostery. Validation covers only two targets and focused perturbations; broader experiments are needed to establish rule generality.

Acknowledgments and Disclosure of Funding

This research received no external funding. The author declares no competing interests.

References

[1] Thomas Hermann and Dinshaw J Patel. Adaptive recognition by nucleic acid aptamers. *Science*, 287(5454):820–825, 2000.

[2] Simone Fortunati, Federica Pedrini, Erica Del Grosso, Lorena Baranda Pellejero, and Alessandro Bertucci. Design of specific nucleic acid-based biosensors for protein binding activity. *Analysis & Sensing*, 2(6):e202200037, 2022.

[3] William H Hudson and Eric A Ortlund. The structure, function and evolution of proteins that bind dna and rna. *Nature reviews Molecular cell biology*, 15(11):749–760, 2014.

[4] Cailing Ji, Junyuan Wei, Lei Zhang, Xinru Hou, Jie Tan, Quan Yuan, and Weihong Tan. Aptamer–protein interactions: from regulation to biomolecular detection. *Chemical reviews*, 123(22):12471–12506, 2023.

[5] Luis M Alvarez-Salas. Nucleic acids as therapeutic agents. *Current topics in medicinal chemistry*, 8(15):1379–1404, 2008.

[6] Alan M Gewirtz, Deborah L Sokol, and Mariusz Z Ratajczak. Nucleic acid therapeutics: state of the art and future prospects. *Blood, The Journal of the American Society of Hematology*, 92 (3):712–736, 1998.

[7] Andrey A Buglak, Alexey V Samokhvalov, Anatoly V Zherdev, and Boris B Dzantiev. Methods and applications of in silico aptamer design and modeling. *International journal of molecular sciences*, 21(22):8420, 2020.

[8] Michiaki Hamada. In silico approaches to rna aptamer design. *Biochimie*, 145:8–14, 2018.

[9] Jayesh A Kulkarni, Dominik Witzigmann, Sarah B Thomson, Sam Chen, Blair R Leavitt, Pieter R Cullis, and Roy Van Der Meel. The current landscape of nucleic acid therapeutics. *Nature nanotechnology*, 16(6):630–643, 2021.

[10] Henning Ulrich, Cleber A Trujillo, Arthur A Nery, Janaina M Alves, Paromita Majumder, Rodrigo R Resende, and Antonio H Martins. Dna and rna aptamers: from tools for basic research towards therapeutic applications. *Combinatorial chemistry & high throughput screening*, 9(8): 619–632, 2006.

[11] Sara Tombelli, Maria Minunni, and Marco Mascini. Analytical applications of aptamers. *Biosensors and Bioelectronics*, 20(12):2424–2434, 2005.

[12] Maria C DeRosa, Amy Lin, Prabodhika Mallikaratchy, Erin M McConnell, Maureen McKeague, Rutika Patel, and Sarah Shigdar. In vitro selection of aptamers and their applications. *Nature Reviews Methods Primers*, 3(1):54, 2023.

[13] Yi Xi Wu and Young Jik Kwon. Aptamers: The "evolution" of selex. *Methods*, 106:21–28, 2016.

[14] Matthew R Dunn, Randi M Jimenez, and John C Chaput. Analysis of aptamer discovery and technology. *Nature Reviews Chemistry*, 1(10):0076, 2017.

[15] Subash Chandra Bose Gopinath. Methods developed for selex. *Analytical and bioanalytical chemistry*, 387(1):171–182, 2007.

[16] Haoping Wang, Xiangjie Meng, and Yang Zhang. Biomolecular interaction prediction: the era of ai. *Advanced Science*, 12(38):e09501, 2025.

[17] Shreyas Kaptan and Ilpo Vattulainen. Machine learning in the analysis of biomolecular simulations. *Advances in physics: X*, 7(1):2006080, 2022.

[18] Minhyeok Lee. Recent advances in deep learning for protein-protein interaction analysis: A comprehensive review. *Molecules*, 28(13):5169, 2023.

[19] Tamar Schlick, Stephanie Portillo-Ledesma, Christopher G Myers, Lauren Beljak, Justin Chen, Sami Dakhel, Daniel Darling, Sayak Ghosh, Joseph Hall, Mikaeel Jan, et al. Biomolecular modeling and simulation: a prospering multidisciplinary field. *Annual review of biophysics*, 50(1):267–301, 2021.

[20] Nicolas Renaud, Cunliang Geng, Sonja Georgievska, Francesco Ambrosetti, Lars Ridder, Dario F Marzella, Manon F Réau, Alexandre MJJ Bonvin, and Li C Xue. Deeprank: a deep learning framework for data mining 3d protein-protein interfaces. *Nature communications*, 12 (1):7068, 2021.

[21] Mohammed AlQuraishi and Peter K Sorger. Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms. *Nature methods*, 18 (10):1169–1180, 2021.

[22] Shuangli Li, Jingbo Zhou, Tong Xu, Liang Huang, Fan Wang, Haoyi Xiong, Weili Huang, Dejing Dou, and Hui Xiong. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 975–985, 2021.

[23] Yitian Fang, Yi Jiang, Leyi Wei, Qin Ma, Zhixiang Ren, Qianmu Yuan, and Dong-Qing Wei. Deepprosite: structure-aware protein binding site prediction using esmfold and pretrained language model. *Bioinformatics*, 39(12):btad718, 2023.

[24] Vitalii Stebliankin, Azam Shirali, Prabin Baral, Jimeng Shi, Prem Chapagain, Kalai Mathee, and Giri Narasimhan. Evaluating protein binding interfaces with transformer networks. *Nature Machine Intelligence*, 5(9):1042–1053, 2023.

[25] Abel Chandra, Laura Tünnermann, Tommy Löfstedt, and Regina Gratz. Transformer-based deep learning for predicting protein properties in the life sciences. *eLife*, 12:e82819, jan 2023. ISSN 2050-084X. doi: 10.7554/eLife.82819. URL `https://doi.org/10.7554/eLife.82819`.

[26] Ananthan Nambiar, Maeve Heflin, Simon Liu, Sergei Maslov, Mark Hopkins, and Anna Ritz. Transforming the language of life: transformer neural networks for protein prediction tasks. In *Proceedings of the 11th ACM international conference on bioinformatics, computational biology and health informatics*, pages 1–8, 2020.

[27] Nguyen Quoc Khanh Le. Leveraging transformers-based language models in proteome bioinformatics. *Proteomics*, 23(23-24):2300011, 2023.

[28] Jian Jiang, Lu Ke, Long Chen, Bozheng Dou, Yueying Zhu, Jie Liu, Bengong Zhang, Tianshou Zhou, and Guo-Wei Wei. Transformer technology in molecular science. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 14(4):e1725, 2024.

[29] Yifan Wu, Min Gao, Min Zeng, Jie Zhang, and Min Li. Bridgedpi: a novel graph neural network for predicting drug–protein interactions. *Bioinformatics*, 38(9):2571–2578, 2022.

[30] Andrea Mastropietro, Giuseppe Pasculli, and Jürgen Bajorath. Learning characteristics of graph neural networks predicting protein–ligand affinities. *Nature Machine Intelligence*, 5(12): 1427–1436, 2023.

[31] Jie Yang, Yapeng Li, Guoyin Wang, Zhong Chen, and Di Wu. An end-to-end knowledge graph fused graph neural network for accurate protein-protein interactions prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 21(6):2518–2530, 2024.

[32] V Shobha Rani, S Venu Gopal, S Deepan, and Adepu Ranjith Kumar. Deep graph learning approaches for protein-ligand binding site prediction. In *2025 IEEE 4th International Conference for Advancement in Technology (ICONAT)*, pages 1–6. IEEE, 2025.

[33] Christina B. Azodi, Jiliang Tang, and Shin-Han Shiu. Opening the black box: Interpretable machine learning for geneticists. *Trends in genetics : TIG*, 36 6:442–455, 2020.

[34] Emrullah Şahin, Naciye Nur Arslan, and Durmuş Özdemir. Unlocking the black box: an in-depth review on interpretability, explainability, and reliability in deep learning. *Neural Computing and Applications*, 37:859 – 965, 2024.

[35] Nattavut Sriwiboon. Explainable artificial intelligence (xai) for efficient lung cancer diagnosis using transformer–cnn hybrid with metadata fusion. *Neural Computing and Applications*, 38, 2026.

[36] Valerie Chen, Muyu Yang, Wenbo Cui, Joon Sik Kim, Ameet Talwalkar, and Jian Ma. Applying interpretable machine learning in computational biology—pitfalls, recommendations and opportunities for new developments. *Nature Methods*, 21:1454 – 1461, 2024.

[37] Ruheng Wang, Yi Jiang, Junru Jin, Chenglin Yin, Haoqing Yu, Fengsheng Wang, Jiuxin Feng, Ran Su, Kenta Nakai, Quan Zou, and Leyi Wei. Deepbio: an automated and interpretable deep-learning platform for high-throughput biological sequence prediction, functional annotation and visualization analysis. *Nucleic Acids Research*, 51:3017 – 3029, 2023.

[38] Felix Wong, Satotaka Omori, Alicia Li, Aarti Krishnan, Ryan S. Lach, Joseph Rufo, Maxwell Z. Wilson, and James J. Collins. An explainable deep learning platform for molecular discovery. *Nature Protocols*, 20:1020 – 1056, 2024.

[39] Hans Tomas Rube, Chaitanya Rastogi, Siqian Feng, Judith F. Kribelbauer, Allyson Li, Basheer Becerra, Lucas A. N. Melo, Bach-Viet Do, Xiaoting Li, Hammaad Adam, Neel H. Shah, Richard S. Mann, and Harmen J. Bussemaker. Prediction of protein–ligand binding affinity from sequencing data with interpretable machine learning. *Nature Biotechnology*, 40:1520 – 1527, 2022.

[40] Antonio Lavecchia. Explainable artificial intelligence in drug discovery: Bridging predictive power and mechanistic insight. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 15, 2025.

[41] Yue Liu, Haocheng Tang, Taoyu Niu, and Junmei Wang. A comparative study of deep learning and classical modeling approaches for protein–ligand binding pose and affinity prediction in coronavirus main proteases. *Journal of Chemical Information and Modeling*, 66(1):731–743, 2026.

[42] Nelson R. C. Monteiro, Carlos J. V. Simões, Henrique V. Ávila, Maryam Abbasi, José Luís Oliveira, and Joel P. Arrais. Explainable deep drug–target representations for binding affinity prediction. *BMC Bioinformatics*, 23, 2022.

[43] Ali Tŭgrul Balcı, Mark Maher Ebeid, Panayiotis V. Benos, Dennis Kostka, and Maria Chikina. An intrinsically interpretable neural network architecture for sequence-to-function learning. *Bioinformatics*, 39:i413 – i422, 2023.

[44] Xiaowei Li, Cheng Chen, Zihao Zhao, Qingyong Wang, and Lichuan Gu. Interpretable graph-embedding framework based on joint feature similarity for drug–drug interaction prediction. *Electronics*, 2026.

[45] Chandra Mohan Dasari, Santhosh Amilpur, and Raju Bhukya. Exploring variable-length features (motifs) for predicting binding sites through interpretable deep neural networks. *Eng. Appl. Artif. Intell.*, 106:104485, 2021.

[46] Jieping Ye and Jun Liu. Sparse methods for biomedical data. *SIGKDD explorations : newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining*, 14 1:4–15, 2012.

[47] Isabella Caranzano, Corrado Pancotti, Cesare Rollo, Flavio Sartori, Pietro Lio, Piero Fariselli, and Tiziana Sanavia. Sparsity is all you need: Rethinking biological pathway-informed approaches in deep learning. *ArXiv*, abs/2505.04300, 2025.

[48] Andreas Madsen, Siva Reddy, and A. P. Sarath Chandar. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55:1 – 42, 2021.

[49] Carla Piazzon Ramos Vieira and Luciano Antonio Digiampietri. Machine learning post-hoc interpretability: a systematic mapping study. *Proceedings of the XVIII Brazilian Symposium on Information Systems*, 2022.

[50] Yashbir Singh, Quincy A. Hathaway, Varekan Keishing, Sara Salehi, Yujia Wei, Natally Horvat, Diana Victoria Vera-Garcia, Ashok Choudhary, Almurtadha Mula Kh, Emilio Quaia, and Jesper B. Andersen. Beyond post hoc explanations: A comprehensive framework for accountable ai in medical imaging through transparency, interpretability, and explainability. *Bioengineering*, 12, 2025.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[52] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *Neural Information Processing Systems*, 2021.

[53] Tinglin Huang, Zhenqiao Song, Rex Ying, and Wengong Jin. Protein-nucleic acid complex modeling with frame averaging transformer. *Advances in Neural Information Processing Systems 37*, 2024.

[54] Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572, 2019.

