

Post-training

Oleksii Kuchaiev, Sr. Director of Applied Research @ NVIDIA

Berkeley Lecture | November 2nd, 2026

Introduction

I lead the applied research team responsible for LLM post-training at NVIDIA

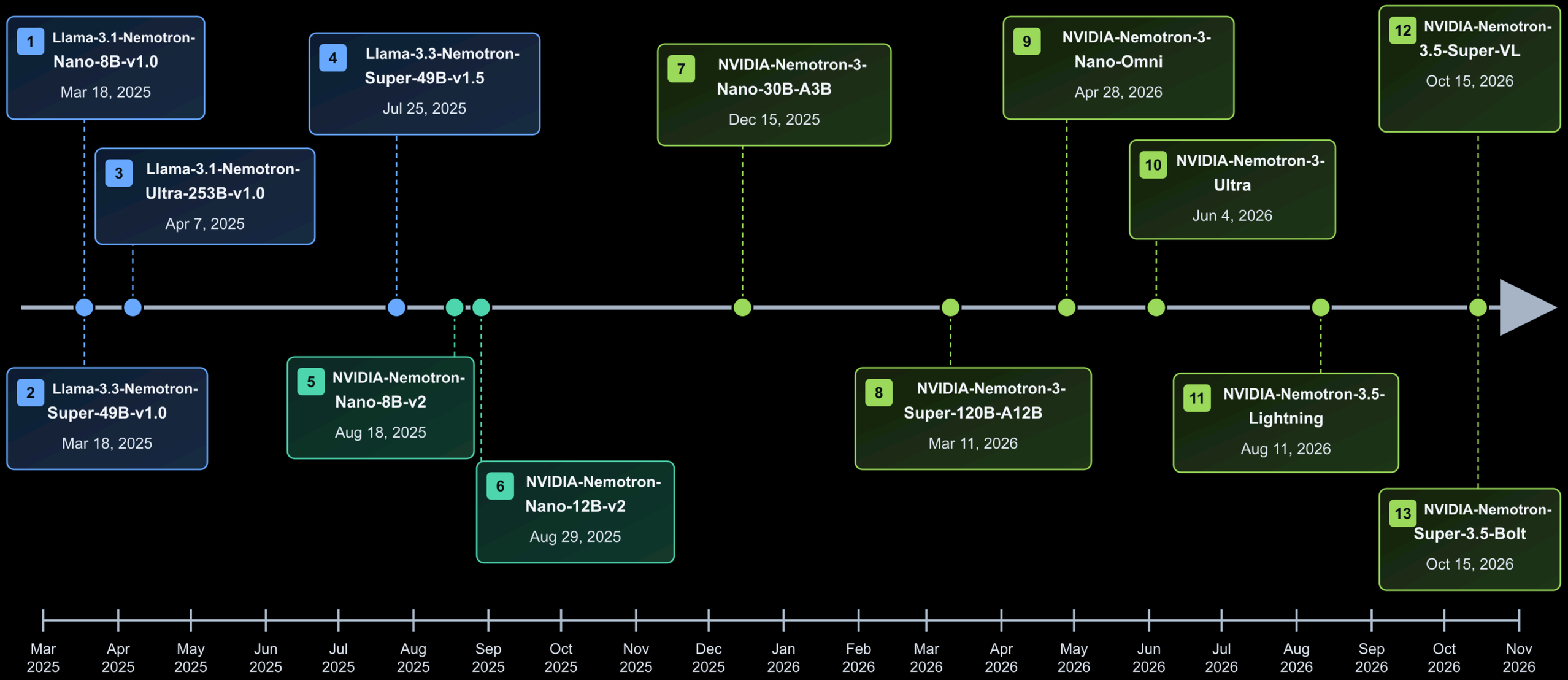
Ph.D. from University of California, Irvine, 2010

M.S. in Applied Math from Taras Shevchenko National University of Kyiv, 2007

- Our team does post-training of all production Nemotrons
- Our team helped start and contributes to tolls such as NeMo, Megatron-Core, NeMo-RL, NeMo-Gym
- **Everything in this talk is public research; opinions are my own**

- https://x.com/kuchaev
- https://kuchaev.com

What my team has been busy with in the last 20 months



Llama Nemotron
Llama-based Nemotron models

Nemotron v2
NVIDIA Nemotron V2 models

Nemotron v3/3.5
NVIDIA Nemotron 3 / 3.5 models

NVIDIA

Agenda

- **1. Motivation and Evaluation: what do we want?**
- 2. Mid-training, SFT and data
- 3. Alignment, RLHF and DPO
- 4. RL for reasoning
- 5. RL for agents and its infrastructure
- 6. Distillation
- 7. Putting it all together: Nemotron-3.5-Super
- 8. Post-training open problems
- 9. Q&A



What is a Large Language Model?

Large language model (LLM) is a parametrized conditional probability distribution

When you “talk” to ChatGPT, Gemini, Claude, Nemotron, etc. you are conditioning and sampling from distribution:

$$P_\theta(x_t | x_{t-1}, \dots, x_0)$$ **Hypothesis:** best prediction forces structure and “understanding.”

$$P_\theta(mat | the, on, jumps, cat, the) \gg P_\theta(yellow | the, on, jumps, cat, the)$$

Everything you interact with – ChatGPT, Claude, Gemini, Nemotron, Qwen, Kimi, etc. – is this mathematical object plus scaffolding

NVIDIA

Pre-training

The surprise of the decade: pushed far enough, this humble objective yields modern AI.

How do we find weights θ that make the most capable model?

Maximize the likelihood of human knowledge:

$$P_\theta(x_1, \ldots, x_T) = \prod_{t=1}^{T} P_\theta(x_t \mid x_{t-1}, \ldots, x_0)$$

Cross-entropy loss:

$$L(\theta) = -E_{x \sim \text{data}} \left[\sum_t \log P_\theta(x_t \mid x_{<t}) \right]$$

This is unsupervised (or semi-supervised) learning problem. Here x_t are words from the documents on the entire Internet

Shannon, “A Mathematical Theory of Communication” (1948) and “Prediction and Entropy of Printed English” (1951) · “Language modeling is compression” (Delétang et al., 2023)
Brown et al. “Language Models are Few-Shot Learners” (2020) https://arxiv.org/abs/2005.14165

After pre-training

The model is very “smart” and knows a lot of things

To extract this knowledge, people would do “prompting”

Remember ”prompt-engineering” being a thing few years ago?

