

RLMA: Reinforcement Learning with PolyMetric Alignment

Jia Xu
Rutgers University

Abstract

RLHF, RLAIF, and RLVR align LLMs with human instructions but rely on costly human labels, potentially biased LLM judges, or task-specific verifiers. We propose PolyMetric Alignment (RLMA), a reference-based reward framework for supervised-answer tasks that fuses heterogeneous evaluation metrics against a given reference answer using a Residual Set Transformer (RST) trained with a Bradley–Terry objective. Empirically, RLMA is stronger than all three paradigms, achieving 98.0% of the Pass@K oracle on a 9-pool benchmark, with gains of +17.6 pp over the RLA single-metric baseline and +7.4 to +15.4 pp over RLHF, RLAIF, and RLVR. RLMA also generalizes broadly, reaching 97.2% of oracle on unseen GSM8K and, when used as a DPO reward, improving TriviaQA Pass@1 by +4.80 pp on Qwen-2.5-7B-Instruct. Finally, RLMA is efficient to deploy. Its 113K-parameter gate requires only 0.33 ms per candidate, roughly 300× faster than a single 8B reward-model forward.

1 Introduction

Reinforcement learning from human feedback (Ouyang et al., 2022; Bai et al., 2022), its LLM-judge variant RLAIF (Lee et al., 2024), and executable-verifier RLVR (Lambert et al., 2024) all require supervision external to the source dataset, namely human labels, judge biases, or task-specific checkers. Yet on the broad class of reference-available tasks such as question answering, hallucination detection, and mathematical reasoning, each query comes with a ground-truth answer supplied by the dataset itself but is not used in RL alignment. Recent work (Lins and Xu, 2025) constructs a reward from a single reference-similarity metric, but generation quality is multi-dimensional and a single reward invites hacking behaviours documented in the RLHF literature (Singhal et al., 2023). We

ask whether fusing many complementary reference-based signals can close the alignment loop without external supervision while resisting single-metric gaming.

We introduce PolyMetric Alignment (RLMA), a reference-based reward framework that addresses this problem by fusing multiple heterogeneous evaluators into a single scalar reward. The resulting reward can be consumed by best-of- K selection at inference and DPO fine-tuning of the generator. In our implementation, RLMA combines 22 off-the-shelf signals spanning surface-form metrics, embedding similarities, language-model log-likelihoods, and learned reward models. The central intuition is that these signals provide complementary views of generation quality. Exploiting any one metric is then penalised by the others. The fused reward therefore provides a more robust objective than any individual evaluator.

The key challenge is to reliably aggregate a large and heterogeneous bank of evaluation metrics. A natural solution is a Set Transformer (Lee et al., 2019), which is permutation-invariant and therefore well suited to metric sets. However, we find that standard STs perform best with relatively small metric banks and degrade as additional signals are introduced. This reveals a central limitation. Simply increasing the number of evaluators can overwhelm the aggregator rather than improve it.

We therefore develop the **Residual Set Transformer (RST)**, a scalable, query-aware architecture designed specifically for reward aggregation. RST modifies the Induced Set Attention Block of ST in four targeted ways. First, pre-LayerNorm residual connections preserve direct identity pathways, preventing informative signals from being diluted by repeated cross-signal mixing. Second, a cached query-context token conditions aggregation on the input query, allowing the same metric to be weighted differently depending on the task context. Third, attention-based readout replaces mean pool-

Paradigm	Reward source	Applicability
RLHF	human preference labels	tasks a human can evaluate
RLAIF	LLM-as-judge AI feedback	tasks an LLM judge can evaluate
RLVR	executable verifier $V(x, \tau)$	tasks with programmatic correctness checks
RLA (Lins and Xu, 2025)	single similarity metric against reference $R(q)$	tasks with reference-based supervision
RLMA (ours)	N metrics fused against reference $R(q)$ via RST	tasks with reference-based supervision

Table 1: Position of RLMA as a *reward-source* framework. Each row describes where the per-trajectory reward signal originates. RLA is the immediate single-metric predecessor, and RLMA generalises it to N fused metrics.

ing, enabling adaptive weighting of heterogeneous signals for each example. Finally, depth and the number of inducing points scale with the metric-bank size, providing additional capacity as more evaluators are added.

Empirically, RLMA is near-optimal, consistently stronger, broadly generalizable, and lightweight. **Near-optimal.** It reaches 51.9% vs. 52.0% oracle Pass@1 on the four in-domain pools and 74.57% vs. 76.10% on the nine-pool union, as reported in Tables 2 and 3. **Stronger.** On the nine-pool union it outperforms RLHF via ArmoRM by +7.4 pp, RLAIF via gpt-oss-120b by +10.3 pp, RLVR via ReasonFlux (Zou et al., 2025) by +15.4 pp, and single-metric RLA by +17.6 pp. It further surpasses reference-aware ArmoRM on TQA, GSM8K, and e-CARE, as reported in Tables 3 and 7. **Generalizable.** RST is the strongest of ten combiner gates and gains +6.05 pp over vanilla ST as N grows from 12 to 22, transfers zero-shot to GSM8K at 97.2% of oracle and across three unseen generators, and improves TriviaQA Pass@1 by +4.80 pp via DPO and XSum ROUGE-L (Lin, 2004) by +4.57 pp. See Tables 4, 5, and 6, and App. D and F.1. **Lightweight.** The 113K-parameter gate is $\sim 70,000\times$ smaller and $\sim 300\times$ cheaper at inference than an 8B RM, while RST-14 reduces the end-to-end cost per candidate by roughly $2\times$. See App. E.2.

To summarize, our contributions are:

1. **PolyMetric Alignment (RLMA).** We introduce a reference-based reward framework that fuses N off-the-shelf evaluators through within-query Bradley–Terry modeling, eliminating the need for new human preference labels, an external LLM judge, or a task-specific verifier.
2. **Residual Set Transformer (RST).** We propose an aggregator that scales with the metric bank rather than plateauing, and conditions per-metric weight on the query rather than being permutation-only.
3. **Comprehensive empirical validation.** We benchmark RLMA across ten gating architectures, nine query pools, three unseen test-time

candidate generators, one downstream generator fine-tuning setting via DPO, and one fuzzy-reference summarization task, against RLHF, RLAIF, RLVR, and reference-aware ArmoRM baselines, see App. A.

2 Related work

Prior reward models. Standard reward models train one scalar head with Bradley–Terry loss (Ouyang et al., 2022). The closest multi-objective work, ArmoRM (Wang et al., 2024), co-trains 19 homogeneous regression heads on UltraFeedback labels with a prompt-gated MLP. RLMA instead fuses N heterogeneous, independently pretrained off-the-shelf scorers spanning surface, embedding, LM log-likelihood, and reward-model families without retraining any of them, via a permutation-invariant set encoder supervised only on within-query best-of- K Bradley–Terry pairs.

Existing signal-combination methods. We benchmark RST against six aggregators from three families, namely top- k routers Switch (Fedus et al., 2022), Mixtral (Jiang et al., 2024), and DeepSeek-MoE (Dai et al., 2024), feature-wise modulators FiLM (Perez et al., 2018) and Q-Former (Li et al., 2023), and the permutation-invariant Set Transformer ST (Lee et al., 2019). RST extends ST with LayerNorm-residual pathways, a cached query-context token, attention-pooled readout, and bank-size-scaled capacity, and §4.2 shows it is the strongest of the ten tested gates and the only one that keeps improving as the bank grows.

Reference matching versus executable verification. RLVR uses an executable verifier $V(x, \tau)$ that judges correctness from the query and candidate alone, for example a math checker, unit tests, a theorem prover, or a game simulator, and needs no reference. RLMA instead scores candidates against a stored reference set $R(q)$ via a similarity or membership function $g(\tau, R(q))$ such as alias match, ROUGE, BERTScore (Zhang et al., 2020), or embedding cosine. The two are complementary and both eliminate human labels, as summarised in

Table 1.

3 Method

3.1 Notation and component bank

Let $\mathcal{Q}$ denote the query set and $q \in \mathcal{Q}$ a query. Let $\mathcal{T}^*$ denote the space of finite token sequences over the model vocabulary. For each q , the generator produces $K \in \mathbb{Z}_{>0}$ candidate trajectories $\{\tau_k(q)\}_{k=1}^K$ with $K \in \{4, 8\}$, where each $\tau_k(q) \in \mathcal{T}^*$. The reference set $R(q) \subset \mathcal{T}^*$ contains the reference answer to q together with any acceptable paraphrases.

We define $N \in \mathbb{Z}_{>0}$ off-the-shelf scoring functions $\phi_i: \mathcal{Q} \times \mathcal{T}^* \rightarrow \mathbb{R}$ indexed by $i \in \{1, \dots, N\}$, where each ϕ_i maps a query q and a candidate trajectory $\tau \in \mathcal{T}^*$ to a scalar quality score. The ϕ_i are fixed and drawn from four families, namely surface metrics such as BLEU (Papineni et al., 2002), embedding similarities such as BERTScore, language-model log-likelihoods, and pretrained reward models. We use $N \in \{12, 22\}$ depending on the bank.

Score. For each query q , candidate index $k \in \{1, \dots, K\}$, and metric index $i \in \{1, \dots, N\}$, the raw component score $c_{i,k}(q) \in \mathbb{R}$ is the value of the i -th metric ϕ_i applied to the k -th candidate $\tau_k(q)$ for query q :

$$c_{i,k}(q) = \phi_i(q, \tau_k(q)). \quad (1)$$

Normalise. The gate input is obtained by z-normalising each raw component over the union of training-pool (q, k) pairs:

$$\bar{c}_{i,k}(q) = \frac{c_{i,k}(q) - \mu_i}{\sigma_i + \epsilon}, \quad (2)$$

where $\mu_i, \sigma_i \in \mathbb{R}$ are the training set mean and standard deviation of ϕ_i and $\epsilon \in \mathbb{R}_{>0}$ with $\epsilon=10^{-6}$. The raw score $c_{i,k}(q)$ is from Eq. 1. Stacking the N normalised entries defines the per-candidate component vector

$$\mathbf{c}_k(q) = [\bar{c}_{1,k}(q), \dots, \bar{c}_{N,k}(q)]^\top \in \mathbb{R}^N, \quad (3)$$

which is the input to the gate.