[55] Ali Bashir, Qin Yang, Jinpeng Wang, Stephan Hoyer, Wen-Chuan Chou, Cory Y. McLean, Geoff Davis, Qiang Gong, Zan Armstrong, Junghoon Jang, Hui Kang, Annalisa Pawlosky, Alexander Scott, George E. Dahl, Marc Berndl, Michelle Dimon, and B Scott Ferguson. Machine learning guided aptamer refinement and discovery. *Nature Communications*, 12, 2021.

[56] Andrea Di Gioacchino, Jonah Procyk, Marco Molari, John S Schreck, Yu Zhou, Yan Liu, Rémi Monasson, Simona Cocco, and Petr Šulc. Generative and interpretable machine learning for aptamer design and analysis of in vitro sequence selection. *PLoS computational biology*, 18 (9):e1010561, 2022.

[57] Mahsa Torkamanian-Afshar, Sajjad Nematzadeh, Maryam Tabarzad, Ali Najafi, Hossein Lanjanian, and Ali Masoudi-Nejad. In silico design of novel aptamers utilizing a hybrid method of machine learning and genetic algorithm. *Molecular Diversity*, 25:1395 – 1407, 2021.

[58] Zihao Chen, Longqin Hu, Bao-Ting Zhang, Aiping Lu, Yaofeng Wang, Yuanyuan Yu, and Ge Zhang. Artificial intelligence in aptamer–target binding prediction. *International Journal of Molecular Sciences*, 22, 2021.

[59] Michael Kohlberger and Gabriele Gadermaier. Selex: Critical factors and optimization strategies for successful aptamer selection. *Biotechnology and Applied Biochemistry*, 69:1771 – 1792, 2021.

[60] Ryoga Ishida, Tatsuo Adachi, Aya Yokota, Hidehito Yoshihara, Kazuteru Aoki, Yoshikazu Nakamura, and Michiaki Hamada. Raptranker: in silico rna aptamer selection from ht-selex experiment based on local sequence and structure information. *Nucleic acids research*, 48(14): e82–e82, 2020.

[61] Munazah Andrabi, Kenji Mizuguchi, Akinori Sarai, and Shandar Ahmad. Prediction of mono- and di-nucleotide-specific dna-binding sites in proteins using neural networks. *BMC Structural Biology*, 9:30 – 30, 2009.

[62] Chun Hua Li, Libin Cao, Ji Guo Su, Yong Xiao Yang, and Cunxin Wang. A new residue-nucleotide propensity potential with structural information considered for discriminating protein-rna docking decoys. *Proteins: Structure*, 80, 2012.

[63] Yaoxin Jiang, Hui-Fang Liu, and Rong Liu. Systematic comparison and prediction of the effects of missense mutations on protein-dna and protein-rna interactions. *PLoS Computational Biology*, 17, 2021.

[64] Zhen Yang, Xueqing Deng, Yang Liu, Weikang Gong, and Chunhua Li. Analyses on clustering of the conserved residues at protein-rna interfaces and its application in binding site identification. *BMC Bioinformatics*, 21, 2020.

[65] Junkang Wei, Siyuan Chen, Licheng Zong, Xin Gao, and Yu Li. Protein–rna interaction prediction with deep learning: structure matters. *Briefings in Bioinformatics*, 23, 2021.

[66] M. Michael Gromiha and Kannan Harini. Protein-nucleic acid complexes: Docking and binding affinity. *Current opinion in structural biology*, 90:102955, 2024.

[67] Zhijie Liu, Jun tao Guo, Ting Li, and Ying Xu. Structure-based prediction of transcription factor binding sites using a protein-dna docking approach. *Proteins: Structure*, 72, 2008.

[68] Kalli Kappel and Rhiju Das. Sampling native-like structures of rna-protein complexes through rosetta folding and docking. *bioRxiv*, 2018.

[69] Kannan Harini, Daisuke Kihara, and M Michael Gromiha. Pda-pred: Predicting the binding affinity of protein-dna complexes using machine learning techniques and structural features. *Methods*, 2023.

[70] Binh P. Nguyen, Quang H. Nguyen, Giang-Nam Doan-Ngoc, Thanh-Hoang Nguyen-Vo, and Susanto Rahardja. iprodna-capsnet: identifying protein-dna binding residues using capsule neural networks. *BMC Bioinformatics*, 20, 2019.

[71] Jared M. Sagendorf, Raktim Mitra, Jiawei Huang, Xiaojiang S. Chen, and Remo Rohs. Pnabind: Structure-based prediction of protein-nucleic acid binding using graph neural networks. *bioRxiv*, 2024.

[72] Ka chun Wong, Yue Li, Chengbin Peng, Alan M. Moses, and Zhaolei Zhang. Computational learning on specificity-determining residue-nucleotide interactions. *Nucleic Acids Research*, 43:10180 – 10189, 2015.

[73] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[74] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1993.

[75] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv*, abs/1701.06538, 2017.

[76] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[77] Helen M. Berman, John D. Westbrook, Zukang Feng, Gary L Gilliland, Talapady N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic acids research*, 28 1:235–42, 2000.

[78] Helen M. Berman, Wilma K. Olson, David L. Beveridge, John D. Westbrook, Anke Gelbin, Tamas Demeny, Shu-Hsin Hsieh, Annankoil R. Srinivasan, and Bohdan Schneider. The nucleic acid database. a comprehensive relational database of three-dimensional structures of nucleic acids. *Biophysical journal*, 63 3:751–9, 1992.

[79] Bartosz Adamczyk, Maciej Antczak, and Marta Szachniuk. Rnasolo: a repository of cleaned pdb-derived rna 3d structures. *Bioinformatics*, 38:3668 – 3670, 2022.

[80] Meredith Corley, Margaret C Burns, and Gene W Yeo. How rna-binding proteins interact with rna: molecules and mechanisms. *Molecular cell*, 78(1):9–29, 2020.

[81] David E. Draper. Themes in rna-protein recognition. *Journal of molecular biology*, 293 2: 255–70, 1999. URL https://api.semanticscholar.org/CorpusID:33091280.

[82] Susan Jones, D. A. Daley, Nicholas M. Luscombe, Helen M. Berman, and Janet M. Thornton. Protein-rna interactions: a structural analysis. *Nucleic acids research*, 29 4:943–54, 2001.

[83] Remo Rohs, Sean M West, Alona Sosinsky, Peng Liu, Richard S. Mann, and Barry Honig. The role of dna shape in protein-dna recognition. *Nature*, 461:1248 – 1253, 2009.

[84] Diane Lejeune, Nicolas Delsaux, Benoît Charloteaux, Annick Thomas, and Robert Brasseur. Protein–nucleic acid recognition: Statistical analysis of atomic interactions and influence of dna structure. *Proteins: Structure*, 61, 2005.

[85] Katie A. Wilson, Devany J Holland, and Stacey D. Wetmore. Topology of rna–protein nucleobase–amino acid π–π interactions and comparison to analogous dna–protein π–π contacts. *RNA*, 22:696 – 708, 2016.

[86] Dennis M. Krüger, Saskia Neubacher, and Tom N. Grossmann. Protein–rna interactions: structural characteristics and hotspot amino acids. *RNA*, 24:1457 – 1465, 2018.

[87] Nicholas M. Luscombe, Susan E Austin, Helen M. Berman, and Janet M. Thornton. An overview of the structures of protein-dna complexes. *Genome Biology*, 1:reviews001.1 – reviews001.37, 2000.

[88] Colin W. Garvie and Cynthia Wolberger. Recognition of specific dna sequences. *Molecular cell*, 8 5:937–46, 2001.

[89] Remo Rohs, Xiangshu Jin, Sean M West, Rohit Joshi, Barry Honig, and Richard S. Mann. Origins of specificity in protein-dna recognition. *Annual review of biochemistry*, 79:233–69, 2010.

[90] Binhan Yu, B. Montgomery Pettitt, and Junji Iwahara. Dynamics of ionic interactions at protein–nucleic acid interfaces. *Accounts of Chemical Research*, 53:1802 – 1810, 2020.

[91] Roopa Thapar, Andria P. Denmon, and Edward P. Nikonowicz. Recognition modes of rna tetraloops and tetraloop-like motifs by rna-binding proteins. *Wiley Interdisciplinary Reviews: RNA*, 5, 2014.

[92] Barbara J Calnan, Bruce Tidor, Sara Biancalana, Derek Hudson, and Alan D Frankel. Arginine-mediated rna recognition: the arginine fork. *Science*, 252(5009):1167–1171, 1991.

[93] Nicholas M Luscombe, Roman A Laskowski, and Janet M Thornton. Amino acid–base interactions: a three-dimensional analysis of protein–dna interactions at an atomic level. *Nucleic acids research*, 29(13):2860–2874, 2001.

[94] Garima Mishra and Yaakov Levy. Molecular determinants of the interactions between proteins and ssdna. *Proceedings of the National Academy of Sciences*, 112(16):5033–5038, 2015.

[95] Maoxuan Lin, Fareeha K Malik, and Jun-tao Guo. A comparative study of protein–ssdna interactions. *NAR genomics and bioinformatics*, 3(1):lqab006, 2021.

[96] Jun-Tao Guo and Fareeha Malik. Single-stranded dna binding proteins and their identification using machine learning-based approaches. *Biomolecules*, 12(9):1187, 2022.

[97] Arumay Pal and Yaakov Levy. Structure, stability and specificity of the binding of ssdna and ssrna with proteins. *PLoS computational biology*, 15(4):e1006768, 2019.

[98] Christophe Biot, Eric Buisine, Jean-Marc Kwasigroch, René Wintjens, and Marianne Rooman. Probing the energetic and structural role of amino acid/nucleobase cation-π interactions in protein-ligand complexes. *Journal of Biological Chemistry*, 277(43):40816–40822, 2002.

[99] Yulander Duh, Yu-Yuan Hsiao, Chia-Lung Li, Jason C Huang, and Hanna S Yuan. Aromatic residues in RNase T stack with nucleobases to guide the sequence-specific recognition and cleavage of nucleic acids. *Protein Science*, 24(12):1934–1941, 2015.

[100] Incheol Shin, Keumseok Kang, Juseong Kim, Sanghun Sel, Jeonghoon Choi, Jae-Wook Lee, Ho Young Kang, and Giltae Song. Aptatrans: a deep neural network for predicting aptamer-protein interaction using pretrained encoders. *BMC bioinformatics*, 24(1):447, 2023.

[101] Neda Emami and Reza Ferdousi. Aptanet as a deep learning approach for aptamer–protein interaction prediction. *Scientific reports*, 11(1):6074, 2021.

[102] Zheng Fang, Zhongqi Wu, Xinbo Wu, Shixin Chen, Xing Wang, Saurabh Umrao, and Abhisek Dwivedy. Apipred: An xgboost-based method for predicting aptamer–protein interactions. *Journal of Chemical Information and Modeling*, 64(7):2290–2301, 2024.

A Data processing and label construction

All datasets used in this work are obtained from public databases, official source pages, or previously published experimental studies. For structural interaction learning, we construct protein–nucleic-acid complex datasets from public structural resources, including the Protein Data Bank (PDB) [77], the Nucleic Acid Database (NDB) [78], and curated RNA structural resources such as RNASolo [79] where applicable. The structural datasets contain paired protein and nucleic-acid sequences, three-dimensional coordinates, and contact or distance information from which we derive patch–motif supervision for training, validation, and testing.

For downstream aptamer prioritization, we use five experimentally annotated RNA aptamer datasets targeting GFP, NELF, HNRNPC, CHK2, and UBLCP1 [53]. Candidate sequences and experimental annotations are obtained from the corresponding public source pages and original studies. These downstream labels are not used to train the structural interaction model. They are used only for downstream candidate-ranking evaluation and, within the downstream protocol, for validation-only readout and affinity-direction selection.

For qualitative visualization, we use selected protein–nucleic-acid complexes from the PDB [77]. These structures are used only to illustrate learned patch–motif grammar decompositions, favorable and suppressive rules, and residual corrections. They are not treated as an additional quantitative benchmark and are not used for model selection.

Detailed dataset statistics, original source links, preprocessing scripts, processed train/validation/test splits, and the PDB identifiers used for visualization are provided in the public code repository.

A.1 Pair-disjoint split protocol

Each sample is assigned a pair identifier defined by the MD5 hash of the concatenated protein and partner sequences:

$$\text{pair_key}(x) = \text{MD5}(\text{protein_seq}(x) \,\|\, \text{partner_seq}(x)) \,. \tag{12}$$

Samples are grouped by this identifier, unique pairs are shuffled with a fixed seed, and train/validation/test splits are created at the pair level using ratios of $0.87/0.065/0.065$. RNA and DNA sources are split separately and then merged split-wise. MD5 is used only to group exact sequence pairs; it is not a biological similarity measure, and this split does not enforce homology separation.