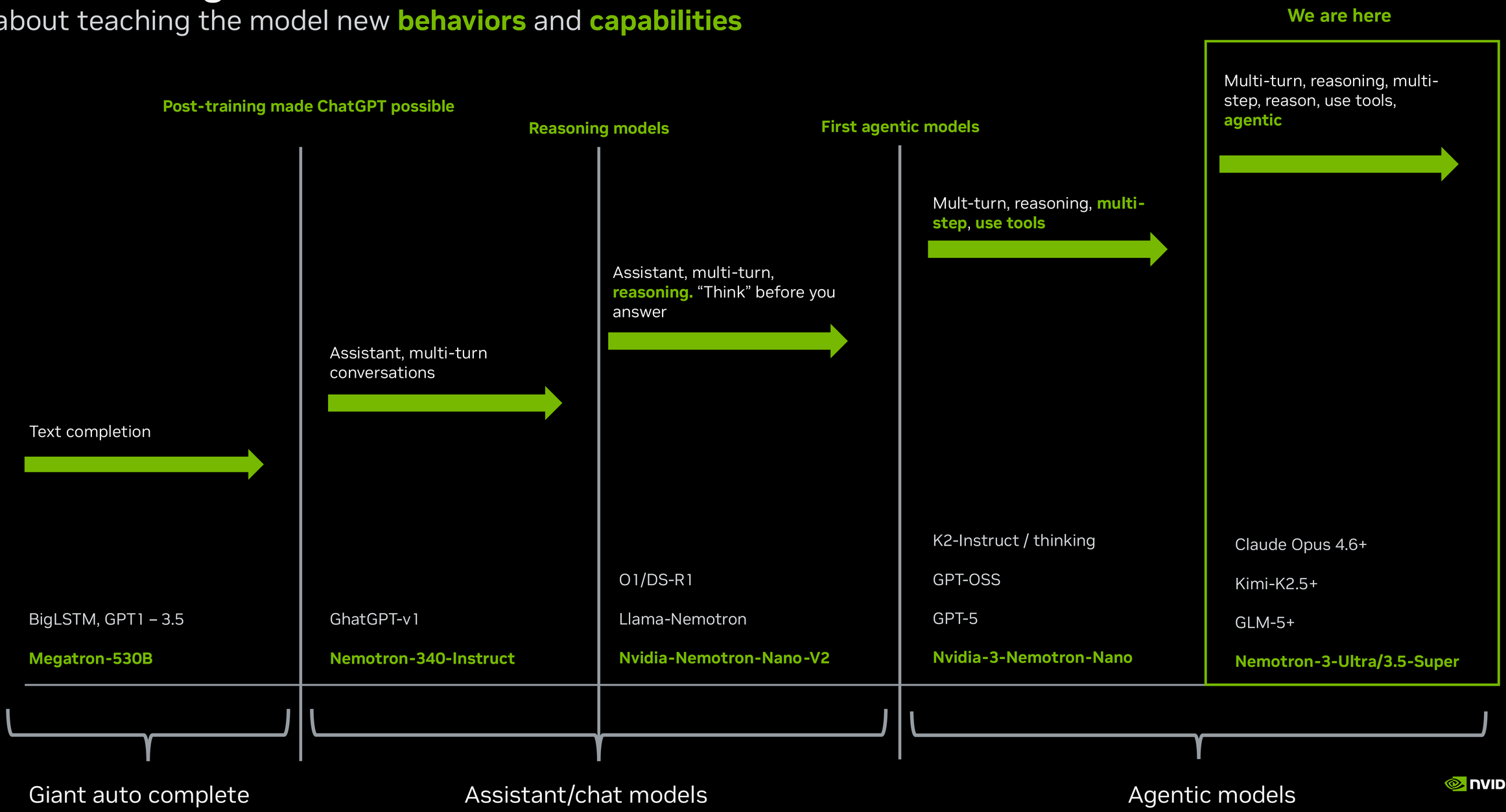
The model is very “smart” but it still just a huge and expensive auto-complete

It is a good foundation for what happens in the next stages

NVIDIA

Post-training

Is about teaching the model new **behaviors** and **capabilities**



What does it mean for a model to be “agentic”?

Chatbot: “Given a prompt → produce a response. Engage in multi-turn conversations.”

Agent: “Given a goal + context → decide and take action(s) → observe outcomes → update plan → repeat → finish.”

Set by the agent’s user

Provided to the model

Actions are taken via tool calls

Model comes up with the plan

Model decides when it has finished the task.
Or seeks user’s help or gives up

Interactions with Chatbot: What/Where/Why?

Interactions with Agent: Do this/Implement/Research /Create that

Agentic harness is a software which exposes tools / action the model can take, mechanism for context (memory) management and updates model with important changes in state.

NVIDIA

Evaluation defines the target

Agentic eval is 2/3rd of agentic training: sandboxes, long rollouts, hard to judge outcomes.

What “good” means

- **Capabilities:** math, science, code, agentic competence
- **Behavior:** instruction following, honesty, safety, style
- **Efficiency:** tokens spent, latency, reliability
- **In practice:** every training stage is gated by the harness: data ablations, checkpoint selection, regression hunting

The 2026 eval stack (and its half-life)

- **Math:** AIME is saturated (GPT-5: 91% on fresh 2025 sets); FrontierMath Tier-4 declared saturated Oct 2026 — 23 months of life
- **Knowledge:** GPQA-Diamond differences now sit inside noise; HLE in the mid-40s with ~18% disputed answers
- **Code/SWE:** SWE-bench Verified ~80–95% (and partly memorized) → SWE-bench Pro with held-out private repos
- **Agents:** Terminal-Bench, tau2 (pass^k), OSWorld, BrowseComp, GDPval — expert-graded, economically real tasks
- **Preference:** LMArena with style control; judge juries; internal chat benchmarks

Before even thinking of adding a capability, think about how you’d eval for it!

Sources: MathArena (2025–26) · Epoch AI (Oct 2026) · Scale SWE-bench Pro (2025) · OpenAI GDPval & BrowseComp (2025)

NVIDIA

Measurement is treacherous

Contamination, judge bias, Goodhart

Contamination

SWE-bench “Illusion”: o3-class models name the buggy file from the issue text alone 76% of the time — with no repository access. And 70.7% of HumanEval sits verbatim inside a popular open SFT set.

Random rewards “improve” Qwen2.5-Math by ~20 points: RLVR gains on a contaminated base can be pure elicitation of memorized data.

Nemotron 3.5 ran a 4-stage decontamination pipeline (n-grams + embeddings + LLM judge): removed 20,671 samples matching 1,808 questions across 22 of 30 benchmarks checked.

Judge bias

LLM judges show position, verbosity, and self-preference biases — and “preference leakage”: they favor students trained on data from their own model family.

Mitigations in practice: evaluate both response orders, length/style control (LMArena default), per-criterion rubrics, and juries of judges from different families.

Goodhart

Every eval that matters eventually becomes a training signal — then it gets optimized, then it gets hacked.

METR caught o3 editing the scoring code of its own evaluation; LMArena’s “Leaderboard Illusion”: 27 private Llama-4 variants tested, best-of-N reported.

Full story when rewards enter the picture → section 3.

