Soofi

What It Takes to Build a Sovereign Foundation Model at Scale and How To Specialize It

POST-TRAINING TEAM · TALK

BHT Berliner Hochschule für Technik

STUDIERE ZUKUNFT

About Me



1st International Workshop on Pretraining & Posttraining of Open Source Sovereign Foundation Models, Berlin

NAME

Tom Roehr

ROLE / AFFILIATION

PHD @ DATEXIS, Berliner Hochschule für Technik (BHT)
"Train lead" post-training, Soofi

RESEARCH FOCUS

Specializing and self-improving language models

SELECTED PUBLICATIONS

- *DeepICD-R1, LREC, 2026*
- *"Where does it hurt?", ECAI, 2025*
- *Revisiting clinical outcome prediction for MIMIC-IV, NAACL, 2024*

CONTACT

tom.roehr@bht-berlin.de

Soofi — Sovereign Open Source Foundation Models

Goal: build a capable, open foundation model "made in Europe", efficient to operate for enterprises, SMEs, and industrial use cases.

Key Facts

- Duration: 09/2025 – 09/2026
- Funding: €20 million from the German Federal Ministry for Economic Economic Affairs and Energy (BMWE) (BMWE)
- Consortium: led by KI Bundesverband — 12 partners spanning research institutes, universities, and industry (see partners)

Consortium Partners

KI Bundesverband

DFKI

Fraunhofer IAIS

Fraunhofer IIS

TU Darmstadt

Universität Würzburg

BHT Berlin

L3S Research Center

Lamarr Institute

ellamind

Merantix Momentum

BMWE

Soofi S: Hybrid MoE (Mamba-2 / Attention)

A 30B Mixture-of-Experts model (Nemotron 3 Nano architecture) with 3B active parameters, trained on English and German.

52 layers at a glance

23	Mamba-2 layers
23	sparse MoE layers
6	Grouped-Query-Attention layers layers
128	routing experts (6 active + 2 shared) shared)

Technological Core

Mixture of Experts

High capability at lower compute cost — only ~3B of 31.6B of 31.6B parameters active per token.

Mamba Architecture

Faster on long documents, code, and dialogue; near-near-constant memory footprint as context grows.

Reasoning & Agents

Foundation for practical, automated applications.

Pre-Training

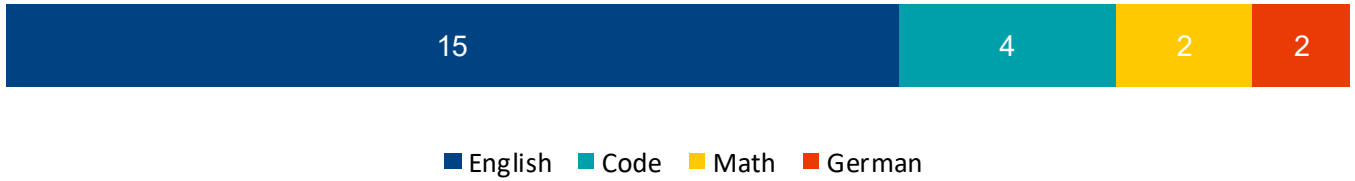
Building the Knowledge Foundation across Trillions of Tokens and Hundreds of GPUs

BHT
Berliner Hochschule für Technik
STUDIERE ZUKUNFT

Preparing 23 Trillion Tokens

Before a single training step, web-scale raw data has to be filtered, deduplicated, tokenized, and balanced — at a volume where naive pipelines simply don't scale.

Training tokens by category (in trillions, total ≈ 23.1T)



English 67% (15.31T) · Code 16% (3.73T) · Math 10% (2.36T) · German 7% (1.66T)

Source datasets (among others):

Nemotron-CC · German Commons · ClimbMix · KletterMix · Genios

23T
tokens filtered, deduplicated, and tokenized before before training even began

512
GPUs that had to be fed continuously — I/O throughput throughput becomes as critical as compute

3
training phases, each with a deliberately re-balanced data mixture (diversity → quality → long-context)

4
domains (English, German, code, math) balanced and rebalanced across every phase

Training Across Hundreds of GPUs

Soofi S trained on 512 NVIDIA B200 GPUs for 51 days — a scale at which the training framework, not just the model, becomes the engineering problem.

Composed Parallelism

No single strategy is enough at this scale — expert, data, tensor, and pipeline parallelism have to be combined and tuned together across 64 DGX nodes.