A.2 Protein patch construction

Let the protein coordinates be $\mathbf{X} \in \mathbb{R}^{L\times A\times 3}$, where L is the number of residues and A is the number of atoms per residue. The residue centroid of residue i is

$$\mathbf{c}_i = \frac{1}{A}\sum_{a=1}^{A} \mathbf{X}_{i,a}. \tag{13}$$

Pairwise centroid distances are computed as

$$D_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|_2. \tag{14}$$

For each residue i, a patch is defined as the set of its k nearest residues under this centroid distance, with $k = 8$ in the final pipeline.

A.3 Secondary structure and motif segmentation

The partner sequence is first normalized by converting it to uppercase, replacing `T` with `U`, and keeping only `A`, `U`, `G`, `C`, and `N`. Secondary structure is then predicted with ViennaRNA:

$$(\text{dot-bracket}, \text{MFE}) = \text{RNA.fold}(\text{sequence}). \tag{15}$$

The dot-bracket string is converted into token-level structural labels using a base-pair map. Paired positions are classified as either `stem_core` or `stem_edge`. Contiguous unpaired segments are classified as `hairpin_loop`, `internal_loop`, `bulge`, `multi_loop`, or `tail_unpaired`. Consecutive positions with the same label are merged into motifs.

A.4 Binary patch–motif supervision

For motif M_j with nucleotide index set $\mathcal{N}(M_j)$ and residue r_i, the residue-to-motif distance is computed by aggregating nucleotide distances. In the final pipeline, the default aggregation operator is `topk_mean` with $k = 3$:

$$d^{\mathrm{res}}_{i\to j} = \frac{1}{k}\sum_{\ell=1}^{k} \mathrm{smallest}_\ell\left(\{d(i,n) : n \in \mathcal{N}(M_j)\}\right). \tag{16}$$

For a patch P_p with residue set $\mathcal{R}(P_p)$, the patch-to-motif distance is aggregated again with `topk_mean`:

$$d^{\mathrm{patch}}_{p\to j} = \frac{1}{k}\sum_{\ell=1}^{k} \mathrm{smallest}_\ell\left(\{d^{\mathrm{res}}_{i\to j} : i \in \mathcal{R}(P_p)\}\right). \tag{17}$$

The binary patch–motif label is then

$$Y^{\mathrm{bin}}_{pj} = \mathbb{I}\Big[d^{\mathrm{patch}}_{p\to j} < 6.0\Big]. \tag{18}$$

Patch importance is derived by

$$Y^{\mathrm{patch}}_{p} = \mathbb{I}\left[\max_j Y^{\mathrm{bin}}_{pj} > 0\right]. \tag{19}$$

A.5 Structure-aware continuous label

The final continuous supervision target is `patch_motif_structure_energy_v2`. For residue i and motif M_j, let the vector of residue-to-nucleotide distances inside the motif be $\mathbf{d}_{ij} = \{d(i,n) : n \in \mathcal{N}(M_j)\}$. A softmax-weighted effective distance is computed as

$$\alpha_{ijn} = \frac{\exp(-\beta_{\mathrm{nt}} d(i,n))}{\sum_{n'\in\mathcal{N}(M_j)} \exp(-\beta_{\mathrm{nt}} d(i,n'))}, \qquad \tilde{d}_{ij} = \sum_{n\in\mathcal{N}(M_j)} \alpha_{ijn} d(i,n). \tag{20}$$

The residue-to-motif soft score is then

$$s^{\mathrm{res}}_{ij} = \exp\left(-\frac{\tilde{d}_{ij}}{\tau_{\mathrm{contact}}}\right). \tag{21}$$

For a patch P_p, patch-level aggregation uses a softmax over residue scores:

$$\beta_{pij} = \frac{\exp(\beta_{\mathrm{patch}} s^{\mathrm{res}}_{ij})}{\sum_{i'\in\mathcal{R}(P_p)} \exp(\beta_{\mathrm{patch}} s^{\mathrm{res}}_{i'j})}, \qquad s^{\mathrm{patch}}_{pj} = \sum_{i\in\mathcal{R}(P_p)} \beta_{pij} s^{\mathrm{res}}_{ij}. \tag{22}$$

Each motif is assigned a structure multiplier

$$m_j = w_{\mathrm{type}}(M_j)\,(1 + \lambda_{\mathrm{pair}}\rho_j)\,(1 + \lambda_{\mathrm{GC}} g_j)\,(1 + \lambda_{\mathrm{pos}} u_j), \tag{23}$$

where $w_{\mathrm{type}}(M_j)$ is a group-specific motif-type weight, ρ_j is the motif pairing ratio, g_j is GC content, and u_j is a position weight derived from the normalized motif center. In the final pipeline, motif groups are `stem`, `loop`, and `unpaired`, with default weights 1.5, 1.1, and 0.8, respectively.

The raw patch–motif score is

$$r_{pj} = s^{\mathrm{patch}}_{pj}\, m_j. \tag{24}$$

The final continuous target is

$$Y^{\mathrm{struct}}_{pj} = 1 - \exp\left(-\frac{r_{pj}}{\tau_{\mathrm{label}}}\right), \tag{25}$$

followed by clipping to the interval $[0, 1]$.

Table 6: Original and protein-cluster results: five-seed means with pair-clustered 95% confidence intervals.

Split	Model	Pearson [95% CI]	Spearman [95% CI]
Exact pair	Transformer	$0.5108\ [0.451, 0.556]$	$0.4820\ [0.426, 0.535]$
	PIGRAM	$0.4937\ [0.447, 0.542]$	$0.3833\ [0.329, 0.439]$
	Grammar Foundation	$0.4770\ [0.419, 0.527]$	$0.3820\ [0.323, 0.437]$
Protein cluster	Transformer	$0.495\ [0.443, 0.542]$	$0.454\ [0.397, 0.505]$
	PIGRAM	$0.479\ [0.423, 0.530]$	$0.367\ [0.313, 0.422]$
	Grammar Foundation	$0.447\ [0.390, 0.502]$	$0.362\ [0.307, 0.417]$

A.6 Default final preprocessing hyperparameters

The final dataset pipeline uses $\beta_{\mathrm{nt}} = 3.0$, $\beta_{\mathrm{patch}} = 6.0$, $\tau_{\mathrm{contact}} = 4.0$, $\tau_{\mathrm{label}} = 0.20$, $\lambda_{\mathrm{pair}} = 0.5$, $\lambda_{\mathrm{GC}} = 0.25$, $\lambda_{\mathrm{pos}} = 0.25$, and motif position mode `center_mild`.

A.7 Protein-cluster evaluation and statistical uncertainty

The original exact-sequence-pair-disjoint split contains 3,680/289/259 training/validation/test structures and 3,373/252/253 unique pairs. For the additional split, MMseqs2 matches with at least 30% full-protein sequence identity and 80% bilateral coverage define a graph. Each connected component and all associated structures are assigned exclusively to one partition. A post-split audit finds no cross-split match meeting this threshold. The resulting split contains 3,670/281/277 structures and 251 unique test pairs. Transformer, PIGRAM, and Grammar Foundation are retrained from scratch over five seeds. This analysis does not directly audit local-domain or three-dimensional-interface similarity.

We compute 95% percentile confidence intervals using 10,000 bootstrap replicates resampling unique sequence pairs with replacement, retaining all associated structures and patch–motif cells. Shared pair indices are used across models. Within each replicate, metrics are recomputed per seed and then averaged; paired differences are computed using the same resampling. The 2.5th and 97.5th percentiles define the interval. These intervals quantify test-pair sampling uncertainty, whereas five-seed SDs quantify training variability.

On the protein-cluster split, the paired PIGRAM–Transformer differences are -0.016 (95% CI $[-0.039, 0.009]$) for Pearson and -0.087 ($[-0.119, -0.054]$) for Spearman. Thus, the Pearson difference remains unresolved and Transformer has a supported local Spearman advantage on both splits. PIGRAM improves Pearson over Grammar Foundation by 0.0167 ($[0.004, 0.032]$) on the original split and 0.032 ($[0.012, 0.052]$) on the protein-cluster split.

Separately, 12/253 original-split test pairs have a training-protein match at at least 70% identity and 80% bilateral coverage. Excluding their associated test structures while keeping trained models fixed changes Pearson by -0.0027 for Transformer, -0.0029 for PIGRAM, and -0.0038 for Grammar Foundation, preserving model ordering. This sensitivity analysis does not replace the retrained cluster split or establish domain/interface independence.

B Patch–Motif Encoding and Explicit Attribute Definitions

This appendix specifies the encoder parameterization and explicit attribute groups used by PIGRAM. The main text emphasizes the separation between latent embeddings, which support neural refinement and residual rescue, and explicit attributes, which define the human-readable grammar vocabulary.

B.1 Encoder parameterization

For each protein patch P_i and nucleic-acid motif M_j, PIGRAM constructs latent embeddings $\mathbf{z}_i^{(p)}, \mathbf{z}_j^{(m)} \in \mathbb{R}^d$ and explicit attribute vectors $\mathbf{a}_i^{(p)} \in \mathbb{R}^{56}$, $\mathbf{a}_j^{(m)} \in \mathbb{R}^{27}$. The patch encoder combines center-residue information, aggregated local neighborhood information, and an embedded summary of the explicit patch attributes:

$$\mathbf{z}_i^{(p)} = f_p\Big([\mathbf{c}_i,\ \mathbf{n}_i,\ \mathbf{s}_i^{(p)}]\Big), \tag{26}$$

where $\mathbf{c}_i$ is the center-residue feature, $\mathbf{n}_i$ is the learned neighborhood feature, and $\mathbf{s}_i^{(p)}$ is the embedded explicit patch summary.

The motif encoder combines a sequence summary, scalar structural context, and motif-type information:

$$\mathbf{z}_j^{(m)} = f_m\Big([\mathbf{q}_j,\ \mathbf{s}_j^{(m)},\ \mathbf{t}_j]\Big), \tag{27}$$

where $\mathbf{q}_j$ is the motif sequence summary, $\mathbf{s}_j^{(m)}$ is the scalar-structure summary, and $\mathbf{t}_j$ is the motif-type embedding.

B.2 Patch attributes

Each protein patch is associated with a 56-dimensional explicit attribute vector $\mathbf{a}_i^{(p)} \in \mathbb{R}^{56}$. The attributes are grouped into amino-acid composition, physicochemical composition, local geometry, sequence topology, and derived patch-level proxies, as summarized in Table 7.

Table 7: Explicit protein-patch attribute groups.

Attribute group	Dim.	Description
Amino-acid composition profile	21	Mean one-hot amino-acid composition over the patch.
Grouped physicochemical profile	9	Fractions of positive/basic, negative/acidic, aromatic, hydrophobic, polar, glycine, proline, small, and bulky residues.
Distance statistics	10	Mean, standard deviation, minimum, maximum, median, non-center mean, non-center standard deviation, non-center minimum, non-center maximum, and average local-contact fraction.
Sequence-topology descriptors	9	Mean absolute offset, offset standard deviation, maximum absolute offset, fractions within offset thresholds 2, 4, and 8, fraction beyond 16, signed offset mean, and signed balance.
Derived patch-level proxies	7	Net charge, charge magnitude, cation–π proxy, aromatic-hydrophobic packing proxy, compactness proxy, locality balance, and asymmetry strength.

B.3 Motif attributes

Each nucleic-acid motif is associated with a 27-dimensional explicit attribute vector $\mathbf{a}_j^{(m)} \in \mathbb{R}^{27}$. The attributes encode motif identity, edge context, structural scalars, nucleotide composition, and derived motif-level biophysical proxies, as summarized in Table 8.

B.4 Vocabularies

The discrete vocabularies used to construct the explicit attributes are summarized in Table 9.

Table 8: Explicit nucleic-acid motif attribute groups.

Attribute group	Dim.	Description
Motif-type one-hot vector	8	Seven named structural motif categories together with one reserved/padding channel.
Edge-side one-hot vector	3	Encodes whether the motif is assigned to no edge, the left edge, or the right edge.
Structural-context scalars	5	Log-scaled motif length, pairing ratio, GC content, terminal indicator, and normalized center position.
Nucleotide-composition features	5	Fractions of A, C, G, U, and purine.
Derived motif-level biophysical proxies	6	Normalized nucleotide entropy, dominant nucleotide fraction, maximum homopolymer-run fraction, flexibility proxy, wobble-pair proxy, and DNA/RNA indicator.

Table 9: Discrete vocabularies used for explicit patch and motif attributes.

Vocabulary	Entries
Amino-acid vocabulary	`A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y, X`.
RNA vocabulary	`A, C, G, U, N`.
Motif-type vocabulary	`stem_core`, `stem_edge`, `hairpin_loop`, `internal_loop`, `bulge`, `multi_loop`, and `tail_unpaired`, with one additional reserved/padding channel.
Edge-side vocabulary	`none`, `left`, and `right`.