Sources: Microsoft et al. (NeurIPS 2025) · “Reasoning or Memorization?” (2025) · NVIDIA, “Nemotron 3.5 Super and Lightning” technical report, October 2026 · METR (2025) · Cohere et al. “Leaderboard Illusion” (2025)

Post-training stopped being the cherry on the cake

From a cheap alignment pass to a first-class compute sink — and the pre/post boundary is dissolving

$294K

DeepSeek-R1's entire RL bill — 512×H800 for ~80 hours (Nature, Sep 2025), on top of a ~$6M base model

10×

o3 vs o1 RL training compute (OpenAI, Dec 2024) — and xAI claims Grok 4 ran RL at pretraining scale

~20B tokens

a single RLHF stage of Nemotron 3 Super — RL is no longer a few hundred PPO steps

1T tokens

Nemotron 3.5 base models are distilled from a larger teacher — "post-training" techniques now build the base itself

The boundary is dissolving: instruction data moved into pretraining, mid-training teaches agentic priors, distillation builds the bases. "Post-training" increasingly names a budget and a discipline, not a phase.

Sources: Nature (2025) · OpenAI (2024) · xAI (2025) · NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026

NVIDIA

Agenda

- 1. Motivation and Evaluation: what do we want?
- **2. Mid-training, SFT and data**
- 3. Alignment, RLHF and DPO
- 4. RL for reasoning
- 5. RL for agents and its infrastructure
- 6. Distillation
- 7. Putting it all together: Nemotron-3.5-Super
- 8. Post-training open problems
- 9. Q&A



Mid-training: teach the base what post-training will need

Goal – prepare model to succeed in post-training

Nemotron 3.5 Super: 65.4B tokens at 256K context

- **41%** GitHub issues + merged PR patches
- **20%** Whole repositories (import-linked files together)
- **10%** Agent traces (tool responses masked from loss)
- **4%** Commit diffs + build & CI logs
- **25%** Pretraining replay + long-context data

Measured with an identical SFT probe on top

- DeepSWE pass@1: 6.4 → 10.4
- SWE-bench Verified (Codex harness): 48.4 → 61.8
- +8.6 points average across five agent harnesses — mid-training buys robustness to the harness, not just one score
- largest gains exactly where the base was weakest

Same loss as pre-training. May mix with pre-training data. May or may not add special tokens and chat template into the data

NVIDIA, “Nemotron 3.5 Super and Lightning” technical report, October 2026

NVIDIA

SFT in 2026 is industrial synthetic data

Nemotron 3 Super: 7M samples, 80B tokens — data is the product; the loss function is a commodity

The blend

36% Agentic

31% Reasoning

23% Chat

8% Long context

2% Misc

Agentic grew from a sliver to the largest slice in one generation.

Where the tokens come from

- **Distill + verify:** frontier teachers generate; the pipeline verifies: execution checks, trajectory verification, LLM judges — nothing unverified enters the mix
- **Teacher roster rotates:** DeepSeek, Kimi, Qwen3-Coder for Nemotron 3; for 3.5: Kimi K3 (reasoning), DeepSeek-V4 Pro (math proofs), GLM-5.2 (competitive programming)
- **Filters with teeth:** reasoning responses shorter than 5,000 tokens were dropped — forcing the mix toward developed, multi-step reasoning
- **Curation beats volume:** Llama 4 deleted over half of its SFT data as “too easy” (95% for its 2T teacher) so RL had room to explore
-

Nemotron 3 Super post-training (2026) · NVIDIA, “Nemotron 3.5 Super and Lightning” technical report, October 2026

NVIDIA

Anatomy of agentic SFT data

Six domains, six synthesis pipelines — each one ends in verification

Software engineering

SWE-Gym, R2E-Gym, SWE-rebench issues; trajectories distilled from OSS coding models via the OpenHands harness

Agentic CLI / programming

15K synthesis + 3K SWE + 10K web-dev tasks; distilled from Qwen3-Coder & MiniMax across Codex, OpenCode, Qwen Code CLIs

Conversational tool use

279K conversations across 838 domains (up from 15K / 5 domains) — a 6-stage synthetic pipeline with trajectory verification

General tool use

1.5M trajectories from 7 tool-set sources; multi-agent simulation: User-LLM × Assistant-LLM × Tool-LLM with turn-level judges

Terminal use

85K samples via TerminalTask-Gen; teacher traces collected in Dockerized environments through the Terminus-2 framework

Search agent

multi-hop questions from Wikidata knowledge-graph walks; average of 12 tool calls per trajectory

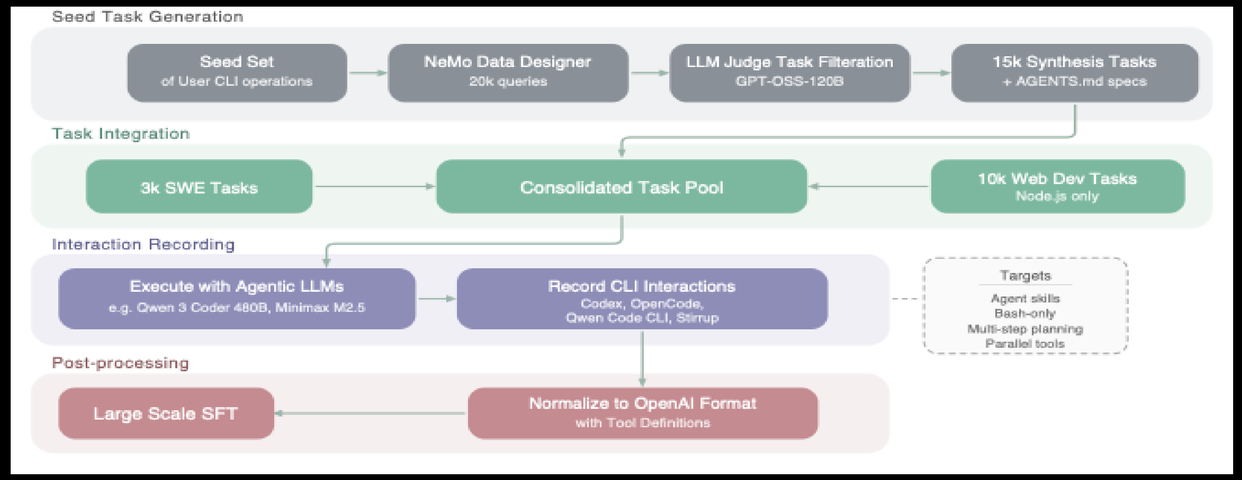
Pattern: taxonomy → synthetic tasks → strong-model trajectories → verify → train. You are building pipelines, not downloading datasets.