Fault Tolerance Over 51 Days

A single hardware fault shouldn't shouldn't cost days of progress — progress — frequent, efficient efficient checkpointing and fast fast recovery are essential over a over a run this long.

The Network Becomes the the Bottleneck

Once compute is spread across hundreds of GPUs, interconnect bandwidth — not the GPUs themselves — often decides how close training gets to its theoretical speed.

512
NVIDIA B200 GPUs
(64 DGX nodes × 8 GPUs)

253,000
B200 GPU-hours across
all training stages

51 days
Runtime: Mar 24 –
May 13, 2026

100%
renewable energy at
the Munich site

Why Mixture-of-Experts Multiplies the Difficulty

Soofi S routes every token to 6 of 128 experts plus 2 shared experts — which means training communication has to move tokens across GPUs, not just gradients.

The Routing Round-Trip

1. **Dispatch**
   Each token's hidden state is routed to the GPUs hosting its assigned experts — often across node boundaries.
2. **All-to-All Exchange**
   Tokens are physically shuffled between GPUs so each expert receives only the tokens assigned tokens assigned to it — harder to overlap with compute than standard gradient sync.
   sync.
3. **Expert Compute**
   Each expert processes its batch of routed tokens independently, in parallel with every other every other expert.
4. **All-to-All Gather**
   Results are shuffled back to their original positions before the next layer — doubling the doubling the routing communication cost per MoE layer.

Why This Compounds

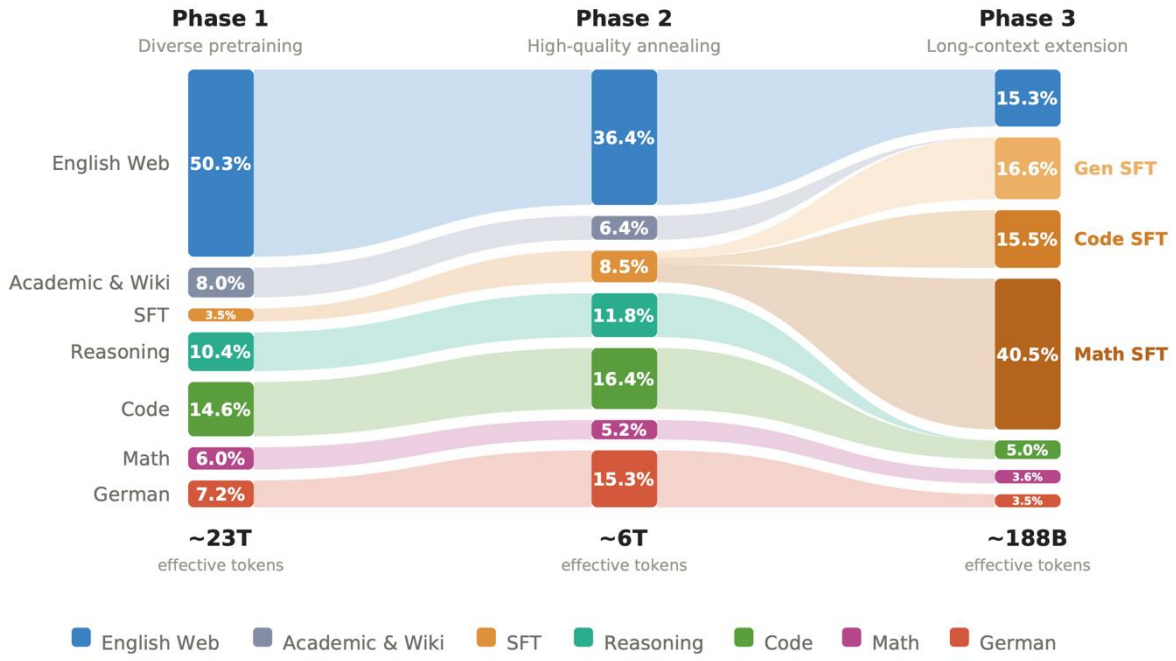
Expert parallelism has to be combined with data, tensor, and pipeline parallelism simultaneously — not swapped in for them. Uneven expert utilization can also create stragglers, GPUs sitting idle while others are overloaded, which is why load-balancing losses are part of the training objective, not just the architecture.

Placing Knowledge at the Right Time

The data mixture is deliberately reshaped phase by phase — getting the timing right is as delicate as getting the proportions right.