C Coarse Grammar Implementation Details

C.1 Attribute projections and rule anchors

For each sign branch $s \in \{\mathrm{neg}, \mathrm{pos}\}$, explicit patch and motif attributes are projected into rule-specific vectors:

$$\mathbf{u}_i^s = \mathrm{Proj}_p^s(\mathbf{a}_i^{(p)}), \qquad \mathbf{v}_j^s = \mathrm{Proj}_m^s(\mathbf{a}_j^{(m)}). \tag{28}$$

Learnable rule anchors are added before normalization, producing the rule-indexed vectors $\mathbf{u}_i^{(s,k)}$ and $\mathbf{v}_j^{(s,k)}$ used by the RBF compatibility in Eq. 4. This anchor-based parameterization allows different rules within the same sign branch to specialize to different regions of the explicit attribute space.

C.2 Asymmetric bandwidth parameterization

Suppressive and favorable branches use different admissible bandwidth intervals:

$$\sigma_{s,k}^{\mathrm{eff}} \in \mathcal{I}_s, \qquad \mathcal{I}_{\mathrm{neg}} = [0.10, 0.80], \qquad \mathcal{I}_{\mathrm{pos}} = [1.20, 2.40]. \tag{29}$$

During training, each bandwidth follows a schedule-controlled value:

$$\sigma_{s,k}^{\mathrm{schedule}}(p) = (1-p)\sigma_{s,k,\mathrm{start}} + p\,\sigma_{s,k,\mathrm{end}}, \tag{30}$$

where $p \in [0,1]$ denotes training progress. A learnable log-space residual correction is then applied:

$$\Delta_{s,k}^{\mathrm{clamped}} = \mathrm{clip}(\Delta_{s,k}, -c, +c), \qquad \sigma_{s,k}^{\mathrm{eff}} = \mathrm{clip}_{\mathcal{I}_s}\left(\sigma_{s,k}^{\mathrm{schedule}}(p)\exp(\Delta_{s,k}^{\mathrm{clamped}})\right). \tag{31}$$

This asymmetry is physically motivated rather than merely heuristic. Suppressive constraints often behave like sharp steric or electrostatic incompatibilities: once the patch and motif enter an unfavorable geometric or chemical regime, the interaction should be penalized quickly and locally. Favorable interactions are usually less brittle. They can tolerate modest geometric variation, conformational slack, or sequence-level substitutions while remaining broadly compatible. Narrow suppressive bandwidths therefore encourage sharp exclusion zones, whereas broader favorable bandwidths model softer compatibility basins with greater structural tolerance.

C.3 Rule confidence statistics

To summarize rule usage, we compute the normalized absolute rule mass:

$$p_{ij}^{(s,k)} = \frac{|C_{ij}^{(s,k)}|}{\sum_{s',k'} |C_{ij}^{(s',k')}| + \varepsilon}. \tag{32}$$

The dominant coarse rule is defined as

$$\rho(i,j) = \arg\max_{s,k} |C_{ij}^{(s,k)}|. \tag{33}$$

We also compute two confidence statistics:

$$\mathrm{topfrac}_{ij} = \max_{s,k} p_{ij}^{(s,k)}, \tag{34}$$

$$\mathrm{entropy}_{ij} = -\sum_{s,k} p_{ij}^{(s,k)} \log\Big(p_{ij}^{(s,k)} + \varepsilon\Big). \tag{35}$$

The top-rule fraction measures how concentrated the grammar explanation is around a single dominant rule, whereas rule entropy measures whether multiple rules contribute comparably. Together with the dominant rule assignment, these statistics are used for coarse-rule auditing, fine-refinement selection, and residual-route conditioning.

D Fine Pseudo-Label Generation and Fine-Refiner Details

This appendix describes how PIGRAM constructs fine pseudo-labels and trains rule-specific fine refiners. The goal is not to define ground-truth biological rule identities, but to expose stable within-rule heterogeneity that can scaffold selective fine refinement. In the final pipeline, pseudo-labels are generated from the pure sign-separated coarse grammar teacher; no residual head, rescue route, or mixture-style fallback pathway is enabled during this stage.

D.1 Coarse teacher outputs

For each sample, the coarse teacher produces the coarse grammar score, rule-contribution tensor, and explicit patch and motif attributes:

$$G_{ij}^{\mathrm{coarse}}, \qquad \{C_{ij}^{(k)}\}_{k=1}^{K}, \qquad \mathbf{a}_i^{(p)}, \qquad \mathbf{a}_j^{(m)}. \tag{36}$$

For each patch–motif pair, the dominant coarse rule is defined as

$$\rho(i,j) = \arg\max_{k} \left|C_{ij}^{(k)}\right|. \tag{37}$$

The normalized absolute rule mass is

$$p_{ij}^{(k)} = \frac{|C_{ij}^{(k)}|}{\sum_{\ell} |C_{ij}^{(\ell)}| + \varepsilon}. \tag{38}$$

In addition to these quantities, we compute grammar-side statistics such as dominant-rule margin, signed and absolute rule contribution, RBF response, coarse grammar score, and absolute coarse grammar score.

D.2 Rule-wise filtering and refinement plan

Patch–motif pairs are collected according to their dominant coarse rule and filtered to focus fine refinement on interaction-relevant regions. For each coarse rule r, the candidate set is

$$\Omega_r = \{(i,j) : \rho(i,j) = r,\ Y_{ij}^{\mathrm{struct}} \geq 0.5\}. \tag{39}$$

Thus, fine pseudo-labels are generated only from high-response pairs, avoiding the use of low-response background pairs to define fine subrules.

Let $\mathcal{R}_{\mathrm{fine}}$ denote the set of coarse rules selected for refinement. In the final distillation plan,

$$\mathcal{R}_{\mathrm{fine}} = \{0,1,2,3,4,5\}, \tag{40}$$

while Rule 6 is kept coarse-only because it showed structural signal but was less stable across training and validation audits. The refinement plan was fixed using training and validation data only, before test-set evaluation. The rule-specific fine-refinement plan is summarized in Table 10.

Table 10: Final fine-refinement distillation plan. Fine labels are generated from the pure coarse teacher using high-response pairs with $Y_{ij}^{\mathrm{struct}} \geq 0.5$. Rule 6 is kept coarse-only in the final model.

Coarse rule	Refined?	Fine clusters K_r	Feature mode	Train support
0	Yes	3	Specialized score-aware	405,940
1	Yes	4	Score-aware	230,408
2	Yes	4	Score-aware	138,841
3	Yes	3	Score-aware	304,323
4	Yes	2	Score-aware	15,991
5	Yes	3	Score-aware	3,694
6	No	1	Off	–

D.3 Rule-local clustering features

For each selected rule r, clustering is performed only within the corresponding candidate set Ω_r. The base feature contains explicit patch and motif attributes:

$$\mathbf{h}_{ij}^{\mathrm{attr}} = \left[\mathbf{a}_i^{(p)},\, \mathbf{a}_j^{(m)}\right]. \tag{41}$$

The final distillation plan uses score-aware features that augment explicit attributes with grammar-side statistics:

$$\mathbf{h}_{ij}^{\mathrm{aug}} = \left[\mathbf{a}_i^{(p)},\, \mathbf{a}_j^{(m)},\, G_{ij}^{\mathrm{coarse}},\, |G_{ij}^{\mathrm{coarse}}|,\, m_{ij}^{\mathrm{dom}},\, C_{ij}^{(\rho)},\, |C_{ij}^{(\rho)}|,\, \mathrm{RBF}_{ij}^{(\rho)}\right], \tag{42}$$

where m_{ij}^{dom} is the dominant-rule margin and $\rho = \rho(i,j)$ is the teacher-assigned dominant coarse rule. Rule 0 uses a specialized score-aware mode because it is the largest mixed suppressive family and benefits from a more targeted splitting strategy. All clustering features are standardized within each rule family. For large rule families, at most 120,000 pairs are stored for fitting the rule-local K-means model.

D.4 Pseudo-label assignment

Within each selected coarse rule r, K-means clustering is applied to the standardized rule-local features. The resulting cluster assignment defines the fine pseudo-label:

$$\tilde{y}_{ij}^{\mathrm{fine}} \in \{1, \ldots, K_r\}, \qquad (i,j) \in \Omega_r. \tag{43}$$

Pairs outside selected rules, low-response pairs, or pairs failing rule-specific filters do not contribute to the fine classification loss.

For each processed sample, we store the teacher-assigned dominant coarse rule $\rho(i,j)$, the fine pseudo-label $\tilde{y}_{ij}^{\mathrm{fine}}$ when available, and a binary distillation mask indicating whether the pair contributes to fine-rule supervision. These generated fields are used only as scaffolds for selective fine refinement during grammar-foundation training, rather than as fixed biological annotations.

D.5 Fine-refiner input and output

For each selected coarse rule $r \in \mathcal{R}_{\mathrm{fine}}$, the fine refiner receives a rule-specific patch–motif representation:

$$\mathbf{x}_{ij}^{(r)} = \left[\mathbf{z}_i^{(p)},\, \mathbf{z}_j^{(m)},\, \mathbf{z}_i^{(p)} \odot \mathbf{z}_j^{(m)},\, |\mathbf{z}_i^{(p)} - \mathbf{z}_j^{(m)}|,\, \mathbf{a}_i^{(p)},\, \mathbf{a}_j^{(m)},\, p_{ij}^{(r)},\, C_{ij}^{(r)},\, G_{ij}^{\mathrm{coarse}}\right]. \tag{44}$$

Here $p_{ij}^{(r)}$ is the normalized absolute rule mass of coarse rule r, and $C_{ij}^{(r)}$ is its signed rule contribution.

Each rule-specific fine refiner produces a fine-subrule classifier and a vector of scalar grammar refinements:

$$\mathrm{clslogits}_{ij}^{(r)} \in \mathbb{R}^{K_r}, \qquad \boldsymbol{\delta}_{ij}^{(r)} \in \mathbb{R}^{K_r}. \tag{45}$$

The fine-subrule assignment probabilities are

$$\mathbf{q}_{ij}^{(r)} = \mathrm{softmax}\left(\mathrm{clslogits}_{ij}^{(r)}\right), \tag{46}$$

and the rule-local fine grammar contribution is

$$g_{ij}^{(r)} = \sum_{k=1}^{K_r} q_{ij,k}^{(r)} \delta_{ij,k}^{(r)}. \tag{47}$$

D.6 Use during coarse-to-fine training

During grammar-foundation training, the fine-rule classification loss is applied only to pairs whose distillation mask is true:

$$\Omega_{\text{fine}} = \{(i, j) : \texttt{distill_mask}_{ij} = 1\}. \tag{48}$$

The final generated annotation set contains 1,099,197 distilled train pairs, corresponding to 9.41% of all train patch–motif pairs. The validation and test distilled fractions are 9.61% and 8.51%, respectively.

The fine pseudo-labels supervise the rule-specific fine classifier, while the fine correction values are learned jointly through the interaction objective. Because the coarse-to-fine grammar is trained after pseudo-label generation, the final learned grammar is not required to preserve the teacher clusters exactly. The pseudo-labels serve as scaffolds that encourage local subdivision of heterogeneous coarse families; final rule semantics are determined by auditing the trained coarse-to-fine model rather than by assuming that the initial clusters are fixed biological identities.

E Residual Rescue, Route Atlas, and Attribute Analyses

This appendix provides additional implementation and analysis details for the bounded residual branch, including route features, route-atlas definitions, fine-subtype routing, attribute-level route signatures, and attribute-shuffle ablations.

E.1 Residual branch and route features

After the coarse-to-fine grammar foundation is trained, all grammar-foundation parameters are frozen, including the patch encoder, motif encoder, coarse grammar, fine refiners, and auxiliary grammar heads. The residual Transformer branch is trained on detached patch and motif embeddings, so gradients from the residual objective cannot modify the explicit grammar foundation.

For each patch–motif pair, the residual branch produces a bounded correction Δ^B_{ij}, and the route gate predicts a bounded activation r_{ij}. The route gate is conditioned on grammar-side confidence features rather than serving as an unconstrained latent shortcut. In the final implementation, the route feature vector $\mathbf{g}_{ij}$ includes rule activations, grammar scores, fine-correction magnitude, top-rule fraction, rule entropy, inverse entropy, and probability confidence. These features help identify grammar-hard regions while keeping the frozen grammar field as the primary predictor.

E.2 Route-atlas definitions

Route-atlas analysis is conducted on the final PIGRAM model after the grammar foundation is frozen and the bounded residual branch is trained. For each patch–motif pair, we record the frozen grammar score H^A_{ij}, final score S^{final}_{ij}, route gate r_{ij}, bounded residual correction Δ^B_{ij}, and routed correction $r_{ij}\Delta^B_{ij}$. The final score follows Eq. 2.