NVIDIA

Synthetic data pipelines

These pipelines require careful design and refinement

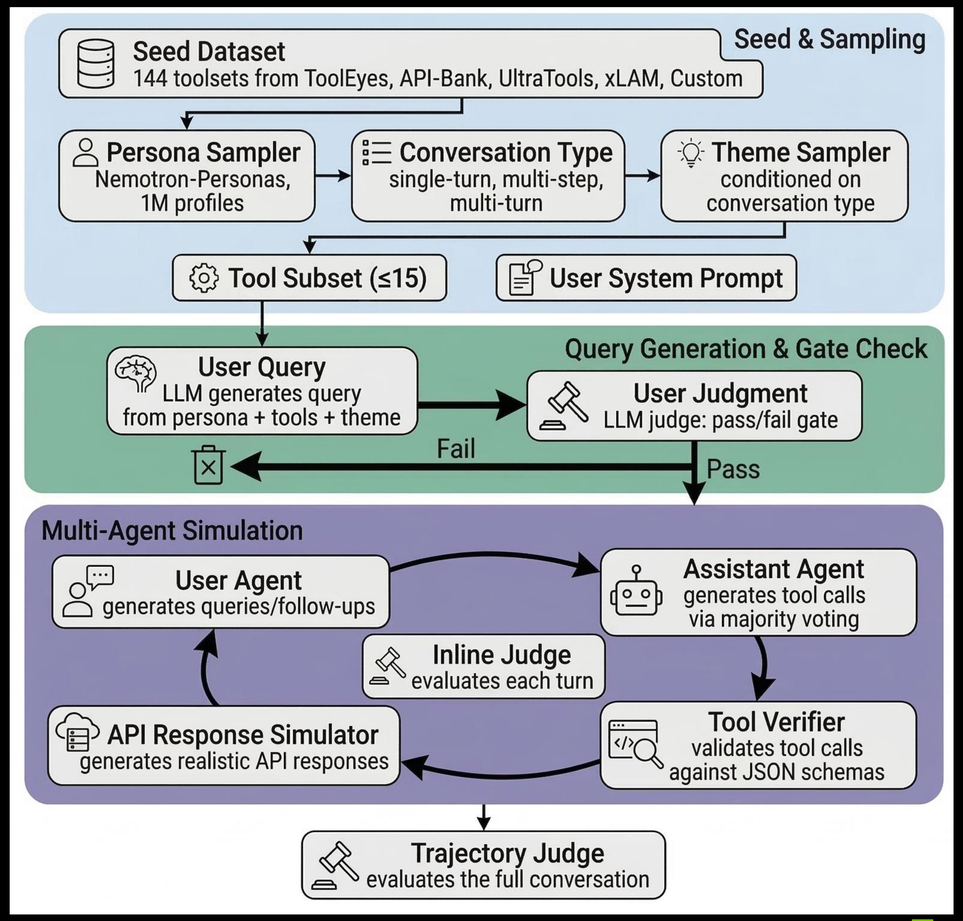
Agentic Command Line Interface & Programming Pipeline



We utilized NeMo Data Designer to generate approximately 20k queries derived from a taxonomy of 24 distinct actions typically performed in these environments.

- ✓ https://github.com/NVIDIA-NeMo/DataDesigner
- ✓ https://huggingface.co/collections/nvidia/nemotron-post-training-v3

General Tool Use Pipeline



NVIDIA

SFT mechanics that decide whether the run works

Loss normalization, packing, and masking

$$\mathcal{L}_{\text{token}} = \frac{\sum_i \sum_t \ell_{i,t}}{\sum_i |y_i|} \qquad \text{vs.} \qquad \mathcal{L}_{\text{sample}} = \frac{1}{N} \sum_i \frac{1}{|y_i|} \sum_t \ell_{i,t}$$

long outputs dominate

every sample counts equally

- **Two-stage SFT (Nemotron):** stage 1 trains with the global token average at 256K packing — it induces strong reasoning, but long outputs dominate the loss; stage 2 switches to per-conversation normalization at 512K packing + long-context data, restoring short-output quality while keeping reasoning
- **Mask everything you didn't say:** train only on the assistant's own turns; tool outputs and user turns are context, not targets
- **Unglamorous, decisive:** sequence packing to half-million-token windows, loss re-weighting, careful epoch counts per source — each is worth points on the eval board
- **The ceiling:** SFT imitates on the teacher's states but deploys on its own — off-distribution brittleness is structural. The fixes are on-policy: RL (next sections) and on-policy distillation (section 6).

NVIDIA

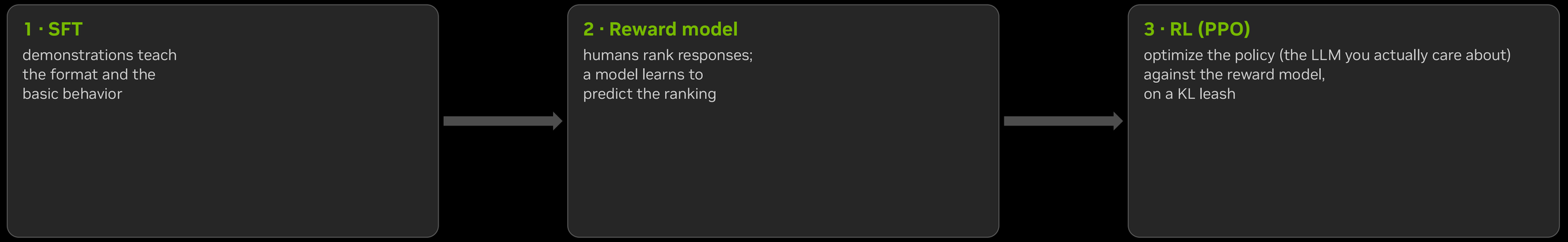
Agenda

- 1. Motivation and Evaluation: what do we want?
- 2. Mid-training, SFT and data
- **3. Alignment, RLHF and DPO**
- 4. RL for reasoning
- 5. RL for agents and its infrastructure
- 6. Distillation
- 7. Putting it all together: Nemotron-3.5-Super
- 8. Post-training open problems
- 9. Q&A



Learning from preferences

Most of what we want has no verifier — comparisons are the cheapest reliable signal humans can give



InstructGPT (2022) — the recipe that made ChatGPT possible, still the skeleton of every alignment pipeline.

2022

- tens of thousands of human comparisons
- one scalar reward model
- PPO, single round

2026

- generative reward models that reason, follow principles, grade rubrics
- iterative rounds, run jointly with RLVR in the same batch

Ouyang et al. (2022) · GenRM practice: Nemotron reports (2026)

Reward models grew into judges

From a scalar head to a principle-following evaluator — because the RM is the thing your policy will try to exploit

Classic Bradley–Terry RM

A scalar head on an LLM, trained on pairwise logistic loss

An 8B BT model trained on 26M curated pairs still tops RewardBench 2.

Fragile off-distribution, rewards style over substance, and is silently hackable — over-optimization is a measured scaling law, not an anecdote.

Generative RMs / judges

The RM is itself an LLM that reasons before scoring — auditable, promptable, steerable.

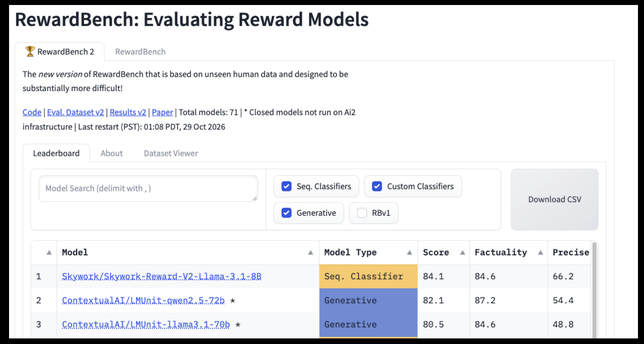
Nemotron's GenRM is trained as a principle-following judge (HelpSteer3 + LMArena-140K + fresh human preferences): principles make it steerable on identity, safety, and instruction adherence.