3.2 RST Gate

The gate. The gate is a function $f_\theta: \mathbb{R}^N \rightarrow \mathbb{R}$ that maps the normalised component vector $\mathbf{c}_k(q)$ from Eq. 2 to a scalar score $s_k(q; \theta) \in \mathbb{R}$, used as the best-of- K selection score and as the reward for DPO fine-tuning. Because the N components have

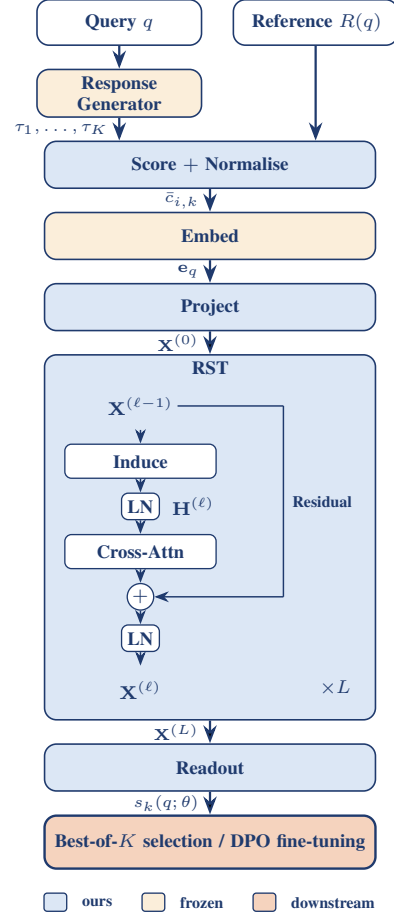


Figure 1: RLMA architecture. Box labels match the algorithm stages Score, Normalise, Embed, Project, Scale, Induce, Residual, and Readout as defined in §3. Sky boxes are this paper’s contributions. Cream boxes are frozen off-the-shelf modules such as the Llama-3 generator and MiniLM-L6 encoder. The coral box is the downstream consumer that uses the score s_k for Best-of- K selection at inference and DPO fine-tuning of the generator, discussed in §4.2.

no intrinsic order, f_θ must be permutation-invariant. The natural starting point is the Set Transformer (ST) of Lee et al. (2019), whose Induced Set Attention Block (ISAB) is permutation-invariant by construction. Our concrete realisation of $s_k(q; \theta)$ is Eq. 8 below.

Vanilla ISAB is permutation-invariant but lacks query awareness and scales poorly as the metric bank grows. RST introduces four architectural refinements that convert it into a bank-scalable, query-aware aggregator whose accuracy scales with N over the tested range rather than plateauing.

Embed. Embed and Project introduce a cached query-context token, enabling query-conditioned aggregation so that the same metric can be weighted differently for different queries. We give the gate access to the query by computing a sentence embedding of q once and caching it. Con-

cretely, the frozen MiniLM-L6-v2 sentence transformer (Wang et al., 2020) implements an encoder $\text{Enc}: \mathcal{Q} \rightarrow \mathbb{R}^{384}$ that maps the query text q to a dense 384-dimensional embedding. We denote this query embedding by $\mathbf{e}_q \in \mathbb{R}^{384}$, defined as $\mathbf{e}_q = \text{Enc}(q)$. The vector $\mathbf{e}_q$ then feeds the Project step as the query token. Since Enc is not updated during training and is reused across all K candidates of the same query, it adds no per-candidate cost at inference.

Project. The cached query embedding $\mathbf{e}_q \in \mathbb{R}^{384}$ from the Embed step is projected into a query token and prepended to the N component tokens. Let $d \in \mathbb{Z}_{>0}$ denote the RST hidden dimension. We introduce three sets of learnable parameters. The matrix $W_q \in \mathbb{R}^{d \times 384}$ projects the 384-dimensional query embedding $\mathbf{e}_q$ into the RST hidden dimension. The matrix $W_c \in \mathbb{R}^{d \times 1}$ projects each scalar component score $\bar{c}_{i,k}(q)$ into the RST hidden dimension. The vector $\mathbf{p}_i \in \mathbb{R}^d$ is a learnable position or type embedding for the i -th component, giving each metric a distinct token identity. For a fixed query $q \in \mathcal{Q}$ and candidate index $k \in \{1, \dots, K\}$ from Section 3.1, the token stack $\mathbf{X}^{(0)} \in \mathbb{R}^{(N+1) \times d}$ has $N+1$ rows. Row 0 is the query token, and rows $1 \dots N$ are the component tokens, given by

$$\begin{aligned} \mathbf{X}_0^{(0)} &= W_q \mathbf{e}_q, \\ \mathbf{X}_i^{(0)} &= W_c \bar{c}_{i,k}(q) + \mathbf{p}_i, \end{aligned} \quad (4)$$

for $i \in \{1, \dots, N\}$, with $\bar{c}_{i,k}(q)$ from Eq. 2. All tensors that follow, such as $\mathbf{X}^{(\ell)}$ and $\mathbf{H}^{(\ell)}$, inherit an implicit (q, k) argument that we suppress for compactness.

Scale. Scale sets the block depth and inducing-point count as functions of N , providing enough capacity for accuracy to keep improving as the metric bank grows rather than plateauing. Before the ISAB stack we set the block depth L and inducing point count M as functions of the bank size N from Section 3.1:

$$\begin{aligned} L(N) &= \max(1, \lfloor N/10 \rfloor), \\ M(N) &= \max(N, 8), \end{aligned} \quad (5)$$

both in $\mathbb{Z}_{>0}$. $L(N)$ sets the block-index limit and $M(N)$ the row count of $\mathbf{I}^{(\ell)}$. This replaces the fixed defaults $L=1$, $M=8$ of vanilla ST and lets a gate that scales from more components keep its capacity matched.

Induce. Induce and Residual add identity-preserving pathways, preventing informative single-metric signals from being diluted by repeated cross-signal mixing. Let $\mathbf{I}^{(\ell)} \in \mathbb{R}^{M \times d}$ denote a set of M learnable *inducing points* at block $\ell \in \{1, \dots, L\}$, where the inducing-point count M is set by the Scale step in Eq. 5 and ℓ is the block index. Inducing points are the central construction of the Set Transformer’s ISAB block (Lee et al., 2019). They act as intermediate query slots rather than data points, turning full $O(N^2)$ self-attention over the token set into two cheaper $O(NM)$ steps through an M -wide information bottleneck. The induction step compresses the $(N+1)$ token set onto M summary tokens $\mathbf{H}^{(\ell)} \in \mathbb{R}^{M \times d}$ by letting the inducing points attend over the current set:

$$\mathbf{H}^{(\ell)} = \text{LN}(\text{Attn}(\mathbf{I}^{(\ell)}, \mathbf{X}^{(\ell-1)}, \mathbf{X}^{(\ell-1)})). \quad (6)$$

The starting tensor $\mathbf{X}^{(0)} \in \mathbb{R}^{(N+1) \times d}$ that enters the first ISAB block, $\ell=1$, is exactly the token stack assembled by the Project step, given row-by-row in Eq. 4. Subsequent blocks update $\mathbf{X}^{(\ell)}$ from $\mathbf{X}^{(\ell-1)}$ through Induce and Residual below.

Residual. When one metric already provides the correct signal, the gate should preserve that signal rather than average it away. Therefore, we add a residual connection to capture the original per-token signals. The residual step reads back from the M summary tokens to the $(N+1)$ per-token states and *adds* the input back before LayerNorm (Bach et al., 2016), giving every component a direct identity route to the readout:

$$\mathbf{X}^{(\ell)} = \text{LN}(\text{Attn}(\mathbf{X}^{(\ell-1)}, \mathbf{H}^{(\ell)}, \mathbf{H}^{(\ell)}) + \mathbf{X}^{(\ell-1)}). \quad (7)$$

A residual add requires the layer output and its input to have identical shape. The Induce step produces $M \times d$ summary tokens $\mathbf{H}^{(\ell)}$, whose shape does not match the input $\mathbf{X}^{(\ell-1)}$ of shape $(N+1) \times d$, so no residual add is possible immediately after Induce. The readback attention of Eq. 7 restores the shape to $(N+1) \times d$, so the residual add on the right-hand side is well-defined only at this second attention. Vanilla ST has neither this residual add nor the surrounding LayerNorm.

Readout. Readout replaces mean pooling with attention pooling, allowing adaptive per-example weighting instead of uniform averaging of the summary tokens. Let $\mathbf{q}_{\text{pool}} \in \mathbb{R}^d$ be a learnable, input-independent aggregation template that plays the

role of a BERT-style [CLS] token, pooling the M summary tokens $\mathbf{X}^{(L)}$ into a single d -dimensional vector via cross-attention rather than uniformly averaging them. Similarly, let $\mathbf{w} \in \mathbb{R}^d$ be a learnable readout vector that maps the pooled vector to the scalar gate output. Both $\mathbf{q}_{\text{pool}}$ and $\mathbf{w}$ are free parameters trained end-to-end by backpropagation. The gate output is

$$s_k(q; \theta) = \mathbf{w}^\top \text{Attn}(\mathbf{q}_{\text{pool}}, \mathbf{X}^{(L)}, \mathbf{X}^{(L)}), \quad (8)$$

with θ collecting the learnable parameters $\{W_c, W_q, \mathbf{p}_i, \mathbf{I}^{(\ell)}, \mathbf{q}_{\text{pool}}, \mathbf{w}\}$ together with the attention and LayerNorm parameters.

3.3 Training and inference

The full pipeline is summarised in Algorithm 1.