To measure local prediction errors, logits are converted to probabilities using the same sigmoid scale used during training:

$$\hat{Y}^A_{ij} = \text{sigmoid}\left(\frac{H^A_{ij}}{\alpha}\right), \qquad \hat{Y}^{\text{final}}_{ij} = \text{sigmoid}\left(\frac{S^{\text{final}}_{ij}}{\alpha}\right), \tag{49}$$

where α is the sigmoid scale. The frozen-grammar and final errors are

$$e^A_{ij} = \left|\hat{Y}^A_{ij} - Y^{\text{struct}}_{ij}\right|, \qquad e^{\text{final}}_{ij} = \left|\hat{Y}^{\text{final}}_{ij} - Y^{\text{struct}}_{ij}\right|. \tag{50}$$

The pair-level error reduction is

$$\Delta e_{ij} = e^A_{ij} - e^{\text{final}}_{ij}. \tag{51}$$

Positive values of Δe_{ij} indicate improvement over the frozen grammar prediction.

Grammar-hard pairs are defined as the top quartile of e^A_{ij}, grammar-easy pairs as the bottom quartile, and high-route pairs as the top 5% of r_{ij}. These groups test whether routing is concentrated on

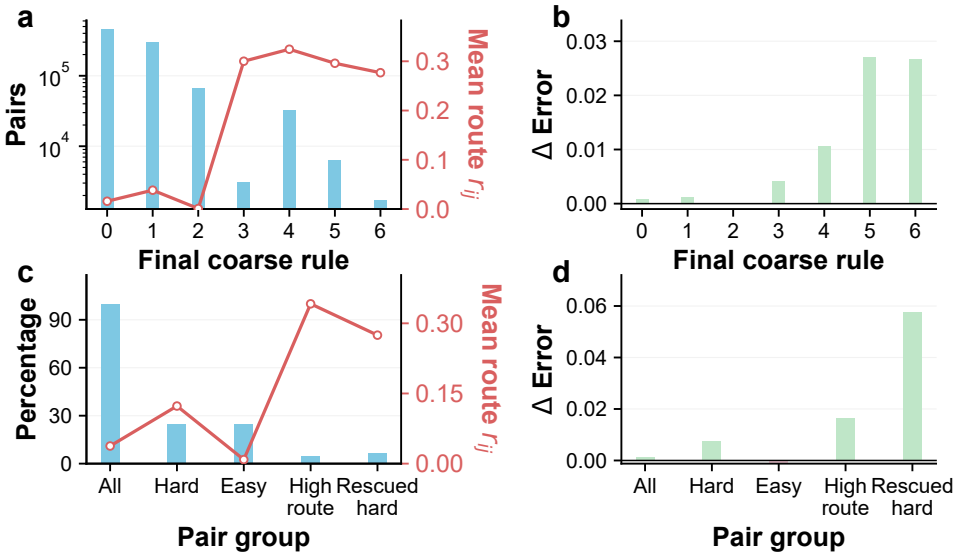


Figure 3: **Route-atlas analysis of bounded residual rescue.** The route gate is sparse overall but concentrates on grammar-hard regions. High-route pairs are enriched for hard cases, and rescued-hard pairs obtain the largest error reduction.

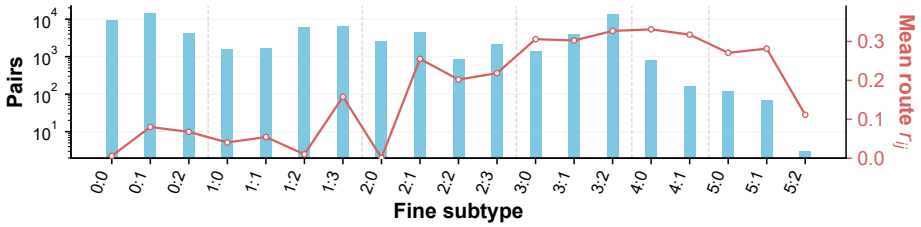


Figure 4: **Fine-subtype route atlas.** Bars show the support of each fine subtype, and the overlaid line shows the mean residual route r_{ij}. Fine subtypes are ordered by parent coarse rule. Residual routing concentrates in specific refined grammar contexts rather than being uniformly distributed across all fine subtypes.

difficult regions. We interpret r_{ij} as an activation variable for bounded residual correction, not as a guarantee that every high-route pair is individually improved.

For coarse-rule summaries, each pair is assigned to its dominant final coarse rule:

$$\rho(i,j) = \arg\max_{s,k} \left| C_{ij}^{(s,k)} \right|. \tag{52}$$

For each coarse rule, we compute support, mean route, mean correction magnitude, mean frozen-grammar error, mean final error, and mean error reduction. These summaries are shown in Figure 3.

E.3 Fine-subtype routing and attribute-level route signatures

We further analyze where the residual branch opens within the learned grammar. First, we group pairs with available fine pseudo-labels by fine subtype. This tests whether residual activation is distributed uniformly across the refined grammar or concentrated in specific local contexts. Figure 4 shows that routing is structured by fine subtype rather than uniformly spread across all refined rules.

Second, we characterize which explicit attributes are enriched in route-open regions. For a group $\mathcal{G}$ and attribute dimension d, the standardized shift is

$$z_d(\mathcal{G}) = \frac{\mathrm{mean}_{(i,j)\in\mathcal{G}}\, \phi_{ij,d} - \mathrm{mean}_{(i,j)\in\mathrm{all}}\, \phi_{ij,d}}{\mathrm{std}_{(i,j)\in\mathrm{all}}\, \phi_{ij,d} + \varepsilon}. \tag{53}$$

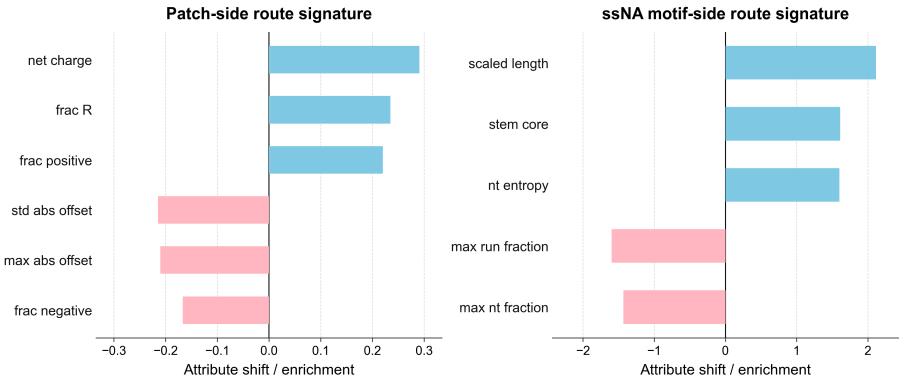


Figure 5: **Attribute-level route signatures.** Positive shifts indicate enrichment in route-open grammar-hard contexts, whereas negative shifts indicate depletion. Route activation is associated with explicit patch–motif attributes, including positively charged and arginine-rich patch environments and longer, stem-core-like, compositionally diverse ssNA motif contexts.

Table 11: Attribute-shuffle ablation of explicit grammar attributes. The ablation measures dependence on attributes, including those coupled to target construction; it does not establish independent discovery of motif preferences. Struct-AUPRC is computed by thresholding $Y^{\text{struct}} \geq 0.5$ and scoring with the final probability map.

Setting	Pearson	Spearman	AUROC	Struct-AUPRC	ΔP	ΔStruct-AUPRC
Original	**0.4961**	**0.3833**	**0.7899**	**0.3851**	0.0000	0.0000
Shuffle protein attr.	0.4697	0.3554	0.7700	0.3546	−0.0264	−0.0305
Shuffle ssNA attr.	0.1522	0.1773	0.6196	0.1495	−0.3439	−0.2356
Shuffle both sides	0.1646	0.1458	0.5975	0.1573	−0.3315	−0.2278

Here ϕ_{ij} denotes either the patch-side attribute vector $\mathbf{a}_i^{(p)}$ or the motif-side attribute vector $\mathbf{a}_j^{(m)}$ broadcast to pair (i, j). Figure 5 shows that route-open grammar-hard regions are associated with interpretable patch–motif attribute regimes.

Together, Figures 4 and 5 show that residual activation is structured rather than uniform: it concentrates within specific fine-rule contexts and explicit patch–motif attribute regimes.

E.4 Attribute-shuffle ablation

To test whether explicit attributes are functionally used by PIGRAM, we perform a test-time attribute-shuffle ablation. The trained weights are fixed, and we randomly permute either protein-side patch attributes, ssNA-side motif attributes, or both within each test sample. This preserves the marginal distribution of attributes but breaks their correspondence to the correct patch or motif.

Because this ablation evaluates the structure-aware grammar pathway, the precision–recall metric is computed by thresholding the continuous structure-aware target $Y^{\text{struct}} \geq 0.5$ and scoring with the final probability map. We denote this metric as Struct-AUPRC to distinguish it from the binary-label AUPRC reported in the main benchmark table.

Table 11 shows that explicit attributes are functionally used by the model. Shuffling protein-side attributes causes moderate degradation, while shuffling ssNA-side motif attributes or both sides causes severe degradation. These results support that PIGRAM relies on both motif-side and patch-side attributes as predictive components of the grammar.

E.5 Controls for motif-prior coupling

The motif multiplier m_j in target construction contains motif-type, pairing, GC, and positional priors that overlap with the explicit grammar inputs. Consequently, the larger effect of ssNA-attribute shuffling reflects both motif information and target-prior coupling; the shuffle table alone cannot

establish independent motif discovery. Aligned atlas components are therefore interpreted as prior-informed.

A deterministic prior-only predictor broadcasts m_j across protein patches without fitting test-set parameters. It achieves Pearson 0.262, Spearman 0.219, and Struct-AUPRC 0.227, below the corresponding unshuffled PIGRAM values in the same control evaluation (0.4961, 0.3833, and 0.3851). These are the attribute-control results, distinct from the headline benchmark's Pearson 0.4937 and distance-label AUPRC.

We also evaluate 3,478 motifs from 259 test structures, excluding motifs with zero target variance across protein patches. Within-motif correlations are macro-averaged over motifs per seed and then averaged over five seeds. Grammar Foundation achieves Pearson/Spearman $0.345/0.328$ and final PIGRAM $0.367/0.336$. The prior-only predictor is constant across patches for a fixed motif, so its within-motif correlation is undefined and it cannot rank those patches. This analysis supports additional protein-conditioned structure beyond the fixed motif multiplier, without treating prior-aligned motif patterns as independent discoveries.

F Exact training losses

This appendix provides the loss definitions used to train PIGRAM. The training procedure has three phases: coarse grammar teacher training, coarse-to-fine grammar foundation training, and frozen-grammar residual rescue training. Fine pseudo-label construction for the second phase is described separately in Appendix D.

F.1 Primary interaction loss

For a raw patch–motif logit S_{ij}, the predicted continuous interaction probability is

$$\hat{Y}_{ij} = \mathrm{sigmoid}\left(\frac{S_{ij}}{\alpha}\right), \tag{54}$$

where α is the sigmoid scale. The primary interaction loss is

$$\mathcal{L}_{\mathrm{int}} = \frac{1}{N_p N_m} \sum_{i=1}^{N_p} \sum_{j=1}^{N_m} \mathrm{Huber}_\delta\left(\hat{Y}_{ij}, Y_{ij}^{\mathrm{struct}}\right). \tag{55}$$

The same Huber form is used for the auxiliary grammar loss by setting $S_{ij} = G_{ij}^{\mathrm{coarse}}$.

F.2 Auxiliary losses

Patch importance supervision uses binary cross entropy with logits:

$$\mathcal{L}_{\mathrm{patch}} = \mathrm{BCEWithLogits}\left(\mathbf{q}, Y^{\mathrm{patch}}; \omega_{\mathrm{pos}}\right), \tag{56}$$

where $\mathbf{q}$ denotes patch-importance logits and ω_{pos} is the positive-class weight estimated from the training split.

Patch-to-motif representation learning uses an InfoNCE objective. Patch and motif embeddings are normalized, and the similarity logit is

$$\ell_{ij} = \frac{\langle \bar{\mathbf{z}}_i^{(p)}, \bar{\mathbf{z}}_j^{(m)} \rangle}{T}, \tag{57}$$

where T is the InfoNCE temperature. For each patch i, positives are defined by $Y_{ij}^{\mathrm{struct}} \geq \tau_{\mathrm{pos}}$. If at least one positive exists, the row contribution is

$$\mathcal{L}_i^{\mathrm{InfoNCE}} = -\frac{1}{|\mathcal{P}_i^+|} \sum_{j \in \mathcal{P}_i^+} \log \frac{\exp(\ell_{ij})}{\sum_{j'} \exp(\ell_{ij'})}. \tag{58}$$

The final $\mathcal{L}_{\mathrm{InfoNCE}}$ is the mean over valid rows.

