

Mixture of Distillations: Specialist RLVR without Forgetting

Research Exploration

Rohan Sikand · Stanford University

March 2026

The Trilemma of Multi-Task RLVR

You want a single model that reasons well across math, code, and science. Three options:

Approach	Method	Problem
Sequential	RLVR on A, then B, then C	Catastrophic forgetting
Simultaneous	Multi-objective RLVR	Interference; infra doesn't exist
Recovery	Sequential + distill (GLM-5)	Retroactive fix; extra stage

Missing option: **train specialists independently, combine via routing.**

No forgetting (specialists never train on each other's tasks). No interference (training is independent). No recovery needed (nothing is lost in the first place). The question is: how do you combine K specialists into one deployable model?

Mixture of Experts vs. Mixture of Distillations

Standard MoE (Mixtral, DeepSeek-MoE):

- Architectural: multiple FFN experts per layer
- Learned router selects experts per token
- Trained end-to-end from pre-training
- Experts are sub-networks within one model
- Routing at the *parameter* level

Mixture of Distillations (ours):

- Training paradigm: multiple RLVR specialists
- Competence router selects teacher per problem
- Specialists trained independently, merged post-hoc
- Experts are full policies (separate models)
- Routing at the *policy* level

MoE answers: “which parameters should process this token?”

Mixture of Distillations answers: “which specialist’s knowledge should teach the student this problem?”

Phase 1: Train specialists independently

$$\pi_{\text{base}} \xrightarrow[\text{RLVR}]{\text{math}} \pi_{\text{math}}^* \quad \pi_{\text{base}} \xrightarrow[\text{RLVR}]{\text{code}} \pi_{\text{code}}^* \quad \pi_{\text{base}} \xrightarrow[\text{RLVR}]{\text{science}} \pi_{\text{science}}^*$$

Phase 2: Distill with routing into unified student

$$\pi_{\theta}^{\text{student}} \leftarrow \text{distill}(\pi_{\text{math}}^*, \pi_{\text{code}}^*, \pi_{\text{science}}^*; \text{router } w)$$

$$\hat{A}_{i,t} = \sum_k w_k(s_i) \cdot \text{sg} \left[\log \frac{\pi_{\theta_k}^*(y_{i,t}|x, y_{i,<t})}{\pi_{\theta}^{\text{student}}(y_{i,t}|x, y_{i,<t})} \right]$$

The student learns from whichever specialist is most competent for each problem. The router $w_k(s_i)$ determines the weighting. GLM-5's teacher-based advantage (with $G = 1$) is the distillation mechanism.

The Router: Which Specialist Teaches Which Problem?

Four options, from simple to learned:

Option 1: Domain-labeled routing (simplest)

Problem is tagged as math/code/science. Route to the corresponding specialist. Hard routing, no ambiguity.

Option 2: Competence-based routing

$$w_k(s_i) \propto \text{Pass}@1_k(s_i)$$

Route to whichever specialist solves the problem most reliably. Computed from a small evaluation set. Allows soft routing: a math problem that involves coding gets partial weight from both specialists.

Why Routing Matters (It's Not Just Weighted Averaging)

Naive multi-teacher distillation: Equal weights for all teachers on all problems.

$$\hat{A}_{i,t} = \frac{1}{K} \sum_k \hat{A}_{i,t}^{(k)}$$

The math specialist gives contradictory signal on code problems. The code specialist gives contradictory signal on math problems. Signals cancel out. Student learns a mediocre average.

Routed distillation: Each teacher only teaches what it knows.

$$\hat{A}_{i,t} = \sum_k w_k(s_i) \cdot \hat{A}_{i,t}^{(k)}, \quad w_k(s_i) \approx \begin{cases} 1 & \text{if specialist } k \text{ is competent} \\ 0 & \text{otherwise} \end{cases}$$

No contradictory signals. Each problem gets clean gradient from the right expert. The student inherits *what to do* and implicitly *when to do it* — the routing signal teaches task recognition.

The $G = 1$ Insight (from GLM-5)

Standard GRPO needs G rollouts per prompt to compute group-relative advantages. In Mixture of Distillations, the specialists provide the reference directly:

Standard multi-task GRPO:

- 32 prompts $\times$ 16 rollouts = 512 generations
- Group comparison within each prompt
- Compute-intensive generation

Mixture of Distillations:

- 1024 prompts $\times$ 1 rollout = 1024 generations
- Advantage from specialist logit gap
- Massive throughput gain

The specialist's logits *are* the reference. No group comparison needed. This means the distillation phase is much cheaper per step than standard RLVR — you can process 16–32 $\times$ more prompts per batch.

Handling Cross-Domain Problems

Real problems don't respect clean domain boundaries. A coding problem may require mathematical reasoning. A science problem may require code.

Soft routing handles this naturally:

Problem	w_{math}	w_{code}	w_{science}
Pure algebra	0.95	0.03	0.02
Numerical simulation	0.30	0.60	0.10
Computational biology	0.10	0.40	0.50
Proof by construction	0.70	0.20	0.10

For cross-domain problems, the student gets a *blended* advantage signal: “approach this like a coder for the implementation parts and like a mathematician for the reasoning parts.” The per-token advantage can even shift mid-sequence — math specialist guides the setup, code specialist guides the implementation.

Per-Token Routing: Dynamic Specialist Selection

The teacher-based advantage operates at the **token level**:

$$\hat{A}_{i,t}^{(k)} = \text{sg} \left[\log \frac{\pi_{\theta_k}^*(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta}^{\text{student}}(y_{i,t} \mid x, y_{i,<t})} \right]$$

This means routing can also operate per-token:

$$\hat{A}_{i,t} = \sum_k w_k(s_i, y_{i,<t}) \cdot \hat{A}_{i,t}^{(k)}$$

where w_k depends on the context so far, not just the prompt.