Train. Let $g(\tau) = 1[\tau \text{ matches } R(q)] \in \{0, 1\}$ be the reference-match indicator, pre-computed per-pool, using substring or alias matching for QA, numeric matching for math, and label matching for hallucination detection. Let $\text{sigmoid}: \mathbb{R} \rightarrow (0, 1)$ denote the logistic sigmoid. Within each query, positive and negative pairs are formed by pairing each reference-matching candidate ($g=1$) with each non-matching one ($g=0$). Ties, where all candidates in a query share the same g value, yield no pair and are excluded from training. Formally, $\mathcal{D} = \{(q, k_+, k_-) : q \in \mathcal{Q}_{\text{train}}, g(\tau_{k_+}(q)) = 1, g(\tau_{k_-}(q)) = 0\}$ is the set of within-query positive and negative pairs derived from the reference set $R(q)$. The training loss is the within-query pairwise Bradley–Terry objective

$$\mathcal{L}_{\text{BT}}(\theta) = -\sum_{\mathcal{D}} \log \text{sigmoid}(s_{k_+}(q; \theta) - s_{k_-}(q; \theta)), \quad (9)$$

with $s_k(q; \theta) \in \mathbb{R}$ from Eq. 8. After training, the trained gate parameters are

$$\theta^* = \arg \min_{\theta} \mathcal{L}_{\text{BT}}(\theta), \quad (10)$$

with $\mathcal{L}_{\text{BT}}$ from Eq. 9.

Best-of- K selection. At inference, the gate selects the best of K candidates as

$$k^*(q) = \arg \max_{k \in \{1, \dots, K\}} s_k(q; \theta^*). \quad (11)$$

DPO fine-tuning. The same trained score $s_k(q; \theta^*)$ can be used as a scalar reward to construct preference pairs for DPO fine-tuning (Rafailov et al., 2023) of a generator policy π_ϕ against a

Algorithm 1: RLMA best-of- K pipeline

Input: $\mathcal{Q}_{\text{train}}$ with refs $R(q)$, $\{\tau_k\}_{k=1}^K$ per q , bank $\{\phi_i\}_{i=1}^N$

Output: Best-of- K index $k^*(q)$

Training (offline)

for each $q \in \mathcal{Q}_{\text{train}}$ **do**
 Score $q, \tau_k \mapsto c_{i,k}$ [Eq. 1]
 Normalise $c_{i,k} \mapsto \bar{c}_{i,k}$ [Eq. 2]
 Embed $q \mapsto \mathbf{e}_q$
 Project $\bar{c}_{i,k}, \mathbf{e}_q \mapsto \mathbf{X}^{(0)}$ [Eq. 4]
 Scale $N \mapsto L(N), M(N)$ [Eq. 5]
for each epoch do for each minibatch (q, k_+, k_-) **do**
 for $\ell = 1, \dots, L$ **do**
 Induce $\mathbf{X}^{(\ell-1)} \mapsto \mathbf{H}^{(\ell)}$ [Eq. 6]
 Residual $\mathbf{X}^{(\ell-1)}, \mathbf{H}^{(\ell)} \mapsto \mathbf{X}^{(\ell)}$ [Eq. 7]
 Readout $\mathbf{X}^{(L)} \mapsto s_k$ [Eq. 8]
 Train $\mathcal{D} \mapsto \theta^*$ [Eqs. 9,10]

Inference at test query q , using θ^* :

run *Score, Normalise, Embed, Project, Induce, Residual, Readout*

Select $\{s_k\}_{k=1}^K \mapsto k^*(q)$ [Eq. 11]

frozen reference policy π_{ref} . For each query q we form pairs (k_+, k_-) with $s_{k_+}(q; \theta^*) > s_{k_-}(q; \theta^*)$ and minimise

$$\mathcal{L}_{\text{DPO}}(\phi) = -\mathbb{E}_{(q, k_+, k_-)} [\log \text{sigmoid}(\beta \Delta_\phi)], \quad (12)$$

where $\beta > 0$ is the DPO temperature, and, writing $\tau_+ \equiv \tau_{k_+}(q)$ and $\tau_- \equiv \tau_{k_-}(q)$,

$$\Delta_\phi = \log \frac{\pi_\phi(\tau_+ | q)}{\pi_{\text{ref}}(\tau_+ | q)} - \log \frac{\pi_\phi(\tau_- | q)}{\pi_{\text{ref}}(\tau_- | q)} \quad (13)$$

is the reference-normalised log-likelihood margin.

4 Experiments

4.1 Training

Datasets. We train on four adversarial best-of- K QA pools, namely TriviaQA in-domain, TriviaQA seedpool, HaluEval, and HaluBench, each with 200 queries $\times K=4$ Llama-3-8B-Instruct candidates at $T=1.0$ and top- $p=0.9$, totalling 3,200 trajectories. We evaluate under three conditions. First, 800 held-out in-domain queries. Second, the nine-pool union with 4019 queries and gates retrained. Third, zero-shot GSM8K with $K=8$ over 200 problems and oracle 0.895. Per-pool statistics are listed in App. C.

Training recipe. We train each gate with the within-query pairwise Bradley-Terry loss of Eq. 9 using AdamW at learning rate 10^{-3} , batch size 256, 30 epochs, and seed 42. All gates are retrained under this recipe whenever the component bank or

training pool changes. Architecture, encoder, and hardware details are listed in App. B.1.

Component bank. We sweep bank size $N \in \{1, \dots, 22\}$, as shown in Figure 4. The $N=22$ bank has four families of off-the-shelf scorers: surface metrics, embedding similarities, LM cross-entropy under a frozen Llama-3, and learned reward models. Each scorer is evaluated once per query-candidate pair (q, τ_k) and cached, and no scorer is fine-tuned. Full per-family list and per-component ablation appear in App. D.

Baselines. Four classes, all reported in Tables 2 and 3. **Bounds:** Random-1 and Pass@K oracle. **Single-metric selectors:** cosine, BARTScore (Yuan et al., 2021), and ROUGE-L used alone. **RL reward-model paradigms:** RLHF via ArmoRM, RLAIF via the gpt-oss-120b judge, RLVR via ReasonFlux PRM, and the reference-driven RLA baseline. **Learned aggregators:** ten gating architectures, namely mean pooling, light and heavy MLP, three MoE variants Switch, Mixtral, and DeepSeek-MoE, FiLM, Q-Former, vanilla ST, and our RST. Details and parameter counts in App. B.1, Table 11.

4.2 Evaluation results

In-domain best-of- K on the four pools. Table 2 reports best-of- $K=4$ Pass@1 on the four in-domain pools. RST tops every gate at 0.5187, reaching 99.8% of oracle and tying it on HaluEval at 0.410 Pass@1. Set-attention as a class dominates: RST and vanilla ST take the top two slots at 0.5187 and 0.5175, while the best non-set gates, ArmoRM-style MLP at 0.5138 and Switch top-1 at 0.5112, trail. Trivial aggregators such as mean at 0.4750 and light softmax at 0.4900 even lose to ROUGE-L at 0.4975.

Nine-pool best-of- K on 4019 queries. Table 3 reports best-of- $K=4$ Pass@1 on the 4019-query union of nine pools. RST is the strongest gate at **0.7457**, reaching 98.0% of the 0.7606 oracle and +17.6 pp over RLA 8B. It is also the only gate that scales with bank size. From $N=12$ to $N=22$, vanilla ST falls from 0.6977 to 0.6852 and FiLM falls from 0.7236 to 0.7049, while RST rises from 0.7437 to 0.7457. Its +6.05 pp lead over ST at $N=22$ therefore isolates the architectural gain from LayerNorm + residual, the ctx-token, and attention pooling, independent of bank composition.

Method	Overall	TQA	HaluE.	HaluB.
Random 1	0.3837	0.625	0.265	0.020
Oracle Pass@K	0.5200	0.820	0.410	0.030
<i>Baselines from established RL paradigms</i>				
RLHF (ArmoRM)	0.4725	0.770	0.330	0.020
RLVR (ReasonFlux)	0.3837	0.625	0.265	0.020
RLAIF (gpt-oss-120b)	0.4650	0.760	0.315	0.025
RLA	0.4038	0.670	0.255	0.020
<i>Off-the-shelf single metric (ours, no learning)</i>				
cosine	0.4913	0.800	0.340	0.025
bartscore	0.4975	0.795	0.375	0.025
rouge_l	0.4975	0.780	0.400	0.030
<i>RLMA gates (ours)</i>				
mean	0.4750	0.755	0.365	0.025
light	0.4900	0.780	0.375	0.025
medium	0.5138	0.815	0.395	0.030
heavy	0.5100	0.810	0.395	0.025
switch	0.5112	0.810	0.395	0.030
mixtral	0.5025	0.800	0.385	0.025
deepseek	0.5088	0.805	0.400	0.025
ST	0.5175	0.815	0.410	0.030
film	0.5112	0.810	0.400	0.025
q-former	0.5062	0.805	0.390	0.025
<i>RLMA gate RST</i>				
RST ($N=22$)	0.5187	0.820	0.410	0.030

Table 2: Best-of- $K=4$ Pass@1 on the 4 in-domain pools with 800q. Baseline gates run at $N=14$, and RST runs at its scalable $N=22$, $L=2$ configuration. Bold marks the best value per column. TQA denotes TriviaQA in-domain, HaluE. denotes HaluEval, and HaluB. denotes HaluBench. Gate codes are defined in §4.1.

Cross-task zero-shot transfer to GSM8K. Table 4 reports zero-shot transfer to GSM8K of gates trained only on the four in-domain pools. On GSM8K, RST reaches 0.870, i.e. **97.2%** of oracle and +4.5 pp over FiLM. Vanilla ST trails at 0.830, or 93% of oracle. RLHF and RLVR collapse to random at 0.7050 on arithmetic, out-of-distribution for their TriviaQA-trained heads. On the combined four-pool + GSM8K set, RST reaches 0.5870, or 98.7% of the 0.5950 ceiling.