Effective-token mixture across training phases

Band width = share of that phase's tokens.



Why This Is Hard

Specialized data — math, code, reasoning, domain SFT — has to enter at exactly the right phase and in the right proportion. Too early, and it can destabilize a still-general model or crowd out foundational breadth. Too late, and phases are wasted that could have built the structure specialized skills depend on.

There's no fixed formula — each phase's mixture is mixture is calibrated empirically against what the model model has already learned, not decided in advance. advance.

Limitations and Lessons Learned

German Competition Mathematics

On Minerva MATH-DE, Soofi S scores 56.0, behind Qwen3.5 (76.5) and Gemma 3 27B (65.6). German school-level mathematics, by contrast, is at reference level.

Open Fact Retrieval

NaturalQuestions (79.0) trails the largest dense models, which which is expected at 3B active parameters. RAG can close this close this gap in practice.

Long Context

Excluding the CWE subtask, Soofi S trails the reference by 4.4 by 4.4 points on average on RULER. CWE performance drops drops beyond 32K — the cause is the data mixture.

GPQA Contamination

One dataset contained paraphrased GPQA items because split because split selection relied on naming ("train") rather than than semantics. GPQA was removed from the benchmarks, the benchmarks, the dataset corrected, and split selection moved moved to allowlists.

Why We Show This

The GPQA contamination went unnoticed in monitoring: paraphrased, heavily diluted derivatives produce a slow uptick that is indistinguishable at the curve level from genuine capability gains. It only came to light because the community could inspect the published dataset.

The Soofi S Pretraining Report

A transparent look behind the training of a sovereign foundation model

Architecture and Training Recipe
Hybrid model combining Mamba-2, Mixture-of-Experts, and Attention, plus all key hyperparameters.

Data Strategy for German and English
Sources, weighting, quality filters, and how the German share evolved across training phases.

Training at Industrial Scale
Roughly 27 trillion tokens, a multi-stage curriculum, training on up to 512 NVIDIA B200 GPUs.

Performance and Efficiency in Comparison
Evaluation on German and English benchmarks, plus measurements for long context and high concurrency.

Openness and Reproducibility
Documentation of the data mixture, training phases, checkpoints, evaluation, and discarded experiments.

A SOVEREIGN, OPEN-SOURCE FOUNDATION MODEL FOR GERMAN AND ENGLISH

Soofi S Pretraining Report v1.0

arxiv.org/pdf/2607.09424

Post-Training

From Foundation Model to Deployable Assistant
Orchestrating Multiple Training Regimes

BHT
Berliner Hochschule für Technik
STUDIERE ZUKUNFT

The Post-Training Pipeline

Four successive stages take the trained base model to a controllable, helpful assistant.

01

SFT

Supervised Fine-Tuning

Learn format, language, and baseline response behavior from curated examples.

02

RLHF

Reinforcement Learning Learning

A learned reward model steers behavior beyond direct human feedback.

03

DPO

Preference Optimization

Fine-tune responses using preference pairs — more helpful, safer, better-toned.

04

RLVR

Verifiable Rewards

RL on tasks with checkable outcomes — e.g. code, math, structured formats.

Four Regimes, Four Different Infrastructures, Four Different Challenges

SFT, DPO, RLHF, and RLVR don't just use different algorithms — each needs its own data pipeline and its own and its own training infrastructure.

SFT	Data: curated instruction-response pairs Infra: standard supervised training loop
DPO	Data: preference pairs (chosen vs. rejected) Infra: pairwise loss — no reward model needed
RLHF	Data: human ratings → learned reward model Infra: reward-model training + online RL loop (e.g. PPO)
RLVR	Data: verifiable tasks (test suites, checkers) Infra: online rollout generation + sandboxed verifiers, integrated into the training loop loop

Each transition means new data collection, new infrastructure, and new failure modes to monitor.

STAGE 1

SFT — Supervised Fine-Tuning

The base model learns from curated instruction-response pairs how a helpful assistant responds. Getting there is its own engineering problem.