Fine-rule classification is applied only to patch–motif pairs whose distillation mask is true:

$$\mathcal{L}_{\text{fine-cls}} = \frac{1}{|\Omega_{\text{fine}}|} \sum_{(i,j)\in\Omega_{\text{fine}}} \text{CE}\left(\mathbf{q}_{ij}^{(\rho(i,j))}, \tilde{y}_{ij}^{\text{fine}}\right), \tag{59}$$

where $\tilde{y}_{ij}^{\text{fine}}$ is the teacher-derived fine pseudo-label and $\mathbf{q}_{ij}^{(\rho(i,j))}$ is the fine-subrule softmax output of the fine head attached to the teacher-assigned coarse rule.

The fine grammar magnitude penalty is

$$\mathcal{L}_{\text{fine-L2}} = \text{mean}_{i,j}\left[\left(G_{ij}^{\text{fine}}\right)^2\right]. \tag{60}$$

Positive-head load balancing is applied to positive coarse-rule usage. Let $C_{ij}^{\text{pos},k}$ denote positive coarse-rule contributions and define

$$R_{ij}^{\text{pos},k} = \text{ReLU}\left(C_{ij}^{\text{pos},k}\right). \tag{61}$$

For high-target pairs with nonzero positive mass, normalized positive-rule usage is

$$\pi_{ij}^{(k)} = \frac{R_{ij}^{\text{pos},k}}{\sum_\ell R_{ij}^{\text{pos},\ell} + \varepsilon}. \tag{62}$$

The batch-level usage vector is denoted by f_k, and the balancing loss is

$$\mathcal{L}_{\text{CV2}} = K_{\text{pos}} \sum_{k=1}^{K_{\text{pos}}} f_k^2 - 1. \tag{63}$$

F.3 Phase-specific objectives

Phase 1: coarse grammar teacher. The pure coarse teacher predicts directly with the coarse grammar score. Its objective is

$$\mathcal{L}_{\text{coarse}} = \lambda_{\text{int}}\mathcal{L}_{\text{int}}\left(G^{\text{coarse}}, Y^{\text{struct}}\right) + \lambda_{\text{patch}}\mathcal{L}_{\text{patch}} + \lambda_{\text{NCE}}\mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{CV2}}(t)\mathcal{L}_{\text{CV2}}. \tag{64}$$

Because the pure coarse teacher has no separate final interaction head, the interaction loss directly supervises the coarse grammar score.

Phase 2: coarse-to-fine grammar foundation. The foundation objective supervises the grammar-centered score defined in Eq. 1, while retaining an auxiliary loss on the coarse grammar score:

$$\begin{aligned}\mathcal{L}_{\text{foundation}} =& \lambda_{\text{int}}\mathcal{L}_{\text{int}}\left(H^A, Y^{\text{struct}}\right) + \lambda_{\text{grammar}}\mathcal{L}_{\text{grammar}}\left(G^{\text{coarse}}, Y^{\text{struct}}\right) + \lambda_{\text{patch}}\mathcal{L}_{\text{patch}} \\ &+ \lambda_{\text{NCE}}\mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{fine-cls}}\mathcal{L}_{\text{fine-cls}} + \lambda_{\text{fine-L2}}\mathcal{L}_{\text{fine-L2}} + \lambda_{\text{CV2}}(t)\mathcal{L}_{\text{CV2}}.\end{aligned} \tag{65}$$

After this phase, the grammar foundation is frozen.

Phase 3: frozen-grammar residual rescue. In the final phase, only the residual Transformer branch and the rescue gate are trainable. The final prediction follows Eq. 2. The rescue objective uses the frozen-grammar error to focus optimization on grammar-hard pairs. The frozen and final probabilities are

$$\hat{Y}_{ij}^A = \text{sigmoid}\left(\frac{H_{ij}^A}{\alpha}\right), \qquad \hat{Y}_{ij}^{\text{final}} = \text{sigmoid}\left(\frac{S_{ij}^{\text{final}}}{\alpha}\right). \tag{66}$$

The detached frozen-grammar error is

$$e_{ij}^A = \left|\hat{Y}_{ij}^A - Y_{ij}^{\text{struct}}\right|. \tag{67}$$

The error-aware weight is

$$\omega_{ij} = \frac{\epsilon + (e_{ij}^A)^\gamma}{\text{mean}_{i,j}\left[\epsilon + (e_{ij}^A)^\gamma\right] + \varepsilon}. \tag{68}$$

The residual prediction loss is

$$\mathcal{L}_{\text{pred}} = \text{mean}_{i,j}\left[\omega_{ij}\,\text{Huber}_{\delta}\left(\hat{Y}_{ij}^{\text{final}}, Y_{ij}^{\text{struct}}\right)\right]. \tag{69}$$

The full residual rescue objective is

$$\begin{aligned}\mathcal{L}_{\text{rescue}} =& \mathcal{L}_{\text{pred}} + \lambda_{\text{sparse}}\,\text{mean}_{i,j}\left[r_{ij}\right] + \lambda_{\text{corr}}\,\text{mean}_{i,j}\left[\left(r_{ij}\Delta_{ij}^{B}\right)^2\right] \\ &+ \lambda_{\text{budget}}\left[\text{ReLU}\left(\text{mean}_{i,j}[r_{ij}] - \tau_{\text{budget}}\right)\right]^2 + \lambda_{\text{cap}}\,\text{mean}_{i,j}\left[\text{ReLU}\left(r_{ij} - \tau_{\text{cap}}\right)^2\right] \\ &+ \lambda_{\text{easy}}\,\text{mean}_{(i,j)\in\mathcal{E}}\left[\text{ReLU}\left(e_{ij}^{\text{final}} - e_{ij}^{A}\right)^2\right],\end{aligned} \tag{70}$$

where

$$e_{ij}^{\text{final}} = \left|\hat{Y}_{ij}^{\text{final}} - Y_{ij}^{\text{struct}}\right|, \tag{71}$$

and $\mathcal{E}$ denotes grammar-easy pairs, defined as the lowest-error quartile of e_{ij}^{A} within the batch.

The sparse and budget terms limit overall residual usage, the correction penalty discourages unnecessarily large applied corrections, the cap term prevents near-complete local takeover, and the easy no-harm term discourages the residual branch from degrading pairs already handled well by the explicit grammar. In the final PIGRAM model, the residual gate is bounded by $r_{\max} = 0.60$ in Eq. 11.

G Full Coarse-to-Fine Grammar Atlas

This appendix reports the full rule-level interpretation of the trained coarse-to-fine PIGRAM grammar. Coarse-rule summaries are computed relative to global high-response patch–motif pairs, whereas fine-subtype summaries are computed relative to the corresponding parent coarse rule. Thus, coarse rules describe global signed compatibility basins, while fine subtypes describe within-basin heterogeneity.

Throughout the tables below, motif-type, pairing, GC, and positional signatures aligned with the supervision multiplier are *prior-informed*. Biological names and literature-based rationales are assigned after training and should be read as interpretations of learned compatibility regimes, not independent discovery or proof of atomic mechanisms. The within-motif control in Appendix E.5 tests protein-conditioned signal beyond the fixed motif prior.

G.1 Final coarse-rule atlas

Table 12 reports the final coarse-rule atlas. Rules 0–2 are suppressive and Rules 3–6 are favorable; support counts are measured on high-response pairs.

Table 12: Complete final coarse-rule atlas. Rules 0–2 are suppressive and Rules 3–6 are favorable. Counts are measured on high-response pairs. Motif associations aligned with the target-construction multiplier are prior-informed.

Rule	Sign	Support	Motif context	Patch context	Rule interpretation
0	Supp.	327,869	Mixed; RNA-majority	Compact; mild negative-charge shift	Mixed RNA suppressive basin
1	Supp.	377,572	Mixed; DNA-enriched	Low charge; low arginine; compact	Compact DNA suppressive basin
2	Supp.	62,812	Stem-edge; DNA-dominant	High-offset topology	Topology-dependent suppression
3	Fav.	32,184	Stem-core; DNA-dominant	Arginine-rich; extended geometry	High-confidence favorable expert
4	Fav.	272,792	Stem-core; DNA-canonical	Moderate offset; broad geometry	Major canonical favorable basin
5	Fav.	21,783	Tail-unpaired; RNA-majority	Local topology; low-pairing context	Rare DNA-shifted tails
6	Fav.	5,625	Tail-unpaired; RNA-enriched	Strongly local topology	Coarse-only transitional family

The coarse atlas separates broad suppressive basins from favorable compatibility families. Rules 0–2 capture suppressive regimes, ranging from broad mixed RNA/DNA contexts to the sharper high-offset stem-edge/DNA regime in Rule 2. Rules 3–6 capture favorable regimes, including the arginine-rich stem-core/DNA expert in Rule 3, the broader canonical stem-core/DNA basin in Rule 4, and rare tail-unpaired families in Rules 5–6. Rule 5 is further refined into DNA-shifted minority submodes, whereas Rule 6 remains coarse-only.

G.2 Final fine-subtype atlas

The final grammar refines Rules 0–5, with rule-specific numbers of fine heads $K_0 = 3$, $K_1 = 4$, $K_2 = 4$, $K_3 = 3$, $K_4 = 2$, and $K_5 = 3$. Rule 6 is not assigned a fine head. Table 13 summarizes the resulting fine-subtype interpretations.

Table 13: Summary of final fine-subtype interpretations. Fine-subtype signatures are interpreted relative to the corresponding parent coarse rule. Motif associations aligned with the target-construction multiplier are prior-informed.

Fine subtype	Motif context	Patch context	Fine interpretation
Rule 0.0	Stem-edge; DNA-enriched	Local edge-conflict	Stem-edge suppressive subtype
Rule 0.1	Mixed stem-core/DNA-like	Mild parent-like shift	Mixed suppressive subtype
Rule 0.2	Weak motif contrast	Cation-π; distance-enriched	Patch-driven suppressive subtype
Rule 1.0	Tail-unpaired; RNA-shifted	Local; compact	Tail/RNA suppressive subtype
Rule 1.1	Stem-edge; DNA-like	Distance; cation-π-enriched	Geometry-enhanced penalty subtype
Rule 1.2	Stem-edge; DNA-dominant	Low offset/locality contrast	Stem-edge suppressive subtype
Rule 1.3	Stem-core; DNA-dominant	Weak parent-relative shift	Near-neutral stem-core subtype
Rule 2.0	Stem-edge; DNA-like	Parent-like high-offset topology	Parent-like suppressive subtype
Rule 2.1	Stem-core; DNA-like	Shift away from stem-edge penalty	Stem-core rescue subtype
Rule 2.2	Mixed stem-core/DNA	Residually suppressive	Mixed suppressive subtype
Rule 2.3	Tail-unpaired; RNA-shifted	Local rescue-like shift	Near-neutral tail/RNA subtype
Rule 3.0	Tail-unpaired; RNA rescue	Noncanonical relative to parent	Rare rescue subtype
Rule 3.1	Stem-core; DNA-strong	High offset; extended geometry	Strong favorable subtype
Rule 3.2	Stem-core; DNA-moderate	Moderate geometry shift	Moderate favorable subtype
Rule 4.0	Stem-core; DNA-canonical	Broad canonical context	Major favorable subtype
Rule 4.1	Stem-core; DNA-shifted	Extended; distance-enriched	Geometry-resolved favorable subtype
Rule 5.0	Tail-unpaired; moderate DNA shift	Local; compact	Local tail subtype
Rule 5.1	Tail-unpaired; stronger DNA shift	Cation-π; aromatic; distance-enriched	Chemistry-enhanced tail subtype
Rule 5.2	Rare stem-core/DNA minority	Distance; cation-π-enriched	Rare stem-core rescue subtype

The fine atlas shows that selective refinement is not merely motif relabeling. Some fine heads separate motif contexts, such as stem-edge, stem-core, and tail-unpaired subfamilies, but many distinctions are driven by patch-side properties such as offset spread, local topology, distance scale, arginine fraction, and cation-π-related proxies. Rule 5 illustrates this point: although the parent rule is tail-unpaired and RNA-majority, its fine subtypes expose structured DNA-shifted minority modes, including a rare stem-core/DNA high-response subtype.

G.3 Biological rationale for grammar rules

The learned grammar rules are not intended to be one-to-one textbook categories. Instead, they define data-driven compatibility regimes that are consistent with known principles of protein–nucleic-acid recognition. Table 14 summarizes the biological rationale for each coarse rule and fine subtype.

Overall, these rationales support a conservative interpretation of the grammar atlas: the learned rules recover compatibility regimes and penalty states consistent with known protein–nucleic-acid recognition principles, rather than externally named mechanistic classes.