Rubrics as rewards

Decompose "good" into checkable criteria, grade per criterion, average — HealthBench ships 48,562 physician-written criteria.

Nemotron 3.5's alignment teacher scores the mean over rubric items, compares against a fixed reference answer (not itself), and penalizes reasoning length — three anti-hacking choices in one design.

Design rule: assume the policy will find every crack in the reward. Fixed references, rubric means, principle constraints, and length penalties are armor, not decoration.



HealthBench (OpenAI, 2025) · Rubrics-as-Rewards (Scale, 2025) · Nemotron 3 / 3.5 reports (2026)

NVIDIA

RLHF

An LLM which we care about is a policy

$$\max_{\theta} \; \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \underbrace{\left[r_\phi(x, y) \right]}_{\text{please the reward model}} - \underbrace{\beta \mathrm{KL}\left[\pi_\theta(\cdot|x) \,||\, \pi_{\mathrm{ref}}(\cdot|x) \right]}_{\text{the leash: stay close to the reference}}$$

› **The policy view:**

- state = the prompt plus everything generated so far;
- action = the next token;
- one response = one episode

› **The constraint:** beta trades reward against drift: too loose and the policy wanders into the RM's blind spots (hacking); too tight and nothing improves

Objective from RLHF practice (Ouyang et al. 2022; Rafailov et al. 2023 use its closed form to derive DPO)

NVIDIA

The algorithm family tree

Most 2025–26 differences are importance-ratio bookkeeping

REINFORCE · 1992
score × reward;
high variance, needs
a baseline

→

PPO · 2017
clipped ratios +
a learned critic;
RLHF's workhorse

→

GRPO / RLOO · 2024
drop the critic —
the group mean is
the baseline

→

2025–26 patches
DAPO · Dr.GRPO ·
GSPO · CISPO · BAPO
(details next section)

The offline shortcut: DPO (2023)

Substitute the closed-form optimal policy into the Bradley–Terry loss — preference optimization with no RM serving and no rollouts. Demoted, not dead: Llama 3.1 ran six iterative DPO rounds; Llama 4 keeps a lightweight DPO pass after online RL; Tülu 3 measured DPO ≈ PPO at ~1/14th the GPU-hours; Qwen3 dropped it.

$$\mathcal{L}_{\text{DPO}} = -\,\mathbb{E}_{(x,\,y_w,\,y_l)}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

The uncomfortable result (ScaleRL, Meta 2025, 400K GPU-hours): RL learning curves are predictable sigmoids, and most algorithm choices move efficiency, not the ceiling. Ceiling movers: model scale, generation length, batch size. Your moat is data, rewards, and infrastructure.

Williams (1992) · Schulman et al. (2017) · Shao et al. (2024) · Rafailov et al. (2023) · Khatri et al., "The Art of Scaling RL Compute" (2025)

Agenda

- 1. Motivation and Evaluation: what do we want?
- 2. Mid-training, SFT and data
- 3. Alignment, RLHF and DPO
- **4. RL for reasoning**
- 5. RL for agents and its infrastructure
- 6. Distillation
- 7. Putting it all together: Nemotron-3.5-Super
- 8. Post-training open problems
- 9. Q&A



Reasoning models: more compute at test time => better accuracy

”Think” before answering

CoT · 2022
“let’s think step by step” — prompting only

Verify · 2023
process vs outcome supervision; verifiers beat raw sampling

o1 · 2024
RL teaches the model to think before answering

R1 · 2025
the open recipe: GRPO + verifiable rewards, $294K

2025–26
thinking is default; IMO & IOI gold; agentic reasoning

R1-Zero, pure RL from a base model: AIME pass@1 15.6% → 71.0% — no SFT, no human reasoning demonstrations.

DeepSeek-R1 (2025; Nature 2025) · ARC Prize (Dec 2024) · Muennighoff et al., s1 (2025)

RLVR: the recipe

Groups of rollouts, a verifier, and ruthless data curation

Prompts
problems with checkable answers: math, code, logic

Sample G
16–64 rollouts per prompt from the current policy

Verify
exact match, unit tests, compilers — binary or graded

Advantage
group-relative: beat your own siblings

Update
policy gradient on the KL leash

$$\hat{A}_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G)} \; i = 1 \ldots G$$ rollouts of the same prompt

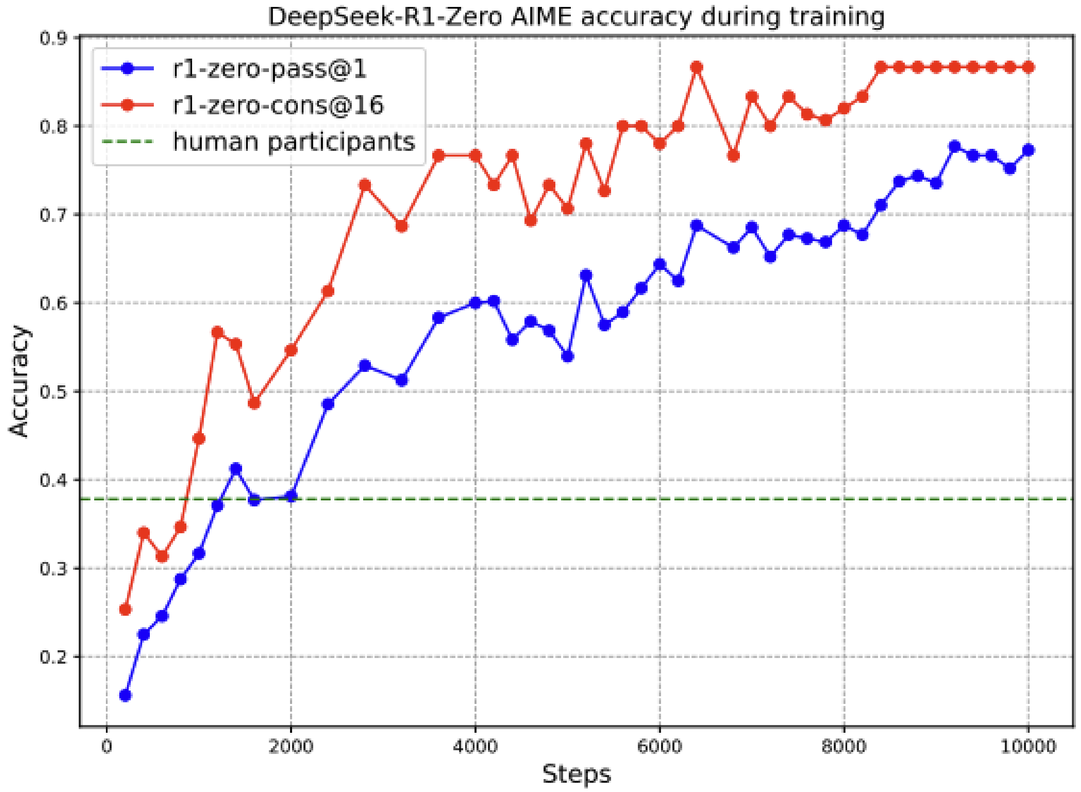
› **Reward profiling:** run every candidate prompt through the current model first; drop the ones it always solves and never solves — no gradient signal there (Nemotron's standard practice)