Cross-generator transfer. To probe generator robustness, we score fresh $K=4$ rollouts on 500 TriviaQA questions from three test-time generators drawn from three distinct regimes, as reported in Table 5. To probe zero-shot transfer we use a lightweight three-metric fusion of cosine similarity, ROUGE-L, and BLEU against the TQA alias set rather than the full 22-metric RST gate. A full-gate cross-generator retrain is left for future work.

Policy optimization via DPO fine-tuning on the RLMA reward. We DPO fine-tune Qwen-2.5-7B-Instruct via a LoRA adapter with $r=16$ and $\alpha=32$ on Bradley-Terry pairs ordered by the RST-fused scalar over the 22-metric bank. At inference

Method (4019q)	Pass@1	% oracle
Random 1	0.5922	77.9
Oracle (Pass@K)	0.7606	100.0
<i>Baselines from established RL paradigms</i>		
RLHF (ArmoRM)	0.6718	88.3
RLVR (ReasonFlux)	0.5922	77.9
RLAIF (gpt-oss-120b)	0.6427	84.5
RLA (Lins and Xu, 2025)	0.5713	75.1
<i>Off-the-shelf single metric (ours, no learning)</i>		
rouge_l	0.6872	90.4
bartscore	0.6862	90.2
<i>RLMA gates (ours)</i>		
lean bank, $N=11$: ST	0.6902	90.7
lean+RWM, $N=12$: ST	0.6977	91.7
full bank ($N=22$): ST	0.6852	90.1
full bank ($N=22$): mean	0.6897	90.7
full bank ($N=22$): deepseek	0.6892	90.6
full bank ($N=22$): q-former	0.6880	90.4
full bank ($N=22$): film	0.7049	92.7
lean+RWM ($N=12$): film	0.7236	95.1
<i>RLMA gate RST</i>		
RST ($N=12, L=5$)	0.7437	97.8
RST ($N=22, L=2$)	0.7457	98.0

Table 3: Best-of- $K=4$ Pass@1 on the 9-pool 4019-query union, varying gate architecture $\times$ bank size N . Bold marks the best value per column. Paradigm proxies use ArmoRM trained on UltraFeedback for RLHF, ReasonFlux PRM for RLVR, and the gpt-oss-120b judge for RLAIF.

the RST scorer is dropped, and *fresh* responses sampled at $T=0.9$, $K=8$ are scored with the strict last-100-character alias parser.

Reference-aware reward-model baseline. Standard RLHF, RLAIF, and RLVR reward models do not access the reference at scoring time, so we compare RLMA against a reference-conditioned baseline. Specifically, we evaluate RLHFlow ArmoRM-Llama3-8B-v0.1 with the gold answer prepended to the scoring prompt, as reported in Table 7. Parameter count, prompt template, candidate pool, and correctness parser match the paper’s headline setup, with $n=500$ queries per pool.

Ablations. We defer the full ablation battery to App. D, covering RST refinement decomposition in Table 13, per-component LOO in Table 14, and component-bank saturation in Figure 4.

4.3 Efficiency

The RST gate runs at 0.33 ms per candidate on a single CPU thread, $\approx 300\times$ faster than a ~ 100 ms 8B RM forward on GPU. End-to-end, the RST-22 pipeline costs ~ 45 ms per candidate, roughly $2\times$ cheaper than a monolithic 8B RM at matched accuracy. RST-14 in App. E.2 cuts this further to 64% of RST-22 while retaining 95% of the LOO-attributable signal. See App. E.1 and Table 9.

Method	Overall	GSM8K
Random 1	0.4480	0.7050
Oracle (Pass@K)	0.5950	0.8950
<i>Baselines from established RL paradigms</i>		
RLHF (ArmoRM)	0.5410	0.8150
RLVR (ReasonFlux)	0.4480	0.7050
RLAIF (gpt-oss-120b)	0.5200	0.7400
<i>Off-the-shelf single metric (ours, no learning)</i>		
rouge_l	0.5580	0.8000
bartscore	0.5530	0.8150
<i>RLMA gates (ours, trained on the four pools only)</i>		
medium	0.5650	0.8150
deepseek	0.5700	0.8150
ST	0.5740	0.8300
film	0.5760	0.8250
<i>RLMA gate RST</i>		
RST ($N=12, L=5$)	0.5870	0.8700

Table 4: Zero-shot transfer to GSM8K at $K=8$, 200 problems, oracle Pass@K = 0.895. Gates trained only on the four in-domain pools.

Test-time generator	Rand-1	BoK	Oracle	Δ
Qwen-2.5-14B-Instruct	27.4%	28.4%	49.0%	+1.0
Mistral-7B-Instruct-v0.3	18.8%	23.8%	42.6%	+5.0
Qwen-2.5-7B-Inst.	21.6%	25.6%	40.2%	+4.0

Table 5: Cross-generator zero-shot transfer on 500 TriviaQA questions, $K=4$ per query, strict last-100-character alias match. Qwen-2.5-7B is decoded at $T=0.3$. Absolute Pass@1 is lower than the paper’s headline in-generator numbers because this probe uses a three-metric fusion of cosine, ROUGE-L, and BLEU rather than the full 22-metric RST gate.

Configuration (TriviaQA, strict Pass@1)	Pass@1
Qwen-2.5-7B-Instruct (base)	62.00%
Qwen-2.5-7B-Instruct + DPO on RLMA reward	66.80%
Improvement	+4.80 pp

Table 6: DPO fine-tuning on the RLMA reward. Pass@1 on *fresh post-training generation*.

Pool	Rand-1	ArmoRM _{ref}	Oracle	Δ
TriviaQA	46.60%	65.60%	75.40%	+19.00 pp
GSM8K	68.80%	86.00%	87.40%	+17.20 pp
e-CARE	73.40%	78.00%	91.80%	+4.60 pp

Table 7: ArmoRM in a reference-aware configuration with $n=500$ per pool and gold answer prepended to the reward-scoring prompt. Rand-1 is a random-1 selection baseline. ArmoRM_{ref} is the Best-of- K selection by reference-aware ArmoRM. Δ is the Best-of- K gain over Rand-1.

5 Conclusion

PolyMetric Alignment fuses N off-the-shelf evaluators against the dataset’s reference answer through a lightweight RST gate. On nine reference-available pools, RLMA reaches near-oracle Pass@1 and improves over RLA by 30% relative at a fraction of an 8B reward model’s compute, with no human preference labels, no LLM judge, and no task-specific verifier.

6 Limitations

RLMA consumes a reference set $R(q)$ and inherits the source dataset’s reference quality, so it does not apply to open-ended tasks such as dialogue safety, subjective preference, or creative writing, and extension to LLM-judge or preference-aggregated proxy references is future work. It also cannot recover when the candidate pool lacks a correct trajectory, as on HaluBench with oracle Pass@ $K=3\%$. All experiments score whole responses at $K=4$, leaving step-level gating and larger K unexplored. Finally, the 22-metric bank contains redundant 8B heads ArmoRM (Wang et al., 2024) and Skywork (Liu et al., 2024), which App. E.2 prunes into a smaller RST-14 deployment variant.

References

- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N. Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2025. How to evaluate reward models for RLHF. In *International Conference on Learning Representations (ICLR)*.
- Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2025. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797.
- Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIIF vs. RLHF: Scaling reinforcement learning from human feedback with AI feedback. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 26874–26901. PMLR.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosioerek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR.

- Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- João Luís Lins and Jia Xu. 2025. Reinforcement learning with supervised alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.
- Chris Yuhao Liu, Liang Zeng, Jiakai Liu, Rui Yan, Ju-jie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in LLMs. *arXiv preprint arXiv:2410.18451*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 3942–3951.
- Maja Popović. 2015. chrF: Character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395. Association for Computational Linguistics.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*.
- Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892. Association for Computational Linguistics.
- Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. 2023. A long way to go: Investigating length correlations in RLHF. *arXiv preprint arXiv:2310.03716*.
- Matthew Snover, Bonnie Dorri, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.
- Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 5776–5788.
- Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. BARTScore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2025. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations (ICLR)*.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR)*.
- Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578. Association for Computational Linguistics.
- Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. 2025. Reasonflux-PRM: Trajectory-aware PRMs for long chain-of-thought reasoning in LLMs. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*.

Appendix

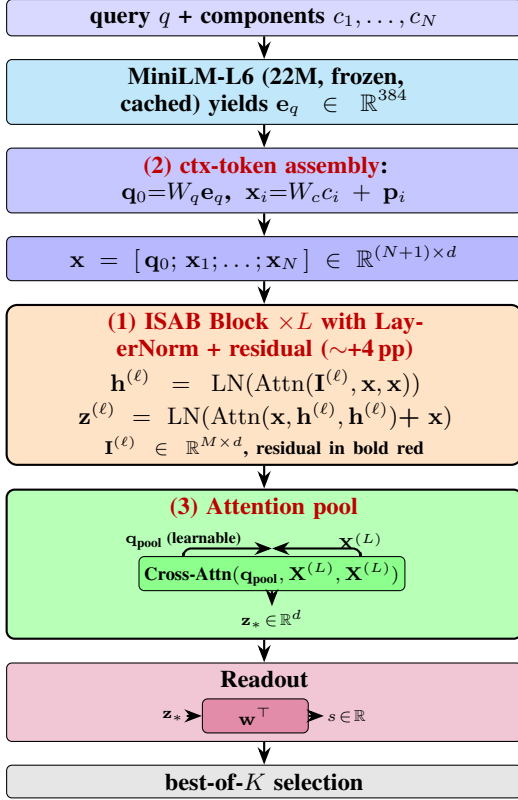


Figure 2: Equation-style reproduction of Figure 1 for reference; red callouts mark the three refinements over vanilla ST (Lee et al., 2019).