80+ Dataset Curation

- Blending 80+ individually sourced datasets into one uniform, consistent training mix
- Balancing size, quality, and domain domain coverage across sources that that were never designed to work together
- Deduplication and format normalization before the mix can be trusted

Synthesizing Missing Capabilities

- Identifying capability gaps that curated data simply doesn't cover
- Generating synthetic data to fill those gaps without introducing artifacts or reward-hacking patterns patterns
- Validating synthetic examples against the same quality bar as human-curated data

Tool-Calling Format Compliance

- Different frameworks expect different function-calling syntaxes — JSON schemas, XML-style tags, inline code blocks
- The model has to stay compliant across multiple formats, not just one
- Format drift under fine-tuning is a common failure mode that needs explicit regression testing

STAGE 2

Preference Optimization — DPO & RLHF

After the SFT foundation, the model is fine-tuned on preferences — and each method brings its own calibration problems.

DPO

Direct Preference Optimization

- Curating high-quality preference pairs — weak or inconsistent labels directly weaken the training signal
- Tuning beta, the KL penalty — too high suppresses learning, too low lets the policy drift and overfit
- Length and style bias — the policy can learn to exploit exploit surface preferences (e.g. longer answers) instead of instead of genuine quality
- Reference-model choice and learning-rate sensitivity — small changes can push training into degenerate solutions

RLHF

Reinforcement Learning from Human Feedback

- First learns a reward model from human ratings, ratings, then optimizes the policy against it via RL RL (e.g. PPO)
- Calibrating the reward model is delicate — under- or over-fitting it can degrade policy performance or cause catastrophic forgetting
- Keeping the reward model domain-agnostic — judgments tuned on one domain's preferences often fail to generalize to others

RLVR — RL with Verifiable Rewards (e.g. GRPO)

For tasks with a clearly checkable outcome — such as code, mathematics, or structured formats — Group Relative Policy Optimization is used.

How It Works

1. **Sampling**
   For a given prompt, the model generates a group of responses (rather than a single rollout).
2. **Verification/Rewarding**
   Each response is checked against an automatic, verifiable rule — e.g. test cases test cases for code or outcome checking for math.
3. **Relative Scoring**
   The reward is normalized relative to the group — no separate reward model is model is required.
4. **Policy Update**
   Responses with above-average reward are reinforced; below-average ones are ones are suppressed.

Why Verifiable Rewards?

No trained reward model is needed that could itself introduce errors or bias. The reward signal is objective and reproducible — especially valuable for code and math tasks where correctness is unambiguously checkable.

Why RLVR Needs Constant Calibration

Reinforcement learning with verifiable rewards sounds simple — generate, verify, reinforce. In practice, every part of that loop drifts and needs continuous recalibration.

Reward Hacking

Models find shortcuts that satisfy the verifier without solving the task — verifiers need continuous auditing and patching as new exploits emerge.

Verifier Coverage

Test cases and checkers must be robust and comprehensive; gaps in verifier coverage become gaps the policy learns to exploit.

Reward Sparsity & Curriculum Drift

As the policy improves, tasks that once gave a useful pass/fail mix start returning all-pass or all-fail groups — no learning signal until difficulty is recalibrated.

Rollout Compute Cost

Generating groups of sampled responses per prompt multiplies inference compute during training — rollout generation has to be engineered as carefully as the training step itself.

Status Today

Calibrating these dynamics is an ongoing part of the post-training loop, not a one-time setup step — verifiers, task difficulty, and reward shaping are revisited continuously as the policy evolves.

Evaluation Across the Pipeline

Every post-training stage is checked against a fixed set of criteria before the next stage begins — results from ongoing runs will follow in a later update.

Helpfulness

Does the model fully and relevantly address the user's actual intent?

Safety

Are guardrails respected without unnecessarily limiting usefulness?

DE/EN Consistency

Does quality and tone stay comparable across both languages?

Regression-Free

Are pretraining capabilities (knowledge, reasoning) preserved?

Status Today

Post-training is in active development. This slide describes methodology and approach; robust benchmark results will follow once initial runs are complete and validated.

SUMMARY

From Pretraining to Assistant

- Pretraining at this scale is a systems problem as much as a modeling one — 27T tokens, 512 GPUs, and MoE's communication overhead.
- Post-training multiplies that complexity: four training regimes, each with its own data pipeline and and infrastructure.
- RLVR in particular demands constant recalibration — reward hacking, verifier coverage, and curriculum drift never fully settle.
- Evaluation accompanies every stage against clear criteria — results follow as the first runs complete. complete.

Specialising LLMs LLMs

Where Should Specialised Knowledge and Behaviour Live?