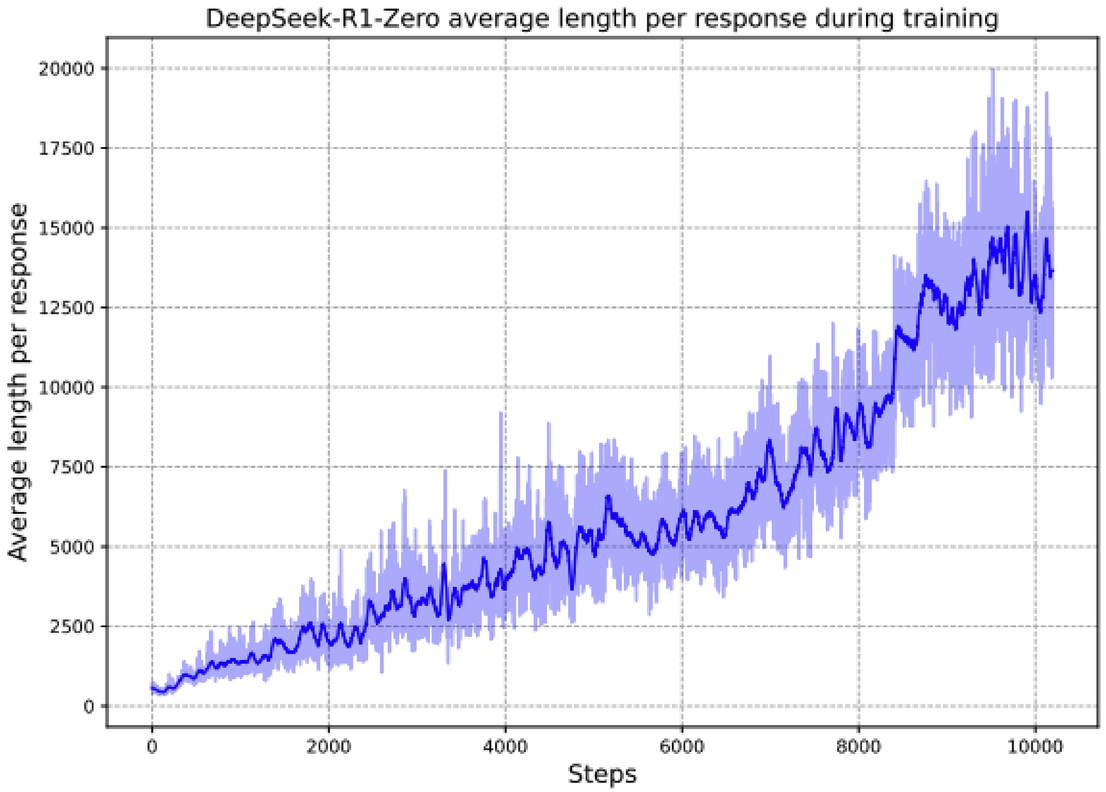
Shao et al., GRPO (2024) · Qwen3 (2025) · Magistral (2025) · NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026

NVIDIA

RL incentivizes reasoning

Length of generations grow together with accuracies





R1-Zero, pure RL from a base model: AIME pass@1 15.6% → 71.0% — no SFT, no human reasoning demonstrations.

DeepSeek-R1 (2025; Nature 2025)

What breaks — and the 2025–26 fixes

Entropy collapse, length inflation, format hacking

DAPO clip-higher + dynamic sampling + token-level loss + overlong penalty — *AIME 50 on Qwen2.5-32B in ~half the steps*

Dr. GRPO remove length & std normalization — they bias toward long wrong answers — *same accuracy, much shorter outputs*

GSPO sequence-level importance ratios instead of per-token — *stabilizes MoE RL (Qwen3) without routing tricks*

CISPO clip importance weights, not token updates — *keeps rare "Wait" pivot tokens learning; ~2× faster than DAPO*

ProRL (NVIDIA) KL leash + periodic reference reset, 2,000–3,000 steps — *keeps improving where short runs plateau*

VAPO the critic done right: value pretraining + adaptive GAE — *60.4 AIME — critics are expensive, not dead*

Zoom out (ScaleRL): these choices change how fast you climb, rarely how high. The ceiling responds to model scale, generation length, and batch size — so spend your cleverness on data, verifiers, and environments.

DAPO, Dr. GRPO, GSPO, CISPO, ProRL, VAPO (2025) · Khatri et al. (2025)

Does RL teach new skills, or sharpen old ones?

The elicitation debate — and its operational resolution

“It sharpens”

- pass@k crossover: the base model catches or passes the RLVR model at k ≈ 32–256 — every RL solution was already in the base’s support (Yue et al.)
- random rewards “work” on contaminated bases: +20 points on Qwen2.5-Math — elicitation, not learning
- one training example can move MATH-500 from 36 → 74

“It can teach”

- ProRL: after 2,000+ steps, pass@128 improves on tasks where the base scores zero at any k — biggest gains where the base is weakest
- grade the chain, not just the answer (CoT-Pass@k): RLVR wins at all k
- mechanism: ~20% of tokens — the high-entropy “forks” — carry nearly all the RL signal

Operational stance: short RL sharpens (pass@1 up, diversity down); prolonged, diverse, entropy-managed RL can genuinely extend — domain-dependently.

Yue et al. (2025) · Spurious Rewards (2025) · One-shot RLVR (2025) · ProRL (2025) · Wen et al. (2025) · “Beyond the 80/20 Rule” (2025)

NVIDIA

Reasoning is now a product dial

You pay for thinking in tokens and latency — efficiency became part of alignment

The dials

- **Every API grew one:** reasoning effort levels (OpenAI), thinking budgets (Gemini), think / no-think switches (Qwen3), extended thinking (Claude)
- **Test-time knobs:** suppress or extend the think block at inference (s1's "Wait"); parallel samples + selection (o3 on ARC, K2 heavy mode, Nemotron's verifier — section 7)
- **Token economics:** Kimi k1.5's "long2short"; overthinking on easy prompts is a measured tax, not a vibe

Trained in, not bolted on (Nemotron 3.5)

- three modes: reasoning-off, regular, medium-effort
- built in SFT by truncating reasoning traces to random budgets (with the closing think tag masked from the loss), then optimized in RL: ~4% of RL prompts run in medium mode with length-adjusted rewards
- result: medium mode spends 2.2× fewer tokens for ~9% lower accuracy across the Artificial Analysis index — a knob customers choose

Kimi k1.5 (2025) · s1 (2025) · NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026

Agenda

- 1. Motivation and Evaluation: what do we want?
- 2. Mid-training, SFT and data
- 3. Alignment, RLHF and DPO
- 4. RL for reasoning
- **5. RL for agents and its infrastructure**
- 6. Distillation
- 7. Putting it all together: Nemotron-3.5-Super
- 8. Post-training open problems
- 9. Q&A



Agents: the hardest post-training target

Models now interact with environments via tools to achieve goals

Agent: “Given a goal + context → decide and take action(s) → observe outcomes → update plan → repeat → finish.”