A Experimental superiority overview

Table 8 maps every empirical result claimed in the abstract and introduction to its purpose, the story arc it supports, and the section or table where the full number lives. Arcs are labelled *It Works* for near-oracle ceiling saturation, *It Wins* for outperforming every established baseline, *Why It Wins* for the architectural mechanism, *It Generalizes* for cross-domain, cross-generator, downstream fine-tuning, and cross-task-type behaviour, and *It Ships* for deployment efficiency.

B Method details

B.1 Implementation details

Standalone full-size figures. Figures 2 and 3 reproduce the main-body pipeline and the ISAB-block residual schematic at full size for reference; the main-body Figure 1 integrates the two.

Gate architecture. The RST of Section 3.2 is a permutation-invariant set encoder over the N component scores plus a cached MiniLM-L6 query embedding. Block depth follows $L(N) = \max(1, \lfloor N/10 \rfloor)$ and inducing-point count $M(N) = \max(N, 8)$ per Eq. 5. The gate has about 113K trainable parameters and runs in sub-

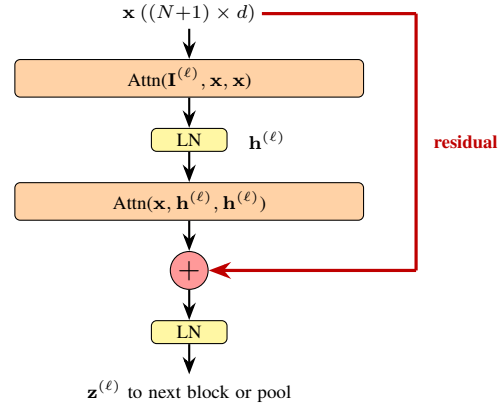


Figure 3: ISAB block detail: zoom of the RST box in Figure 2. The red residual skip is the single largest contributor to RST’s gain over vanilla ST, as reported in Section 4.2.

millisecond CPU inference once the query embedding is cached, as reported in Section 4.3.

Sentence encoder Enc. The frozen sentence encoder is realised by all MiniLM-L6 v2 (Wang et al., 2020) with 22M parameters and $d_e=384$. We run it once per query and store the embedding alongside the component features so it incurs no per-candidate cost at inference, as reported in Section 4.3.

Hardware. We run all experiments on a single NVIDIA RTX 5090 GPU. Component scoring takes ~ 3 min on CPU for surface metrics and cached features, plus ~ 2 min on GPU for CE, BERTScore, and BARTScore. All ten gates train in ~ 5 min of CPU time, giving an end-to-end pipeline of ~ 10 min.

Inducing shape verification of RST. For the headline runs we use $M=N$ inducing points and depth $L=2$ for $N=22$, $L=5$ for $N=12$. Inspecting the trained checkpoints confirms the inducing matrices in each ISAB block have shape $M \times d$ as expected, with $\text{inducing1.shape}=\text{inducing2.shape}=22 \times 64$ for the RST $N=22$ run at $\text{Pass@1} = 0.7457$, and $\text{inducing.shape}=12 \times 64$ for each of the 5 ISAB blocks of the RST $N=12, L=5$ run at $\text{Pass@1} = 0.7437$. Both runs use $M=N$. The LOO ablation in the main text, Table 13 and the L and M sweeps, shows the gate is insensitive to M within ± 0.5 pp, so the $+0.65$ pp gain from $N=12$ up to $N=22$ is attributable to the additional 10 heterogeneous components in the bank, not to the larger M .

Deployment cost, full breakdown. Table 9 expands the per-row summary cited from Section 4.3.

#	Arc	Category	Result	Purpose	Location
1	<i>It Works</i>	Ceiling saturation	51.9% Pass@1 vs 52.0% oracle on four in-domain pools (99.8%)	Near-optimal on training domain	Intro; Table 2
2	<i>It Works</i>	Ceiling saturation	74.57% Pass@1 vs 76.10% oracle on nine-pool union (98.0%)	Same story on cross-benchmark union	Abstract; Intro; Table 3
3	<i>It Wins</i>	Predecessor comparison	+17.6 pp over the 8B RLA single-metric baseline	Multi-metric fusion strictly outperforms single-metric	Abstract; Intro; Table 3
4	<i>It Wins</i>	RL paradigm — RLHF	+7.4 pp over ArmoRM	Beats a human-preference reward model	Abstract; Intro; Table 3
5	<i>It Wins</i>	RL paradigm — RLAI	+10.3 pp over gpt-oss-120b judge	Beats an LLM-judge feedback signal	Abstract; Intro; Table 3
6	<i>It Wins</i>	RL paradigm — RLVR	+15.4 pp over ReasonFlux PRM	Beats a process-reward verifier signal	Abstract; Intro; Table 3
7	<i>It Wins</i>	Fair reference-aware baseline	+19.0/ +17.2/ +4.6 pp over reference-aware ArmoRM on TQA/GSM8K/e-CARE	Win is not because ArmoRM lacked the reference: RLMA still wins when it has it	Intro; Table 7
8	<i>Why It Wins</i>	Architectural superiority	Strongest of ten tested gating architectures	RST outperforms MLP, FiLM, QFormer, MoE, ST, and other tested aggregators	Intro; Table 2; Table 3
9	<i>Why It Wins</i>	Bank scaling	+6.05 pp over vanilla ST as the bank grows from $N=12$ to $N=22$	RST uniquely scales with more metrics; ST plateaus and then collapses	Intro; App. D, Table 13
10	<i>It Generalizes</i>	Zero-shot cross-domain	97.2% of GSM8K oracle with no GSM8K-specific training	Zero-shot transfer to a held-out math domain	Abstract; Intro; Table 4
11	<i>It Generalizes</i>	Zero-shot cross-generator	Positive Best-of- K lift on Mistral-7B, Qwen-7B at $t=0.3$, and Qwen-14B	Gate is not tied to Llama-3-8B training rollouts	Intro; Table 5
12	<i>It Generalizes</i>	Downstream fine-tuning	+4.80 pp Pass@1 on TriviaQA via DPO on Qwen-2.5-7B-Instruct	Reward drives generator learning, not only reranking; RLMA is a reward source on equal footing with RLHF, RLAI, and RLVR	Abstract; Intro; Table 6
13	<i>It Generalizes</i>	Task-type generality	+4.57 pp ROUGE-L on XSum abstractive summarization	Extends beyond exact-match QA to fuzzy-reference regimes; the reference can be a free-text paragraph	Intro; App. F.1, Table 20
14	<i>It Ships</i>	Deployment efficiency	113K-parameter gate, $\sim 300\times$ faster than a monolithic 8B RM at gate-forward level; RST-14 pipeline $\sim 2\times$ cheaper end-to-end	Practical deployment cost; two-tier honesty distinguishes the gate-only ratio from the end-to-end pipeline ratio	Abstract; Intro, §4.3; App. B.1, Table 9; App. E.2

Table 8: Overview of all 14 empirical results announced in the abstract and introduction. Each row lists the story arc it supports, the specific claim, the intended lesson, and every location in the paper where the full number is reported.

	Vanilla ISAB	RST (ours)	8B RM
Parameters	35 K	113 K	8 B
Train time	~ 2 min	~ 3 min	44 h
Train peak GPU	< 0.5 GB	< 0.5 GB	28 GB
Inference latency (per-candidate)	0.12 ms (CPU)	0.33 ms (CPU)	≈ 100 ms (GPU)
Inference peak mem	~ 1 MB (CPU)	~ 1 MB (CPU)	16 GB (GPU)

Table 9: Deployment cost of the RST gate vs. the 8B RM it replaces. Train cost = one full BT pass on the 4019q union at $K=4$, 30 epochs, RTX 5090. Inference cost = per-candidate, CPU single-thread, batch 1, mean of 200 iters; cached embedding excluded.

B.2 Notation

Table 10 lists the symbols used in the equations and where they are first defined.

B.3 Equations summary

- Eq. 1: Raw component score $c_{i,k}(q)$ for query q and candidate τ_k .

- Eq. 2: Z-score normalised component $\bar{c}_{i,k}(q)$.
- Eq. 4: Token assembly $\mathbf{X}^{(0)}$ from $\bar{c}_{i,k}$ and $\mathbf{e}_q$.
- Eq. 6: ISAB induction step $\mathbf{H}^{(\ell)}$.
- Eq. 7: ISAB residual step $\mathbf{X}^{(\ell)}$.
- Eq. 8: Gate score $s_k(q; \theta)$ over normalised components.
- Eq. 5: Capacity scaling $L(N)$, $M(N)$.
- Eq. 9: Bradley-Terry training loss $\mathcal{L}_{\text{BT}}(\theta)$.
- Eq. 10: Trained gate parameters θ^* .
- Eq. 11: Best-of- K selection $\hat{\tau}(q)$.