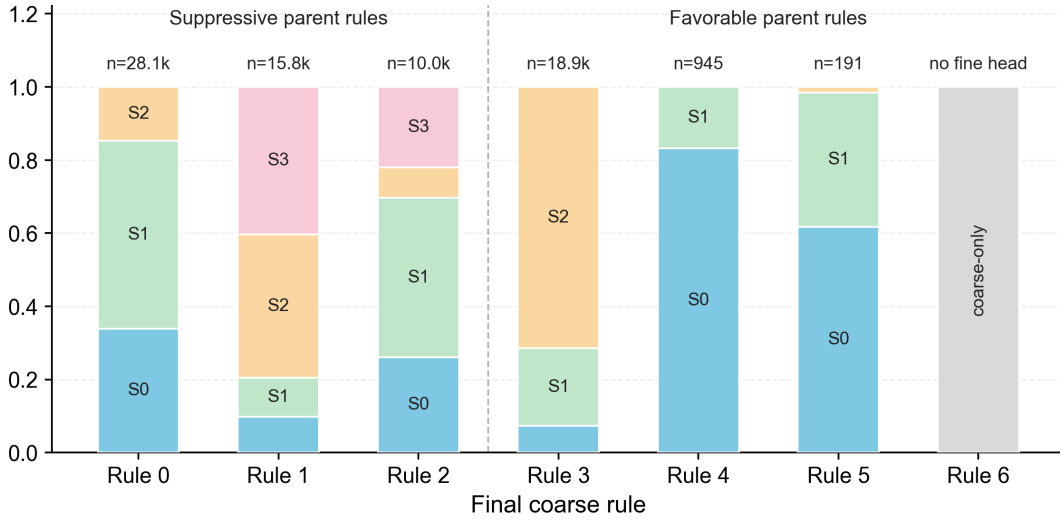


Figure 6: **Selective fine-rule subdivision.** Fine-subtype composition of high-response patch–motif pairs within each final coarse rule. Refined parent rules split into local subtypes to resolve hidden heterogeneity, while Rule 6 remains coarse-only.

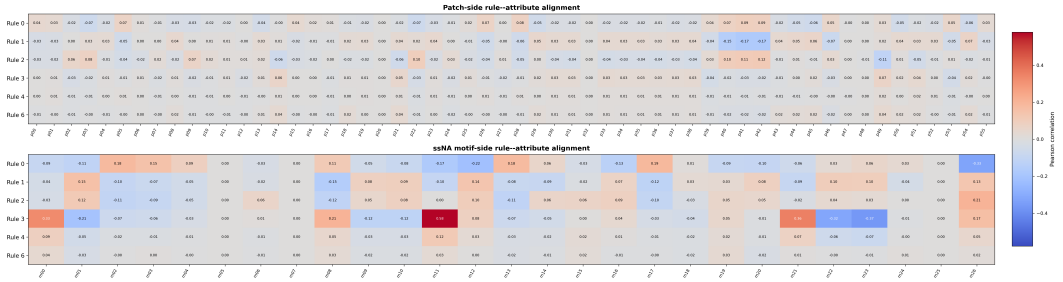


Figure 7: **Rule–attribute alignment heatmaps.** Rows denote final grammar rules and columns denote explicit attributes. Entries are Pearson correlations between rule signals and patch-side (`p*`) or ssNA motif-side (`m*`) attributes over test patch–motif pairs.

G.4 Rule–attribute alignment

To test whether the learned grammar aligns with interpretable descriptors, we compute rule–attribute correlations over evaluated test-set patch–motif pairs. For each grammar rule k and explicit attribute dimension d, we compute

$$\rho_{k,d} = \mathrm{corr}\left(z^{\mathrm{rule}}_{ij,k}, a_{ij,d}\right), \tag{72}$$

where $z^{\mathrm{rule}}_{ij,k}$ is the exported rule signal for patch–motif pair (i, j). When continuous rule contributions are available, $z^{\mathrm{rule}}_{ij,k}$ is the rule activation or contribution; otherwise, it is the one-hot dominant-rule assignment. For patch-side attributes, $a_{ij,d}$ is obtained by broadcasting the patch attribute of patch i across motifs. For motif-side attributes, $a_{ij,d}$ is obtained by broadcasting the motif attribute of motif j across protein patches.

Figure 7 provides a global diagnostic of the learned grammar. Because correlations are computed over test-set patch–motif pairs, the heatmaps should be interpreted as statistical alignment rather than causal attribution. The observed structure indicates that different rules emphasize distinct patch-side and motif-side descriptors, supporting the interpretation that PIGRAM learns an attribute-linked grammar rather than arbitrary latent scoring directions.

Table 14: **Biological rationale for learned grammar rules.** Rules 0–2 are mainly interpreted as suppressive inverse states, whereas Rules 3–6 capture favorable or rescue-like recognition modes.

Rule	Biological rationale / citations
Rule 0	Broad suppressive RNA/mixed basin. Productive RNA recognition requires exposed bases, local pockets, stacking, hydrogen bonding, and multivalent electrostatics; weak or misaligned contacts form a plausible inverse state. [80–82].
Rule 0.0	Stem-edge suppressive state. Edge-conflict geometry can block productive DNA groove readout and close-fit packing even for DNA-enriched motifs. [83].
Rule 0.1	Mixed DNA-like suppressive state. Partial stem-core/DNA character without strong Arg/Lys enrichment, electrostatic support, or extended geometry may not support stable binding. [84].
Rule 0.2	Patch-driven suppressive state. Cation–π or distance-enriched patch chemistry is insufficient when motif contrast or base exposure is weak. [85].
Rule 1	Compact DNA-enriched suppressive basin. Low positive charge, low arginine, and compact geometry are plausible inverse states of canonical DNA-binding interfaces. [83].
Rule 1.0	Tail/RNA suppressive state. Flexible unpaired RNA regions are recognizable, but compact local patches may lack sufficient multivalent contacts. [86].
Rule 1.1	Geometry-enhanced penalty state. Chemical attraction such as cation–π is present, but the DNA-like motif remains geometrically misregistered. [80, 83].
Rule 1.2	Stem-edge suppressive DNA state. Boundary-like DNA contexts lack the continuous groove chemistry and registration typical of canonical stem-core readout. [87, 88].
Rule 1.3	Near-neutral stem-core DNA state. The motif is compatible, but weak patch specialization limits transition into a strong favorable expert. [83].
Rule 2	Topology-dependent suppressive basin. Stem-edge DNA with high-offset topology can disrupt groove access, surface complementarity, and backbone contacts. [83].
Rule 2.0	Parent-like suppressive subtype. High-offset stem-edge topology makes productive groove readout and backbone salt-bridge formation difficult. [83].
Rule 2.1	Stem-core rescue subtype. Shifting from stem-edge toward stem-core can restore canonical DNA groove readout, DNA-shape recognition, and phosphate-backbone contacts. [83, 89].
Rule 2.2	Residual mixed suppressive subtype. Partial stem-core/DNA character remains insufficient without strong charge, Arg/Lys contacts, or favorable geometry. [90].
Rule 2.3	Near-neutral tail/RNA rescue subtype. Exposed tail, loop, or bulge elements can provide local stacking and hydrogen-bond opportunities that partially rescue suppressive states. [80, 91].
Rule 3	Arginine-rich stem-core/DNA favorable expert. Consistent with groove insertion, DNA shape/electrostatic readout, and phosphate-backbone contacts. [83, 89].
Rule 3.0	Rare RNA-tail rescue subtype. Noncanonical RNA loops, bulges, or tails can be stabilized through exposed bases, arginine forks, and local backbone/base contacts. [82, 92].
Rule 3.1	Strong stem-core/DNA favorable subtype. Extended geometry and strong DNA context match high-confidence DNA-recognition modes. [83, 89, 93].
Rule 3.2	Moderate stem-core/DNA favorable subtype. Follows the same canonical DNA-recognition axis as Rule 3.1, but with weaker geometric specialization. [83, 89].
Rule 4	Major canonical stem-core/DNA favorable basin. Consistent with major-groove base readout, minor-groove shape readout, phosphate contacts, and interface packing. [83, 87, 89].
Rule 4.0	Canonical favorable subtype. Matches broad protein–DNA recognition through groove chemistry, shape readout, and backbone contacts. [83, 87, 89].
Rule 4.1	Geometry-resolved favorable subtype. DNA-shifted states can remain favorable through indirect readout, DNA deformability, local/global shape, and distance-sensitive contacts. [83, 89].
Rule 5	Tail-unpaired favorable family. ssDNA/RNA tails, loops, bulges, and overhangs expose bases for local pocket binding, stacking, and backbone contacts. [94, 95].
Rule 5.0	Local tail subtype. Compact pockets can recognize ssDNA tails or overhangs through exposed bases and phosphate-backbone contacts. [96, 97].
Rule 5.1	Chemistry-enhanced tail subtype. Aromatic stacking, cation–π, π–π, and Arg/Lys-backbone contacts can stabilize exposed nucleobases. [94, 98, 99].
Rule 5.2	Rare stem-core rescue subtype. Tail-like aromatic or cation–π chemistry can rescue minority stem-core/DNA states when local deformation or base exposure permits nucleobase contacts. [93, 98].
Rule 6	Coarse-only transitional tail-unpaired family. Local topology and exposed RNA/ssNA bases can support minority favorable interactions, although no resolved fine subtype is assigned. [82, 86, 97].

H Downstream Readout Selection

This appendix provides the candidate-level readout protocol used for downstream aptamer screening. The main controlled downstream comparison in Section 4.5 uses internal baselines evaluated on the same processed candidate pools with the same validation-selected readout protocol.

For each candidate aptamer A, each model produces a patch–motif score matrix $Z(A) \in \mathbb{R}^{N_p \times N_m}$. We convert this matrix into a candidate-level score using a top-K readout:

$$s_K(A) = \frac{1}{K} \sum_{(i,j) \in \mathrm{TopK}(Z(A))} Z_{ij}, \tag{73}$$

where $\mathrm{TopK}(Z(A))$ denotes the K largest patch–motif entries. We sweep $K \in \{1, 3, 5, 10, 20, 50\}$, with $K = 1$ corresponding to a max-response readout.

Because the affinity direction is not uniform across the raw downstream files, both the affinity direction and candidate-level readout are selected using the validation split only. The selected direction and readout are then fixed for sampled-protocol evaluation. The main selection criterion is

$$\mathrm{Score}_{\mathrm{select}} = 0.4\,\mathrm{AUPRC} + 0.3\,\mathrm{Top10} + 0.3\,\mathrm{Top50}. \tag{74}$$

This criterion balances global ranking quality with early-retrieval performance.

I Downstream grammar-guided aptamer validation design

This appendix describes the design of the downstream validation experiments in Section 4.5. The goal was to test whether learned patch–motif grammar rules can generate directional mutation hypotheses with measurable effects on apparent affinity.

I.1 Design objective

As defined in Eq. 2, the final interaction score is decomposed into a grammar foundation and a bounded residual correction:

$$S_{ij}^{\mathrm{final}} = H_{ij}^{A} + r_{ij}\Delta_{ij}^{B}.$$

For mutation design, we focused on the grammar foundation H^A, because it assigns local protein patch–aptamer motif pairs to signed grammar rules. A suppressive rule was treated as a negative design element whose relief may improve apparent affinity, whereas a favorable rule was treated as a positive design element whose disruption should weaken apparent affinity.

I.2 Selection of validation targets and backbones

We selected two complementary validation systems from the downstream grammar-guided scan.

GFP suppressive Rule 0. For GFP, the strongest reproducible signal was a recurrent GC-rich stem-core motif assigned to suppressive Rule 0. Several high-ranked candidates contained related stem-core motifs around nucleotide positions 65–80, with high predicted pairing and elevated GC content. We therefore used GFP to test whether relief of a suppressive grammar context can improve apparent affinity.

Two GFP backbones, GFP_603 and GFP_1434, were selected because both contained strong recurrent Rule 0 stem-core motifs in the same local region. For each backbone, we tested five variants:

$$\{\mathrm{WT}, \mathrm{SP}, \mathrm{SD}, \mathrm{R0}, \mathrm{RC}\}.$$

The expected direction was lower apparent K_D for R0 relative to WT. SD could also lower apparent K_D if the original paired stem-core state acts as a suppressive structural constraint, whereas SP and RC were expected to remain closer to WT.

NELF favorable Rule 4. NELF provided the complementary favorable-rule case. The scan identified clean favorable Rule 4 stem-core candidates, including NELF_6260 and NELF_4894. Both were assigned to favorable Rule 4 contexts and were associated with positive or arginine-enriched protein patches. We therefore used NELF to test whether disruption of a favorable grammar context weakens apparent affinity.

For each NELF backbone, we tested five variants:

$$\{\mathrm{WT}, \mathrm{SP}, \mathrm{SD}, \mathrm{R4}, \mathrm{RC}\}.$$

The expected direction was higher apparent K_D for SD relative to WT. R4 was expected to retain the favorable grammar context better than RC, while SP tested whether approximate structure preservation alone was sufficient to retain binding.

I.3 Variant definitions

For each selected backbone, mutations were introduced only in the grammar-nominated motif region while keeping the remaining aptamer sequence fixed. Table 15 summarizes the variant classes.