Example: Student generating a solution to a computational math problem:

- Tokens 1–20 (problem analysis): math specialist dominates

Why This Solves Continual Learning

Traditional problem: Train on A, then B $\rightarrow$ forget A.

Mixture of Distillations:

1. Train specialist A on task A (no forgetting — only trains on A)
2. Train specialist B on task B (no forgetting — only trains on B)
3. Distill both into student with routing (student inherits both)

When task C arrives:

4. Train specialist C on task C from base model
5. Re-distill: student $\leftarrow \text{distill}(\pi_A^*, \pi_B^*, \pi_C^*; w)$

Or more efficiently:

4. Train specialist C

The Specialist Library

Over time, accumulate a library of RLVR specialists:

Specialist	Domain	Training	Status
π_1^*	Math (algebra, calculus)	RLVR, 400 steps	Frozen
π_2^*	Code (Python, algorithms)	RLVR, 400 steps	Frozen
π_3^*	Science (physics, chemistry)	RLVR, 400 steps	Frozen
π_4^*	Formal reasoning (proofs)	RLVR, 200 steps	Frozen
π_5^*	Tool use (API calls)	RLVR, 300 steps	In training

The unified student is re-distilled periodically from the full library. Each specialist is frozen after training — it never degrades. New capabilities are added by training new specialists and re-distilling, never by modifying existing ones.

This is a **modular, append-only** architecture for accumulating reasoning capabilities.

Comparison to Existing Approaches

Method	Forgetting?	Interference?	Mechanism
Sequential RLVR	Yes	N/A	Each stage degrades prior
Multi-objective RLVR	No	Yes (gradient)	Simultaneous optimization
GLM-5 recovery	Partial fix	During stages	Retroactive distillation
Model merging (TIES)	Partial	At merge time	Weight-space averaging
Standard MoE	No	Minimal	Architectural experts
Mixture of Distill.	No	No	Routed policy distillation

Key distinctions from model merging (TIES, DARE, etc.): those operate in **weight space** (average the parameters). We operate in **output space** (distill the behaviors). Weight merging is lossy and uncontrolled. Output distillation is targeted and routed.

Key distinction from MoE: MoE requires the expert architecture from pre-training. Mixture of Distillations works with any standard dense model — the “experts” are separate checkpoints, not architectural components.

Prior Work

Paper	Approach	Gap
GLM-5 (2025)	Cross-stage teacher distillation	Temporal; sequential forgetting then repair
Model merging (various)	TIES, DARE, SLERP	Weight-space; no routing; uncontrolled interference
Multi-teacher KD (various)	Distill from ensemble	Equal weighting; no routing; not for RLVR specialists
MoE (Mixtral, etc.)	Architectural experts	Requires MoE pre-training; not post-hoc composable
PackNet / HAT	Parameter isolation for CL	Not scaled to LLMs; architectural constraints

The gap: Problem-conditioned multi-teacher distillation where each teacher is an independently RLVR-trained domain specialist, composed into a unified dense model via competence routing. The individual pieces exist; the combination for RLVR is novel.

Experimental Plan

Phase 0: Specialist divergence (2 weeks)

- Train 3 specialists from Qwen3-4B: math, code, science (200 steps each)
- Measure: does each specialist outperform the base and the others on its domain?
- Go/no-go: if specialists don't differentiate, routing has nothing to route

Phase 1: Distillation comparison (3 weeks)

- Compare merging methods:
 - (a) Naive multi-teacher (equal weights)
 - (b) Domain-labeled routing (hard routing by task type)
 - (c) Competence-based routing (soft, by specialist pass rate)
 - (d) Sequential RLVR baseline (train on all three sequentially)
 - (e) Simultaneous multi-task RLVR baseline (if infra permits)
- Metrics: Pass@1 per domain, aggregate, catastrophic forgetting measure

Phase 2: Continual extension (2 weeks)

Theoretical Angle

Proposition (routing preserves specialist knowledge):

Let π_k^* be optimal for task k . Under hard routing ($w_k(s) = 1$ if s is task k), the distilled student π_θ^* satisfies:

$$\text{Pass@1}_{\pi_\theta^*}(s) \geq (1 - \epsilon) \cdot \text{Pass@1}_{\pi_k^*}(s) \quad \forall s \in \text{task } k$$

if the distillation converges. The student matches each specialist on its domain up to distillation error ϵ .

Proposition (no interference under routing):

Adding specialist $K+1$ with competence-based routing does not degrade performance on tasks $1, \dots, K$, because the new specialist receives zero or near-zero routing weight on existing task domains.

Risks

Risk	Severity	Mitigation
Specialists don't differentiate enough	High	Phase 0 validates; use different problem subsets
Distillation can't compress K specialists into one dense model	High	Start with $K=3$; increase if it works
Routing overhead at inference	Low	Router is a tiny classifier; negligible
K separate training runs = $K \times$ compute	Medium	Parallelizable; each run is shorter
Cross-domain problems have no clear specialist	Medium	Soft routing handles blending
Per-token routing is noisy	Medium	Start with per-problem routing; ablate

Honest Assessment

Pros:

- Clean conceptual contribution: MoE for RLVR at the policy level, not the architecture level
- Solves continual learning *by construction* — specialists never forget
- Append-only specialist library is a natural paradigm for accumulating capabilities
- $G = 1$ distillation from GLM-5 makes the merge phase efficient
- Per-token routing enables naturally interleaved cross-domain reasoning
- Works with any dense base model — no MoE architecture required

Cons:

- K separate RLVR runs is expensive (though parallelizable)