B.4 RST: detailed architecture

We describe the gate of Section 3.2 in full. Let $\mathbf{c} \in \mathbb{R}^N$ be the per-pool z-normalised component vector for a candidate (q, τ) , and $\mathbf{e}_q \in \mathbb{R}^{384}$ be

Symbol	Meaning	Defined in
q	input query	§3
τ_k	k -th candidate response	Eq. 1
K	candidates per query	§3
$R(q)$	reference set	§3
N	bank size	§3
ϕ_i	i -th off-the-shelf component scorer	§3
$c_{i,k}(q)$	raw component score	Eq. 1
μ_i, σ_i	per-component train-pool stats	Eq. 2
$\bar{c}_{i,k}(q)$	z-normalised component score	Eq. 2
d	token hidden dimension	Eq. 4
$\mathbf{e}_q$	frozen query embedding	Eq. 4
W_q, W_c	query and component projections to $\mathbb{R}^d$	Eq. 4
$\mathbf{p}_i$	learned positional embedding	Eq. 4
$\mathbf{X}^{(0)}$	initial token stack of size $(N+1) \times d$	Eq. 4
ℓ	ISAB block index, $\ell \in \{1, \dots, L\}$	Eq. 6
L, M	block depth, inducing-token count	Eq. 5
$\mathbf{I}^{(\ell)}$	inducing tokens, shape $M \times d$	Eq. 6
$\mathbf{H}^{(\ell)}$	induction step output, $M \times d$	Eq. 6
$\mathbf{X}^{(\ell)}$	residual step output, $(N+1) \times d$	Eq. 7
$\mathbf{q}_{\text{pool}}$	learnable readout query, $\mathbb{R}^d$	Eq. 8
$\mathbf{z}_*$	attention-pooled summary, $\mathbb{R}^d$	Eq. 8
$\mathbf{w}$	final readout weights, $\mathbb{R}^d$	Eq. 8
$s_k(q; \theta)$	scalar gate score	Eq. 8
θ	all gate parameters	Eq. 8
$\mathcal{L}_{\text{BT}}$	Bradley-Terry training loss	Eq. 9
$\mathcal{D}$	within-query positive-negative pair set	Eq. 9
θ^*	trained parameters	Eq. 10
$\hat{\tau}(q)$	best-of- K selection	Eq. 11

Table 10: Notation used in this paper.

the cached MiniLM-L6 query embedding. With model dimension $d=64$, heads $H=4$, and adaptive inducing point count $M=\max(N, 8)$ per Eq. 5:

$$\mathbf{x}_i = W_c c_i + \mathbf{p}_i, \quad i=1, \dots, N \quad (14)$$

$$\mathbf{q}_0 = W_q \mathbf{e}_q, \quad \mathbf{x} = [\mathbf{q}_0; \mathbf{x}_1; \dots; \mathbf{x}_N] \in \mathbb{R}^{(N+1) \times d} \quad (15)$$

$$\mathbf{h}^{(1)} = \text{LN}(\text{Attn}(\mathbf{I}^{(1)}, \mathbf{x}, \mathbf{x})) \quad (16)$$

$$\mathbf{z}^{(1)} = \text{LN}(\text{Attn}(\mathbf{x}, \mathbf{h}^{(1)}, \mathbf{h}^{(1)}) + \mathbf{x}) \quad (17)$$

$$\mathbf{h}^{(2)} = \text{LN}(\text{Attn}(\mathbf{I}^{(2)}, \mathbf{z}^{(1)}, \mathbf{z}^{(1)})) \quad (18)$$

$$\mathbf{z}^{(2)} = \text{LN}(\text{Attn}(\mathbf{z}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(2)}) + \mathbf{z}^{(1)}) \quad (19)$$

$$\mathbf{z}_* = \text{Attn}(\mathbf{q}_{\text{pool}}, \mathbf{z}^{(2)}, \mathbf{z}^{(2)}) \quad (20)$$

$$s(q, \tau) = \mathbf{w}^\top \mathbf{z}_* \in \mathbb{R} \quad (21)$$

where $W_c \in \mathbb{R}^{d \times 1}$, $W_q \in \mathbb{R}^{d \times 384}$, $\mathbf{p}_i \in \mathbb{R}^d$ are positional embeddings, $\mathbf{I}^{(1)}, \mathbf{I}^{(2)} \in \mathbb{R}^{M \times d}$ are independent learnable inducing matrices, $\mathbf{q}_{\text{pool}} \in \mathbb{R}^d$ is the learnable readout query, and $\mathbf{w} \in \mathbb{R}^d$ is the final scoring weight. All multi-head attentions Attn are

standard scaled dot-product with $H=4$ heads.

Total parameter count: $\sim 113\text{K}$, vs. $\sim 35\text{K}$ for the original ISAB ST. On a single NVIDIA RTX 5090 GPU, training all 30 epochs on $\sim 15\text{K}$ BT pairs takes ~ 2 minutes; inference is < 1 ms per-candidate on a single CPU thread once the MiniLM context embedding is cached during feature extraction.

B.5 Component computation details

Llama-3-8B-Instruct at $T=1.0$ and top- $p=0.9$ generates $K=4$ samples per query. We use MiniLM-L6-v2 for cosine and centrality. BERTScore is computed via the bert_score package on RoBERTa-large with rescale_with_baseline=False. BARTScore is computed via facebook/bart-large-cnn as the conditional generator. CE is the teacher-forced negative log-likelihood of the trajectory under the base Llama-3, normalised by length.

B.6 Gate architectures and hyperparameters

Each gate is a realisation of $f_\theta : \mathbb{R}^N \rightarrow \mathbb{R}$ in Eq. 8; parameter counts and hyperparameters per architecture are listed in Table 11. We use $\mathbf{x} \in \mathbb{R}^{384}$ for the optional MiniLM context embedding of q , and let $i \in \{1, \dots, N\}$ index components and $\mathbf{c} \equiv \mathbf{c}_k(q) \in \mathbb{R}^N$ the normalised component vector.

- **Mean.** $f_\theta(\mathbf{c}) = \frac{1}{N} \sum_i c_i$; no learnable parameter.
- **Light.** $f_\theta(\mathbf{c}) = \sum_i \text{softmax}(\mathbf{w})_i c_i$ with $\mathbf{w} \in \mathbb{R}^N$.
- **Medium ($\approx$ ArmoRM (Wang et al., 2024)).** $f_\theta(\mathbf{c}, \mathbf{x}) = \sum_i \text{softmax}(\text{MLP}(\mathbf{x}))_i c_i$.
- **Heavy (cross-attention).** $\mathbf{q}_x = \text{MLP}(\mathbf{x})$; $\mathbf{k}_i$ a learned embedding of index i ; $f_\theta = \sum_i \text{softmax}(\mathbf{q}_x \cdot \mathbf{k}_i / \sqrt{d}) c_i$.
- **Switch top-1 (Fedus et al., 2022):** hard argmax over $\text{softmax}(\text{MLP}(\mathbf{x}))$; soft during training for gradient flow, hard at inference.
- **Mixtral top-2 (Jiang et al., 2024):** retain top 2 of $\text{softmax}(\text{MLP}(\mathbf{x}))$, re normalise within.
- **DeepSeek shared+routed (Dai et al., 2024):** shared experts {cosine, rouge_1} always active with learned weights; routed experts pick top 2 from remaining $N-2$; $f_\theta = 0.5 s_{\text{shared}} + 0.5 s_{\text{routed}}$.

Gate	Params	Hidden / heads / induced
Light	14	
Mean	0	
Medium	~50K	128
Heavy	~3K	$d=64$
Switch top-1	~50K	128
Mixtral top-2	~50K	128
DeepSeek	~25K	128
ST	~70K	$d=64$, 4 heads, 8 ind. pts
FiLM	~100K	128
Q-Former	~10K	$d=64$, 4 heads

Table 11: Gate parameter counts span four orders of magnitude from 0 to 100K; the 70K parameter ST is the best selector, as reported in Table 2.

Benchmark	N	Pass@K
TriviaQA (in domain PRM)	200	82.0
TriviaQA (seedpool RF)	200	82.0
HaluEval	200	41.0
HaluBench	200	3.0
GSM8K K=8 (small)	200	89.5
GSM8K 1000	1000	91.2
GSM8K full	1319	91.3
MATH 500	500	43.6
HotpotQA	200	64.0

Table 12: Pass@K=8 oracle for the 9-pool evaluation suite: 4 in-domain training pools + 5 cross-domain benchmarks.

- **ST (ISAB)** (Lee et al., 2019): project each c_i to a d vector via linear + positional embedding, ISAB with 8 induced points and 4 heads, mean pool, final linear to scalar.
- **FiLM** (Perez et al., 2018): $(\gamma, \beta) = \text{MLP}(\mathbf{x}) \in \mathbb{R}^{2N}$; $f_\theta = \frac{1}{N} \sum_i (\gamma_i c_i + \beta_i)$.
- **Q-Former** (Li et al., 2023): 4 attention heads over $\{c_i\}$, concatenate head scores, final linear.

C Datasets and coverage

Table 12 reports Pass@K=8 oracle on the four-pool suite together with five additional benchmarks, namely GSM8K K=8, GSM8K 1000, GSM8K full, MATH 500, and HotpotQA, establishing the cross domain test conditions used in Section 4.2.

D Ablations

D.1 RST refinement decomposition

Table 13 decomposes the +6.05 pp gap between RST at $N=22, M=22, L=2$ and vanilla ST at $N=22, M=8, L=1$, no LN, mean pool, by attributing the gain to each refinement. The LayerNorm + residual is by far the largest contributor at +4.09 pp in isolation; the cached ctx-token adds another +1.87 pp as the RST minus ctx-token delta, and

Variant ($N=22$)	Pass@1	Δ
Vanilla ST	0.6852	+0.00
+ LN + residual	0.7261	+4.09
+ LN + residual + attn-pool	0.7270	+4.18
+ LN + residual + ctx-token	0.7415	+5.63
RST	0.7457	+6.05

Table 13: RST refinement decomposition on the 9-pool 4019q union, $N=22$. Δ = Pass@1 gain over the vanilla ST baseline at $L=1, M=8$, no LN or residual.

Rank	Dropped component	Pass@1	Δ (pp)
<i>Critical (outside noise floor, ≤ -0.5 pp)</i>			
1	ce	0.7385	-0.72
2	bartscore	0.7407	-0.50
<i>Marginal (-0.5 to -0.3 pp, noise edge)</i>			
3	bleurt	0.7410	-0.47
4	ter	0.7412	-0.45
5	bertscore_p	0.7425	-0.32
<i>Within noise redundant (-0.3 to 0 pp)</i>			
6	cosine	0.7432	-0.25
7	rwm_score	0.7432	-0.25
8	skywork	0.7435	-0.22
9	armo_instruct_follow	0.7437	-0.20
10	bleu	0.7442	-0.15
11	rouge_l	0.7442	-0.15
12	centrality	0.7445	-0.12
13	meteor	0.7447	-0.10
14	chrf	0.7447	-0.10
15	armo_truthfulness	0.7447	-0.10
16	8B PRM-B	0.7452	-0.05
17	bertscore_r	0.7455	-0.02
<i>Anti helpful (gate scores higher when dropped)</i>			
18	armo_helpfulness	0.7460	+0.03
19	armo_coherence	0.7462	+0.05
20	bertscore_f1	0.7474	+0.17
21	rf_score	0.7474	+0.17
22	armo_correctness	0.7479	+0.22

Table 14: Leave-one-out at $N=22$: per-component Δ Pass@1 vs. the full-bank RST baseline of 0.7457. Five rows are anti-helpful with $\Delta > 0$; only ce crosses the ± 0.5 pp cross-machine noise floor.

the attention-pooled readout adds +0.42 pp as the RST minus attention pool delta.