Table 15: Variant definitions for downstream grammar-guided validation.

Variant	Used in	Design role
WT	GFP, NELF	Wild-type reference; the original candidate aptamer used as the matched backbone baseline.
SP	GFP, NELF	Structure-preserving mutant; changes local motif sequence while approximately preserving stem-core pairing.
SD	GFP, NELF	Structure-disrupting mutant; reduces local stem-core pairing, typically by converting selected G/C positions to A/U.
R0	GFP	Rule 0-relief mutant; reduces the GC-rich or paired suppressive Rule 0 signature.
R4	NELF	Rule 4-conservative mutant; changes sequence identity while attempting to preserve the favorable Rule 4 context.
RC	GFP, NELF	Composition-matched random control; controls for mutation burden and base composition without preserving the grammar rule.

SD is a broader structural perturbation, whereas R0 and R4 are grammar-directed perturbations. R0 is designed to relieve a suppressive signature. R4 is designed to retain a favorable signature while changing sequence identity. RC controls for whether observed affinity changes can be explained by mutation burden or base composition alone.

I.4 Secondary-structure validation before ordering

Before ordering, mutation designs were screened by secondary-structure prediction. For each design, we inspected the predicted full-sequence fold and the local motif-region dot-bracket pattern. We also tracked motif pairing ratio, GC fraction, and homopolymer/run features. Designs that failed the intended structural criterion were excluded or flagged for manual review.

I.5 Affinity assay and statistical analysis

Apparent K_D values were measured by fluorescence ELISA using fluorescence-labeled aptamers. Each sequence was measured in five replicates. Values are reported as mean $\pm$ standard deviation.

For each backbone, each mutant variant was compared against the matched WT sequence using a two-sided Welch's t-test. Significance was denoted as:

$$^{*}p < 0.05, \qquad ^{**}p < 0.01, \qquad ^{***}p < 0.001.$$

These tests evaluate whether a local perturbation changes apparent affinity relative to its own backbone reference.

For cross-backbone summaries, we also computed normalized apparent affinity:

$$\widetilde{K}_{D,b,v,r} = \frac{K_{D,b,v,r}}{K_{D,b,\mathrm{WT},r}},$$

where b denotes the backbone, v denotes the variant class, and r denotes the replicate index. Values below 1 indicate stronger binding than WT; values above 1 indicate weaker binding than WT.

I.6 Raw replicate affinity measurements

Table 16 lists the raw apparent K_D replicate measurements used for the main validation table.

I.7 Normalized cross-backbone summaries

Table 17 summarizes normalized apparent K_D changes across the two tested backbones for each target. These summaries complement the matched-backbone tests reported in the main text and assess whether each variant class moves consistently across independent backbones.

For GFP, SP showed a nominal decrease in GFP_603 alone but did not replicate across backbones in the normalized analysis, supporting the interpretation that SP does not produce a systematic grammar-directed shift. R0 and SD both reduced normalized K_D, with R0 showing the larger decrease. A

Table 16: Raw apparent K_D replicates for grammar-guided validation. GFP tests suppressive Rule 0 relief; NELF tests favorable Rule 4 perturbation. All apparent K_D measurements are in nM.

A. GFP Rule 0 validation

Seq.	R1	R2	R3	R4	R5
603_WT	655	683	678	674	689
603_SP	655	645	661	646	653
603_SD	554	534	544	556	538
603_R0	492	472	501	503	482
603_RC	669	654	647	655	659
1434_WT	678	656	643	664	654
1434_SP	657	643	685	655	664
1434_SD	599	578	597	573	583
1434_R0	512	521	506	515	527
1434_RC	643	645	674	655	653

B. NELF Rule 4 validation

Seq.	R1	R2	R3	R4	R5
6260_WT	297	278	295	278	287
6260_SP	349	355	345	365	345
6260_SD	532	567	578	576	565
6260_R4	320	329	335	330	335
6260_RC	389	378	386	397	385
4894_WT	442	436	446	441	432
4894_SP	488	498	491	482	495
4894_SD	699	678	702	674	689
4894_R4	468	478	486	481	483
4894_RC	501	512	532	522	528

Table 17: Cross-backbone normalized apparent K_D changes for grammar-guided validation. Values below 1 indicate stronger binding than matched WT; values above 1 indicate weaker binding. Normalized K_D is dimensionless; ΔK_D is in nM. Values are mean $\pm$ sample SD over the ten displayed ratios pooled across two backbones. The reported p values compare normalized values with 1 using two-sided one-sample t-tests.

Target	Variant	Normalized K_D	Mean ΔK_D	p vs 1
GFP	SP	0.984 ± 0.036	-11.0 ± 23.9	0.199
GFP	SD	0.848 ± 0.050	-101.8 ± 35.3	5.0×10^{-6}
GFP	R0	0.754 ± 0.038	-164.3 ± 28.8	7.7×10^{-9}
GFP	RC	0.983 ± 0.032	-12.0 ± 21.6	0.125
NELF	SP	1.172 ± 0.074	58.1 ± 14.7	4.3×10^{-5}
NELF	SD	1.766 ± 0.223	262.8 ± 23.0	1.8×10^{-6}
NELF	R4	1.120 ± 0.046	41.3 ± 10.1	1.6×10^{-5}
NELF	RC	1.265 ± 0.097	89.8 ± 16.0	1.2×10^{-5}

direct comparison of normalized values confirmed that R0 reduced apparent K_D more than SD (Welch's two-sided t-test, $p = 2.0 \times 10^{-4}$).

For NELF, SD produced the largest normalized K_D increase, consistent with disruption of a favorable Rule 4 context weakening binding. R4 remained closer to WT than RC and SD across the two backbones, whereas SP showed a smaller and partially overlapping difference from R4. The R4–RC separation was significant in normalized analysis (Welch's two-sided t-test, $p = 9.3 \times 10^{-4}$), supporting that composition matching alone does not preserve the favorable Rule 4 effect. R4 was also lower than SP under a paired normalized comparison across matched replicate indices ($p = 0.0029$), although the more conservative unpaired Welch comparison was not significant ($p = 0.079$). We therefore interpret R4 as preserving WT-like binding better than composition-matched and disruption-oriented controls, while treating the R4–SP separation as directionally consistent but smaller.

J Training cost

We report the wall-clock cost of the staged PIGRAM training pipeline in Table 18. Runtime was measured on a single NVIDIA RTX A6000 GPU and includes training, validation, checkpointing, and model export unless otherwise noted. The staged pipeline adds overhead relative to a single end-to-end baseline, but this cost is paid once to obtain the frozen grammar atlas, fine subtypes, route diagnostics, and downstream design outputs.

K Reproducibility and Resource Details

Model comparisons are evaluated over five training seeds, and downstream evaluations use validation-selected candidate-level readouts. All reported training-cost measurements were obtained on a single NVIDIA RTX A6000 GPU with 48GB memory. Existing datasets and external benchmarks are credited through citations to their original sources.

Table 18: Training cost of the staged PIGRAM pipeline. Runtime is wall-clock time on a single NVIDIA RTX A6000 GPU.

Stage	Training objective	Epoch time	Total runtime
Stage 1	Sign-separated coarse grammar teacher	523 s/epoch	10.2 h
Stage 2	Coarse-to-fine grammar foundation	567 s/epoch	11.0 h
Stage 3	Frozen-grammar residual rescue	589 s/epoch	11.5 h
Total	Full PIGRAM training pipeline	–	32.7 h

L Additional comparisons and residual-contribution analysis

L.1 Additional aptamer–protein baselines

We additionally compare AptaTrans [100], AptaNet [101], and APIPred [102] on the five downstream target pools. AptaTrans uses the published pretrained encoders with independently trained API predictors; AptaNet and APIPred are retrained with their original architectures and datasets. Trainable components are reinitialized or refitted for each of five seeds. The overlap audit found no exact evaluation-target protein or candidate aptamer sequence in the baseline training sets used for this comparison. Because training datasets and negative-sample constructions differ, these results are application-level comparisons rather than controlled comparisons of model architecture.

Table 19: Additional downstream baseline comparisons. Values are five-seed means macro-averaged over five targets; this summary displays point estimates.

Model	Top10	Top50	AUPRC	AUROC
AptaTrans	0.323	0.196	0.215	0.576
AptaNet	0.282	0.187	0.221	0.543
APIPred	0.301	0.186	0.242	0.583
Spatial GNN	0.322	0.238	0.246	0.608
PIGRAM	0.430	0.245	0.314	0.641

L.2 Task-matched spatial control

The Spatial GNN uses the same inputs, motif encoder, supervision, split, loss, and evaluation protocol as PIGRAM. Its protein encoder uses three edge-conditioned residue-graph layers and a neural patch–motif scoring head, without grammar rules or residual routing. Paired confidence intervals use 10,000 shared sequence-pair bootstrap resamples. The Pearson, AUROC, and AUPRC differences favor PIGRAM; the Spearman difference is unresolved.

Table 20: Task-matched structural comparison: PIGRAM minus Spatial GNN.

Metric	Difference	95% CI
Pearson	0.0334	$[0.018, 0.049]$
Spearman	0.0155	$[-0.001, 0.031]$
AUROC	0.0292	$[0.017, 0.042]$
AUPRC	0.0114	$[0.002, 0.021]$

L.3 Exact correction and attribution audit

The exact residual correction is $c_{ij} = r_{ij}\Delta B_{ij}$. Across the 259 original-split test structures, its magnitude exceeds the frozen grammar magnitude in 3.2% of cells, and its addition changes the logit-zero classification in 1.4% of cells.

We also compute 64-step Integrated Gradients on 90 test structures, equally sampled from high-route rescued, high-route unrescued, and low-route cases. A cell is rescued when the residual reduces absolute error relative to the structure-aware target. Attribution is computed with respect to the

detached patch and motif embeddings, using training-set mean embeddings as the baseline; route values are fixed when attributing the gated correction. For the gated correction, the top 10% of route-ranked cells contain 74.5% of absolute attribution mass. Rescued cells account for 24.6% of evaluated cells and receive 71.2% of attribution mass (2.9-fold enrichment). For the ungated proposal, the corresponding fractions are 47.9% and 63.8%, with 2.6-fold enrichment in rescued cells. Thus, localization is also present before gate multiplication. These sampled attribution results complement the exact contribution audit and support localized correction rather than global takeover.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions: a patch–motif formulation, a sign-separated coarse-to-fine grammar, frozen bounded residual rescue, and strict pair-disjoint evaluation. These claims are supported by the method and experiments, including baseline comparisons, grammar-atlas analyses, route-atlas analyses, and downstream screening.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a Conclusion and Limitations section discussing dependence on structural inputs and derived supervision, the limits of patch–motif resolution relative to full atomic physics, and the interpretation of the grammar atlas as a data-driven compatibility map rather than a complete physical law.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results or proofs. The mathematical content consists of model definitions, scoring functions, training objectives, and evaluation metrics.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper describes the model architecture, pair-disjoint split protocol, patch and motif construction, label construction, training phases, evaluation metrics, and downstream readout selection. Additional implementation details are provided in the appendices.

   Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: A public project repository is available at `https://github.com/yhan1127/PIGRAM`. The supplementary package includes preprocessing, training, evaluation, and downstream screening scripts. The paper and appendices describe the required data construction, splits, model components, and evaluation protocol.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper and appendices describe the data splits, model architecture, training objectives, and evaluation protocols. Training hyperparameters and optimizer settings are provided in the public GitHub repository at `https://github.com/yhan1127/PIGRAM`.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

Justification: Main model-comparison subscripts report SD over five training seeds. Appendix A.7 reports 95% pair-clustered bootstrap confidence intervals using 10,000 shared resamples. Appendix I describes the affinity-assay summaries and statistical tests.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The appendix reports the compute environment used for training and evaluation, including GPU type, GPU memory, repeated training seeds, and staged training runtime

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research uses publicly available biomolecular structure and aptamer benchmark data and does not involve human subjects, private personal data, or deployment on sensitive populations. The work is intended for interpretable biomolecular modeling and candidate prioritization.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [No]

    Justification: The paper discusses potential applications and methodological limitations, but does not explicitly discuss potential negative societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a high-risk foundation model, scraped personal dataset, generative media model, or system with immediate misuse risk. The method is a structure-aware scoring model for biomolecular interaction analysis

    Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The original sources of the structural datasets, aptamer benchmarks, and external methods are cited. PDB archive data are available under CC0 1.0.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [Yes]

Justification: The released implementation is documented through the README, environment specifications, configuration files, and accompanying pipeline and reproduction guides. These materials describe the code structure, data preprocessing, training, and evaluation procedures.

Guidelines:

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

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

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

Answer: [N/A]

Justification: This study does not involve crowdsourcing or research with human subjects.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The reported research does not involve human participants, so human-subjects ethical review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLM assistance was limited to writing, editing, or formatting and was not part of the core research methodology.

    Guidelines:

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