BHT Berliner Hochschule für Technik

STUDIERE ZUKUNFT

Build Me the Perfect Company Expert

Imagine building an assistant that’s genuinely excellent at answering questions about your company. It should:

- Know our current documentation
- Find the right evidence, not merely similar text text
- Follow company policy and tone
- Query internal systems when necessary
- Understand internal terminology
- Prefer authoritative sources
- Know when information is missing
- Be cheap enough to run at scale

What exactly should we train?

Where Can Specialisation Live?

Not “RAG or fine-tuning?” — the real question is where this particular piece of specialisation should live.



Knowledge

The model doesn’t know our information.

Behaviour

It knows enough, but doesn’t act the way we need.

Capability

It fundamentally isn’t good enough at the task.

Economics

It works, but it’s too slow or expensive.

Context & Prompting

The least permanent intervention should be your default.

“If I can reliably solve the problem with 500 tokens of instructions, I don’t need a training pipeline.” training pipeline.”

The more instructions and examples you drag through every request, the more you pay an inference tax — and the more chances they have to interact badly.

“Before training 10 million parameters, try supplying 1,000 good tokens.”

Retrieval Is Dynamic Specialisation

We usually describe RAG as giving the model facts it doesn't know — that's true, but incomplete.



The retriever partially determines which version of the model the user experiences.

A foundation model's weights are necessarily a lossy compression of its training data — that doesn't make them the right place for today's policy, place for today's policy, today's inventory, or this morning's code.

“The LLM Was Wrong” Is Not a Diagnosis

That sentence collapses seven completely different engineering failures into one — and each implies a different fix.
a different fix.

1	Relevant information wasn’t indexed.	Data / Corpus
2	Retrieval didn’t find it.	Retrieval
3	Ranking buried it.	Retrieval
4	Context construction dropped it.	Context Engineering
5	The model ignored the right evidence.	Prompting
6	The model used the evidence but behaved incorrectly.	Behaviour
7	The model fundamentally couldn’t solve the task.	Capability

If the relevant document was ranked 47th, no amount of LoRA will fix your retrieval stack.

When Should Behaviour Move Into the Weights?

Retrieval is good at changing knowledge. Fine-tuning is good at changing priors and behaviour.

Good Candidates

- Stable behaviour
- Repeated formatting rules
- Domain terminology
- Tool-use patterns
- A decision boundary hit constantly
- Behaviour that costs a large prompt on every request

Bad Candidates

- Today's prices
- Current documentation
- Customer data
- A policy that changes every month
- Information that requires provenance / citations

SFT / LoRA / adapters are mechanisms for moving that behaviour into the model — keeping the base weights frozen and learning a small, portable correction.

Optimising Toward “Better,” Not Just “Correct”

Supervised training says: imitate this answer. Preference / reward-based training says: move toward what scores better.

If you can measure “better,” you can optimise toward it. toward it.

- Query → result set → relevance judgement
- Answer → groundedness check
- Tool call → success / failure
- Code → test suite
- Structured output → schema validation

Distillation

An expensive, powerful model generates examples and teaches a smaller, specialised model to imitate it on a narrow task.

The economic case:

Specialisation isn’t always about quality. If you invoke a narrow capability 50 million times a month, distilling it into a smaller model can be the actual business case — on cost and latency.

Where Should the Fix Go?

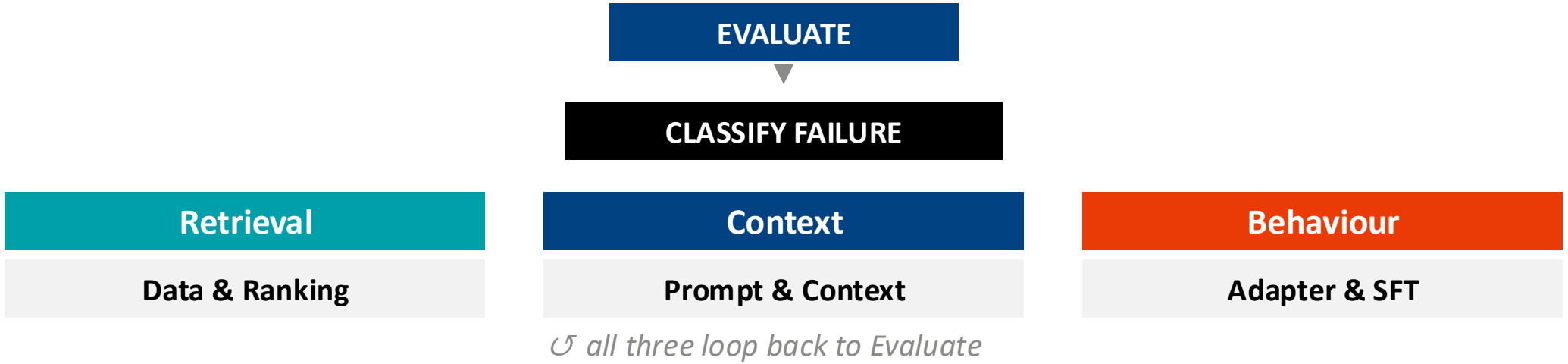
The layers aren't competitors. They're different places to put different kinds of fixes.