What changes when the model acts

- **Horizon:** trajectories of 100+ tool calls and 100K+ tokens; rewards are sparse and mostly terminal
- **Latency:** shell tools, compilers, browsers on CPU; simulated users are another LLM on GPU — the environment is now inside your training loop
- **Credit:** which early decision doomed the episode? Credit assignment across turns is unsolved in general

Environments are the new datasets

Whoever builds them owns the capability

Environment = (tasks, tools, verifier)

Software engineering

real repos + unit tests; SWE-Gym, R2E-Gym; Meta's SWE-RL rewarded patch similarity on real PRs

Terminal / computer use

containerized tasks (Terminal-Bench + Harbor are built for RL rollouts, not just eval)

Search & browsing

multi-hop research tasks; BrowseComp-style; Deep Research trained on them end-to-end

Simulated users

tau-style conversational tool use — a user-LLM plays the customer, turn-level judges score

Finance agent

SEC-filing search, parsing, evidence retrieval — same tools in SFT, RL, and eval (Nemotron 3.5)

Competitive programming

2,847 Olympiad problems with executable verifiers, binary compile-and-test rewards (3.5 RL)

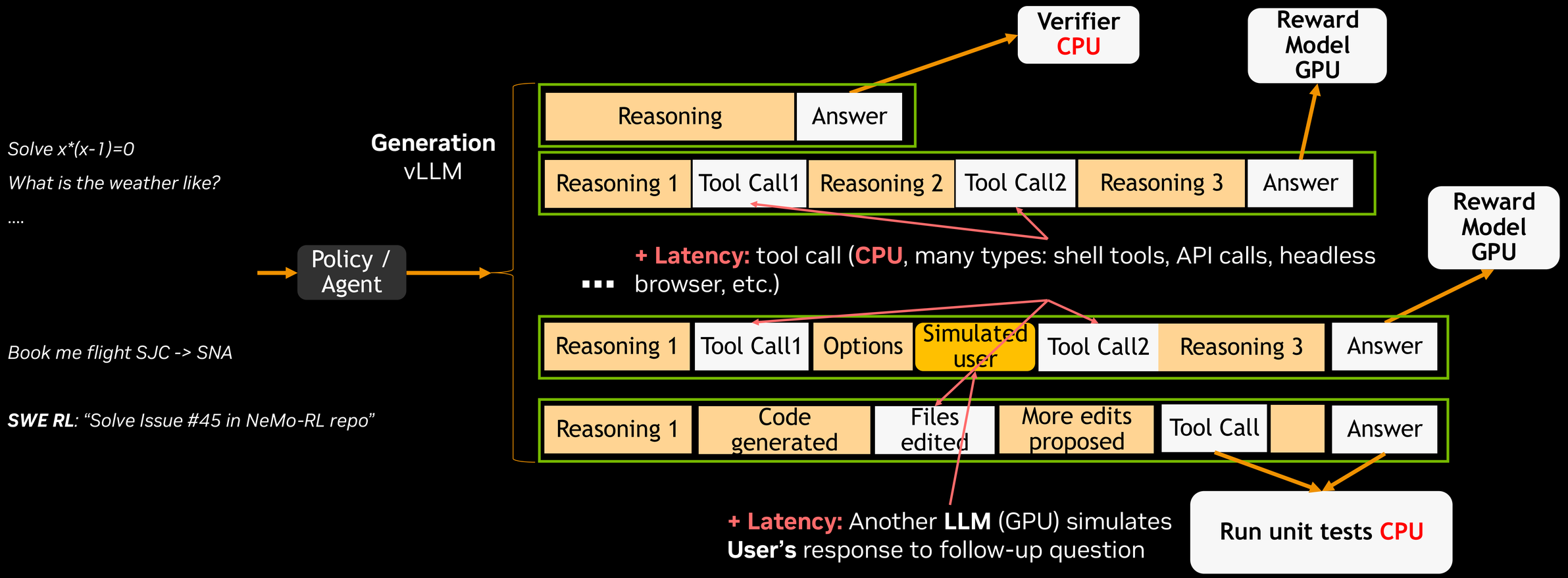
› **Warning:** training environments and benchmarks are converging — contamination by construction. Hold out harnesses and decontaminate tasks.

SWE-RL (Meta, 2025) · Qwen3-Coder (2025) · Kimi K2 (2025) · NeMo Gym: github.com/NVIDIA-NeMo/Gym · NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026

NVIDIA

Multi-environment RL

Train everything at once: diverse batches resist reward hacking and regressions



Every environment instance runs in its singularity container

Generation – it is a list of sequences

Training – is a tensor sequence packing and minimal padding

Training–inference mismatch makes RL off-policy

Rollouts come from the sampler (q); gradients come from the trainer (p) — and in practice q ≠ p



Where the mismatch comes from

- **Numerics:** different kernels, fused ops, reduction order — outputs differ even at the same precision
- **Quantization:** FP8 / INT8 rollouts make generation cheap — but push the sampler away from the trainer
- **Staleness:** asynchronous, disaggregated RL trains on rollouts from older checkpoints
- **MoE:** trainer and sampler can even disagree on expert routing

Same model on paper — different kernels, precision, timing: **q ≠ p**

```
floating point is not associative — in fp4_e2m1:   (0.5 + 0.5) + 2 = 3   but   0.5 + (0.5 + 2) = 2
```

The puzzle: SFT trains happily on data from a different model. RL can collapse from tiny numerical differences between two copies of the same model. Why?

In production: Nemotron 3.5 replays MoE routing decisions and keeps the LM head in FP32 — together cutting train–inference KL error by more than 10×.

Marek & Ryabinin (Together AI), “Score Centering Stabilizes Off-policy Reinforcement Learning”, arXiv:2609.20807, 2026

Training-inference mismatch

Various ways to correct

$$\nabla_\theta \mathbb{E}_{p_\theta}[R] = \mathbb{E}_{p_\theta}\left[R \underbrace{\nabla_\theta \log p_\theta(y)}_{\text{“score”}}\right].$$

Importance sampling corrects for the mismatch

$$\nabla_\theta \mathbb{E}_{p_\theta}[R] = \mathbb{E}_{p_\theta}\left[R \nabla_\theta \log p_\theta(y)\right] = \mathbb{E}_{q_\theta}\left[\frac{p_\theta(y)}{q_\theta(y)} R \nabla_\theta \log p_\theta(y)\right]$$

But IS works on token-level basis, in practice, rare tokens can have very large correction => need to do clipping => introduces bias

Table 1: IS-based correction methods represented as a grid, with implementation notes in the last column. Level determines whether the statistic is computed per token or per sequence. Outside specifies what happens beyond the Low . . . High region. Sign specifies which advantage signs are corrected: both applies the weight to every token; neg only to tokens with negative advantage (positive-advantage tokens keep weight 1); and ppo applies the PPO pessimistic clip, i.e. the minimum of the weighted and clipped objectives, so the clip only binds in the direction that would increase the objective. Dual clipping follows verl's PPO implementation: negative-advantage contributions with ratio above 3 are dropped (whole responses for GSPO). [P] denotes the paper default and [V] the verl default (Sheng et al., 2024).