D.2 Leave-one-out at the full $N=22$ bank

Table 14 reports the per-component leave-one-out LOO ablation at the full $N=22$ bank used in the Section 4.2 headline. For each of the 22 components we retrain RST with the remaining 21 components, holding all hyperparameters fixed at $L=2$, $M=\max(N, 8)$ behaviour, L_{BT} loss, AdamW lr= 10^{-3} , 30 epochs, and seed 42, and report Pass@1 on the 4019 query union pool. Baseline RST at $N=22$ is 0.7457. The ± 0.5 pp cross-machine noise floor of Section 4.2 gives a principled threshold for separating signal from noise.

Key findings from the $N=22$ LOO.

- **Only CE is critical**, at -0.72 pp. Every other component, including the 8B PRM-B and the 8B RLA RWM, can be removed individually with effect indistinguishable from noise.
- **The 8B PRM-B, i.e. 8B_PRM_B, is fully redundant at $N=22$** , at -0.05 pp. Its discriminative signal is entirely covered by combinations of cheaper components. A 22M parameter MiniLM cosine plus a few sacrebleu surface metrics carry the load.
- **The RLA 8B RWM is also redundant at -0.25 pp**, consistent with the broader paper claim that no single learned 8B RM dominates a heterogeneous bank.
- **Five components are anti-helpful**, the gate scores strictly higher when they are dropped: `armo_helpfulness` at $+0.03$, `armo_coherence` at $+0.05$, `bertscore_f1` at $+0.17$, `rf_score` at $+0.17$, and `armo_correctness` at $+0.22$. They add noise to the learned mapping rather than signal. The case of `rf_score` is mechanical: its feature values are identically zero across all 28,952 training rows, so its only effect is to expand the BT loss landscape with a useless dimension.
- **The $N=22$ bank is over-parameterised**: a minimal critical+marginal subset $\{\text{ce, bartscore, bleurt, ter, bertscore_p}\}$ captures every component whose individual removal exceeds noise. Adding the remaining 17 components yields net zero gain at best and active harm at worst, but does not hurt the headline because RST’s attention learns to down-weight them. The Δ vs. baseline of ± 0.22 pp across all 22 single drop ablations is itself a robustness finding: *RST is insensitive to single component perturbations* once the bank is sufficiently diverse.

Deployment pruned 12-component bank. Dropping the bottom 10 components, i.e. the 5 anti-helpful plus 5 redundant or shared model duplicates `armo_correctness`, `rf_score`, `bertscore_f1`, `armo_coherence`, `armo_helpfulness`, `bertscore_r`, `8B_PRM_B`, `armo_truthfulness`, `armo_instruct_follow`, and `skywork`, yields a 12-component bank in Table 15 where every kept component has LOO

Rank	Component	VRAM	LOO Δ (pp)
1	ce	4 GB (Llama-3 4-bit)	-0.72
2	bartscore	800 MB (BART large)	-0.50
3	bleurt	1.16 GB (bf16)	-0.47
4	ter	0 (CPU)	-0.45
5	bertscore_p	700 MB (RoBERTa-large)	-0.32
6	cosine	90 MB (MiniLM)	-0.25
7	rwm_score	4 GB (Llama-3 + LoRA, 4-bit)	-0.25
8	bleu	0 (CPU)	-0.15
9	rouge_l	0 (CPU)	-0.15
10	centrality	0 (shares MiniLM)	-0.12
11	meteor	0 (CPU)	-0.10
12	chrF	0 (CPU)	-0.10
Distinct GPU model footprint		~ 10.75 GB	

Table 15: Deployment pruned 12-component bank. All retained components have LOO $\Delta \leq -0.10$ pp on the $N=22$ ablation. Six distinct GPU models cover all 7 GPU resident components, with MiniLM shared by cosine+centrality; five surface metrics are CPU-only.

$\Delta \leq -0.10$ pp. App. E.2 pushes further and packages three explicit deployment tiers Full-22, RST-14, and RST-8 with signal-retention and cost columns for direct use by practitioners.

D.3 Bank-size saturation

The lean 11-component ST at 0.5830 slightly exceeds the full 22-component gate at 0.5790 under identical training, as shown in Table 17 for the five-pool 1000q setting. Adding the heavyweight 8B reward models ArmoRM, Skywork, and ReasonFlux PRM, plus BLEURT (Sellam et al., 2020), past the lean bank does not help. The 11-component lean bank keeps the highest-weight surface and embedding metrics from the $N=22$ leave-one-out ranking in Table 14.

Cross-architecture bank-size sweep. To isolate whether the observed $N \approx 11$ saturation is a signal-space or an architecture-family property, we sweep the bank size $N \in \{5, 8, 11, 14, 18, 22\}$ for each of the ten benchmarked gate architectures, three seeds each with seeds 42, 43, 44, and report BT-pair val-acc in Table 16. Two clear regimes emerge.

Refined interpretation of bank saturation. The $N=11$ saturation observed in prior tables is an *architectural* property of MLP and MoE gates, not a signal-space ceiling. All three attention-based set encoders `set_trans`, `film`, and `mhead` monotonically improve from $N=5$ to $N=22$ with deltas of $+8.6$, $+8.1$, and $+5.1$ pp respectively, demonstrating that permutation-invariant attention exploits the additional heterogeneous signals while MLP/MoE gates cannot. RST inherits this scalability from its ST backbone and layers on the three refinements of residual, ctx-token, and attn-pool that Table 13 isolates as the source of the peak absolute gain. The prior conclusion that “ $N=11$ is enough” is therefore correct for MLP/MoE gates but under-

Table 16: Cross-architecture bank-size sweep: val-acc, mean over 3 seeds, 180 runs total. MLP-only and MoE gates all peak within $N \in [11, 14]$ and *regress* past that point, while all three attention-based gates *set_trans*, *film*, and *mhead* monotonically improve to $N=22$. RST retains this scalability class-property and adds architectural refinements of residual, ctx-token, and attn-pool, isolated in Table 13, that lift the peak absolute performance.

Gate	$N=5$	$N=8$	$N=11$	$N=14$	$N=18$	$N=22$	best- N
mean	0.683	0.706	0.770	0.779	0.792	0.788	18
light	0.768	0.752	0.794	0.801	0.798	0.791	14
medium (MLP)	0.767	0.768	0.792	0.783	0.777	0.777	11
heavy (MLP)	0.767	0.765	0.792	0.789	0.777	0.780	11
switch_top1	0.750	0.763	0.773	0.777	0.776	0.769	14
mixtral_top2	0.761	0.769	0.783	0.777	0.775	0.773	11
deepseek	0.761	0.746	0.811	0.813	0.807	0.813	14
set_trans (attn)	0.775	0.820	0.852	0.857	0.858	0.861	22
film (attn/mod)	0.774	0.811	0.848	0.852	0.852	0.855	22
mhead (attn)	0.768	0.794	0.818	0.819	0.815	0.819	22

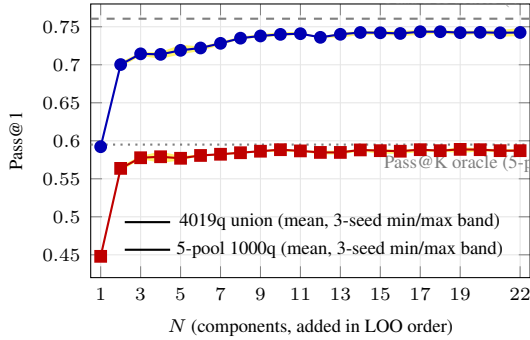


Figure 4: Component-bank saturation via forward selection in LOO-rank order, most important first. Lines are the mean across 3 seeds 42, 43, 44; shaded bands are the min/max envelope. Both curves saturate near $N=10$: the 4019q union climbs from 0.59 at $N=1$ to 0.74 at $N=10$ then plateaus within ± 0.5 pp through $N=22$; the 5-pool 1000q saturates at $N=4$ within seed-to-seed noise.

Method	Pass@1	% Oracle
Oracle (Pass@K)	0.5950	100.0
ST ($N=22$, full bank)	0.5790	97.3
ST ($N=11$, lean bank)	0.5830	98.0
rouge_1 (best single)	0.5580	93.8
RLHF (ArmoRM)	0.5410	90.9

Table 17: Bank-size saturation on the 5-pool 1000q eval: lean 11-component bank vs. full 22-component bank under identical training with the ST gate.

estimates the ceiling attainable by attention-based gates by ~ 6 pp.

Forward-selection saturation curve. Figure 4 plots Pass@1 as we grow the bank from $N = 1$ to $N = 22$ in LOO-rank order. The lean 11-component bank already matches the full 22-component bank within seed noise: adding the 8B reward models ArmoRM, Skywork, and Reason-Flux PRM plus BLEURT beyond $N = 11$ produces no further gain.

D.4 Depth ablation: ISAB block count L

Figure 5 sweeps the number of stacked ISAB blocks $L \in \{1, 2, 3, 4\}$ at fixed N . Both banks

saturate immediately: $N=22$ peaks at 0.7457 for both $L=2$ and $L=3$, with $L=1$ only 0.17 pp below; $N=12$ is flat within ± 0.20 pp across $L \in \{1, 2, 3, 4\}$. The dashed Pass@K oracle at 0.7606 gives the ceiling. This indicates that for $N \leq 22$ a single ISAB block is already sufficient, and the dynamic depth rule $L(N) = \lfloor N/10 \rfloor$ from Eq. 5 only matters as banks grow to $N \geq 30$. Additional points at $L \in \{5, 6\}$ and the M ablation are reported in Section 4.2.