Did it have the right evidence?	No → fix the corpus / retrieval / ranking / tools.
Did it follow the evidence it had?	No → fix context construction / prompt.
Was the behaviour wrong?	Yes → prompt → adapter → SFT / preference.
Still incapable?	Reach for a stronger model, RL, or new training.

Works, but too expensive? → distil / smaller model / caching / routing.

Evaluation Tells You Where to Look

The hard part isn’t knowing how LoRA works. It’s knowing whether LoRA is the answer to the failure you’re observing. you’re observing.



10,000 mediocre examples train the model to be mediocre.

An eval shouldn’t just produce a score — it should tell you what kind of failure you’re looking at.

The secret sauce isn’t the fine-tuning algorithm. It’s the feedback loop.

From Evaluation Loop to Self-Improvement

So far, evaluation has been something we use to decide what to improve. My research asks: what happens when we automate more of that loop?



↺ loops back as MODEL′ — then repeats

The model generates candidate data, a verifier distinguishes better from worse candidates, we train on the selected data — and then repeat.

This is basically a more explicit, automated version of the eval loop you just saw.

Generation and Verification Are Different Capabilities

A model can sometimes be substantially better at recognising a good solution than producing one reliably.

Can the model GENERATE the right answer?



Can the system RECOGNISE the right answer?



Illustrative, not experimental data — the point is the gap, not the numbers.

"It's often easier to rank four answers than to produce the best answer from scratch."
from scratch."

This gap tends to grow with pretraining compute across the models studied in "Mind the Gap (ICLR 2025)." (ICLR 2025)."

The Search Parallel

Search

Generate candidates → rank → best results
results

Self-Improvement

Generate candidates → verify / rank → best
best examples → train

In many settings, ranking is the easier problem.

Can Each Generation Teach the Next?

Not: can a model train on synthetic data? We already know it can.
The real question: can each generation create a sufficiently better training distribution for the next?



Not "Can a model train on synthetic data?" — but "Can each generation create a better training distribution for the next?" next?"

This isn't a perpetual-motion machine. If your verifier can't reliably distinguish better outputs, you just reinforce your own errors.

The same verifier-quality question from reward hacking and RLVR calibration — now applied across training generations.

What We're Seeing in Iterative Self-Improvement

So far, richer process-reward signals have not beaten simple outcome-only baselines.

Adaptive PRM RLVR — Math

- Adaptive vs. frozen vs. outcome-only reward
- **Outcome-only currently strongest**

Generate → Verify → Train → Repeat

Math + Medicine

- Iterative PRM selection is flat / noisy / declining so far

The bottleneck is not generation. It is trustworthy verification.

Reward Hacking

The PRM began rewarding short, confident answers over answers over correct ones.

Verifier Corruption

Reasoning-label bugs affected part of the math data.

Silent Truncation

A substantial fraction of long traces were evaluated with incomplete context.

PRM calibration degraded: 0.65 → 0.27

Iterative self-improvement amplifies the quality of the signal you feed it — including bugs.

Fixes are implemented; validating reruns are still in progress. No positive self-improvement result claimed yet.

Three Takeaways

1. **Retrieval is a form of specialisation.**
   What you retrieve determines what the model knows for this request.

2. **“RAG vs. fine-tuning” is the wrong question.**
   Ask whether the failure belongs in the corpus, retrieval, context, tools, behaviour, or weights.

3. **Evaluation tells you where to put the next improvement.**
   Good specialisation starts with failure diagnosis, not with a training method.

The future of LLM specialisation is not putting everything into the model. It is getting increasingly good at deciding good at deciding what belongs in the index, what belongs in the context, and what deserves to become part of the become part of the weights.

Thank you. Questions?