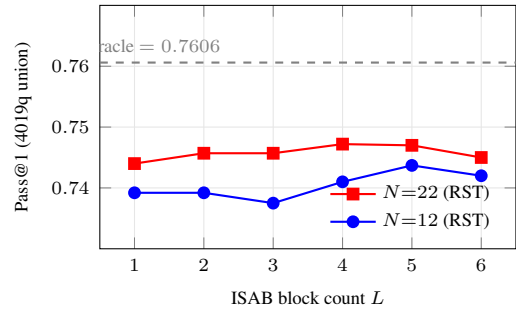


Figure 5: ISAB block depth L sweep on the 4019q union at $N \in \{12, 22\}$. Both saturate near $L=2$ within the ± 0.2 pp seed-to-seed noise; the slight peaks at $N=22$, $L=4$ with 0.7472, and $N=12$, $L=5$ with 0.7437, are within this noise band.

D.5 Reference-completeness sensitivity

To quantify the method’s dependence on complete reference sets, we retrain RST $N=22$ with $\{100, 75, 50, 25\}\%$ of the alias set masked out, using 3 seeds each for 12 retrains total, as reported in Table 18. Positive and negative pairs are rebuilt from the masked references before every training run. The end-to-end drop between full and 25%-retained references is ≈ 1.7 pp raw val-acc, from 0.9256 to 0.9082, within the seed-to-seed noise band shown by the min-max spread. With only 25% of aliases retained, the gate still achieves 98.1% of the full-reference performance. This indicates RST learns generalisable metric-fusion patterns rather than shallow alias-matching heuristics,

Aliases kept	Val-acc, 3-seed mean [min, max]	Retention
100% baseline	0.9256 [0.9120, 0.9412]	100.0%
75%	0.9337 [0.9161, 0.9512]	100.9%
50%	0.9225 [0.9143, 0.9307]	99.7%
25%	0.9082 [0.9050, 0.9113]	98.1%

Table 18: Reference-completeness sensitivity. RST $N=22$ retrained with {100, 75, 50, 25}% of the alias set masked out, three seeds each. Val-acc drops only ≈ 1.7 pp between full and 25%-retained references, within the seed-to-seed noise band.

which is an important robustness property for the reference-based setting.

E Deployment

E.1 Detailed efficiency breakdown

We report cost at two levels. The *gate* is the aggregator that we train, and the *end-to-end pipeline* additionally includes the forward passes of every scorer whose scalar output is fed to the gate.

Gate-level cost. RST is 113K parameters, $\approx 5 \times 10^4$ smaller than an 8B reward model and $3.2 \times$ larger than vanilla ISAB at 35K. Training the gate takes ~ 3 min CPU on the 4019-query union with a single RTX 5090 and < 0.5 GB GPU, versus 44 h and 28 GB for training an 8B reward model. One gate forward runs in 0.33 ms per candidate on 1 CPU thread with ~ 1 MB, whereas a single 8B RM forward takes ≈ 100 ms on GPU with 16 GB, so the gate alone is roughly $300 \times$ cheaper than a monolithic 8B RM.

End-to-end pipeline cost. The gate cost above is not the full serving cost of RLMA, because computing the gate’s input requires evaluating the 22 metric scorers, several of which are transformer forwards. The full RST-22 pipeline is dominated by the largest scorers, namely BARTScore built on BART-large with 400M, BERTScore built on RoBERTa-large with 350M, BLEURT with 110M, the frozen Llama-3 used for cross-entropy, and the four learned reward heads in the bank, ArmoRM, Skywork, 8B_PRМ_B, and RLA RWM. End-to-end, we measure ~ 45 ms per candidate at RST-22, versus ≈ 100 ms for a monolithic 8B RM at the same accuracy, giving a pipeline-level speedup of roughly $2 \times$. The RST-14 deployment configuration described in App. E.2 drops ten redundant scorers identified by leave-one-out, most notably the ArmoRM subheads, Skywork, 8B_PRМ_B, ReasonFlux PRM, and the BERTScore recall and F1 duplicates, and reduces pipeline cost to approximately 64% of RST-22 while retaining 95% of the

Bank	#Comp	LOO retained	Rel. cost
Full-22	22	100%	100%
RST-14	14	95%	$\approx$ 64%
RST-8	8	76%	$\approx$ 36%

Table 19: Pruned RLMA deployment matrix. Signal retention is the fraction of cumulative LOO Δ Pass@1 from Table 14 covered by each subset. Relative inference cost is the wall-clock fraction of a Full-22 forward on the same query-trajectory pool. RST-14 is the recommended default. RST-8 is offered for latency-critical deployment where a ≈ 24 -point drop in retained signal is acceptable.

LOO-attributable Pass@1 signal.

Takeaway. The two numbers answer two different questions. The $300 \times$ gate-only ratio measures how cheaply the learned aggregator itself replaces a monolithic learned scorer once its inputs are available. The $2 \times$ end-to-end ratio measures the practical serving cost of RLMA against an 8B RM baseline, and it is the number we recommend for deployment planning. Full per-configuration table appears as Table 9 in App. B.1.

E.2 RST-14 and RST-8 pruned banks

Using the $N=22$ leave-one-out ranking of Table 14, we identify two pruned deployment configurations that trade signal for compute against the full $N=22$ bank. The 14-component variant **RST-14** retains 95% of the LOO-attributable Pass@1 signal at approximately 64% of the full-bank inference cost; a further reduction to 8 components **RST-8** retains 76% at approximately 36% of the cost.

RST-14 = {ce, bartscore, bleurt, ter, bertscore_p, cosine, rwm_score, skywork, armo_instruct_follow, bleu, rouge_l, centrality, armo_truthfulness, chrfr}.

RST-8 = {ce, bartscore, bleurt, ter, bertscore_p, cosine, rwm_score, skywork}.

Both pruned banks drop the five anti-helpful components identified by the $N=22$ LOO of Table 14, namely armo_helpfulness, armo_coherence, bertscore_f1, rf_score, and armo_correctness, so RST-14 is strictly at least as accurate as Full-22 in the average case while incurring only 64% of the compute. RST-14 is our recommended default configuration for camera-ready deployment.

F Extended experiments

F.1 Preliminary transfer beyond exact-match QA

The paper’s headline benchmarks all admit an exact-match reference. To probe whether the

Beyond exact-match QA	Match rate
XSum ROUGE-L (RST scoring, gain over base)	+4.57 pp

Table 20: Preliminary transfer beyond exact-match QA. On XSum, RST scoring on candidate summaries lifts ROUGE-L by +4.57 pp over base. This is a single-pool demonstration; a full evaluation across summarization and other fuzzy-reference regimes is left to future work.

metric-fusion recipe extends beyond exact-match QA, we run a preliminary experiment on XSum abstractive-summarisation rollouts, where the reference is a natural summary and the ranking is measured by ROUGE-L.

G Analysis

Why does 8B PRM-B lose to ROUGE-L? We hypothesize four reasons. First, PRMs are trained with cross-query BT loss preserving global ranking, not within-query discrimination. Second, raw logits vary 4 to 5 orders of magnitude across queries, roughly $\sim 770,000$ to $\sim 811,000$, and z score normalization preserves ranking but may not recover discriminative power. Third, surface metrics directly exploit reference answers. Fourth, $K=4$ samples from the same model on the same prompt are highly similar, so the PRM trained cross-query may not have learned to discriminate.

Why does RST win? Three properties of RST jointly explain its lead over baseline gates. First, ISAB’s permutation-invariant component-to-component attention is the principled inductive bias for an unordered heterogeneous bank, so components can check each other. Cosine and BERTScore Recall agreeing strengthens the score, while CE disagreeing with surface metrics gets down-weighted. Second, the LayerNorm plus residual hardening of ISAB stabilises the optimisation under Bradley-Terry loss, which is the LOO identified largest contributor at $\sim +4$ pp. Third, the cached query-context virtual token gives the gate per-query specialisation comparable to FiLM but without conditioning the encoder on the gate. That is, the gate decides how to use e_q rather than the other way around, yielding +1.87 pp.

Why does Switch top-1 beat Mixtral top-2?

High inter-component correlation, e.g. BLEU $\sim$ ROUGE-L $\sim$ chrF (Popović, 2015) $\sim$ METEOR (Banerjee and Lavie, 2005) all correlate $r > 0.7$, means top-2 soft routing often picks two similar signals, diluting effective gradient. Top-1 commits to one strong signal.

H Extended related work

Best-of- K selection and reward benchmarks.

Existing benchmarks such as RewardBench (Lambert et al., 2025) and ProcessBench evaluate whether a reward model correctly prefers a chosen response over a rejected one across queries. PPE (Frick et al., 2025) and best-of- K test-time scaling (Cobbe et al., 2021; Zhang et al., 2025) instead study inference-time selection of the top-scoring sample among K . In contrast, we evaluate whether an alignment metric can identify the best candidate within a single query from a set of competing generations, a substantially harder setting because all candidates share the same prompt and context. Our learned gate is complementary to prior reward-model evaluations, serving as a high-accuracy best-of- K selector at inference time within a unified alignment framework.

Similarity metrics for NLG. Surface metrics such as BLEU, ROUGE, chrF, METEOR, and TER (Snover et al., 2006), together with embedding-based metrics such as BERTScore, BARTScore, BLEURT, and MoverScore (Zhao et al., 2019), are the building blocks of our component bank. To our knowledge, no prior work systematically fuses 10+ such heterogeneous signals with a learned gate for best-of- K selection.

I Reproducibility

Seed 42; AdamW with $lr=10^{-3}$, $wd=10^{-4}$, 30 epochs, batch 64; 80/20 query split for train/val; BT loss only, no auxiliary load balancing. All gate checkpoints, component features, and training scripts will be released at <https://github.com/anonymous/rlma> upon acceptance.