Method	Level	Stat	Low	High	Outside	Sign	Implementation notes
Naive IS (Williams, 1992)	token	ratio	0	∞	-	both	—
TIS[V] (Yao et al., 2025) CISPO (MiniMax, 2025)	token	ratio	0	2.0	clip	both	CISPO's dynamic sampling and length penalty omitted.
MIS (IcePop)[P] (Ling Team, 2025)	token	ratio	0.5	5.0	drop	both	Separate PPO surrogate between policy versions omitted.
PPO[P,V] (Schulman et al., 2017)	token	ratio	0.8	1.2	drop	ppo	Dual clipping at 3 (verl default).
DAPO[P,V] (Yu et al., 2025)	token	ratio	0.8	1.28	drop	ppo	Dual clipping at 3; dynamic sampling and overlong reward penalty omitted.
GSPO[P,V] (Zheng et al., 2025)	seq	ratio	0.9997	1.0004	drop	ppo	Token gradients summed rather than response-averaged; added dual clipping at 3.
TOPR[P] (Le Roux et al., 2025)	seq	ratio	0	1.0	clip	neg	—
DPPO[P] (Qi et al., 2026)	token	tv	0	0.2	drop	ppo	No importance-ratio clipping.

Marek & Ryabinin (Together AI), “Score Centering Stabilizes Off-policy Reinforcement Learning”, arXiv:2609.20807, 2026

NVIDIA

Drift: mismatch turns RL into self-distillation

Decompose the expected policy-gradient update at any prefix (covariance identity)

$$\mathbb{E}_q[R\, s_{y_t}] = \underbrace{\mathbb{E}_q[R]\ \bar{s}}_{\text{"drift"}} + \underbrace{\mathrm{Cov}_q(R,\ s_{y_t})}_{\text{"signal"}}$$

$s_{y_t} = \nabla_\theta \log p_\theta(y_t \mid y_{<t})$ — score of the sampled token $\bar{s} = \mathbb{E}_q[s_{y_t}] = \sum_v q_v\, s_v$ — its expectation under the sampler on-policy: $\bar{s} = 0$

Drift is a pure mismatch artifact

It depends on rewards only through their mean — **it knows nothing about which rollouts succeeded**.

Its direction is the SFT gradient toward the sampler: under mismatch, policy gradient quietly distills from sampler into the trainer at every prefix.

...and it compounds into collapse

The sampler is a biased copy of the trainer, re-synced after every update. The trainer chases the sampler's errors; the sampler inherits them back — a feedback loop.

Distilling a fixed teacher is harmless. Distilling a moving copy of yourself is not.

Group centering does not remove it

GRPO advantages sum to zero over the group — but drift lives at prefixes: positive expected advantage after promising prefixes, negative after mistakes.

Drift sits exactly where the learning signal is. Shrunk, not removed.

Marek & Ryabinin (Together AI), "Score Centering Stabilizes Off-policy Reinforcement Learning", arXiv:2609.20807, 2026

NVIDIA

Score centering: cancel the drift, keep the signal

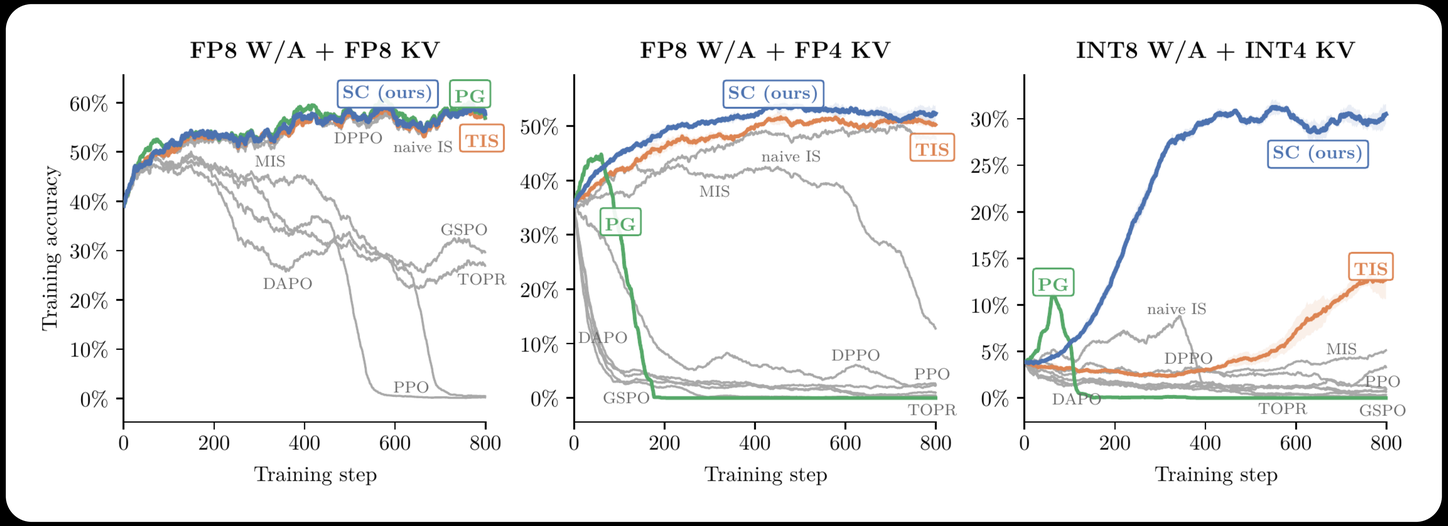
An additive, deterministic correction — no importance ratios, no clipping, no hyperparameters

$$\underbrace{\tilde{s}_{y_t} = s_{y_t} - \bar{s}}_{\text{subtract expected score}} \Longrightarrow \mathbb{E}_q[R\,\tilde{s}_{y_t}] = \underbrace{\mathrm{Cov}_q(R,\, s_{y_t})}_{\text{the on-policy update, sampled under q}}$$

The centered score has zero mean under the sampler, so drift cancels exactly at every prefix — even off-policy. When q = p the correction is zero: a no-op on-policy.

Why it is practical

- needs only the sampler's top-k logprobs (k = 32–128); the trainer models the tail → < 1% wall-clock overhead
- a few-line scalar loss with stop-gradients — drop-in for any policy-gradient stack
- additive and deterministic — importance ratios are multiplicative and random, so they get clipped, which re-introduces drift
- can be combined with importance sampling



Qwen3-30B-A3B · RL on math · quantized sampler

- **FP8 everywhere** — mild mismatch: everything is stable, even uncorrected PG (58%)
- **FP4 KV cache** — PG collapses in < 200 steps; SC 52% ≈ TIS 51%
- **INT8 W/A + INT4 KV** — severe: SC 30%, TIS 12%, everything else < 5%
- **Severe staleness** — compose them: TIS + SC / MIS + SC perform best

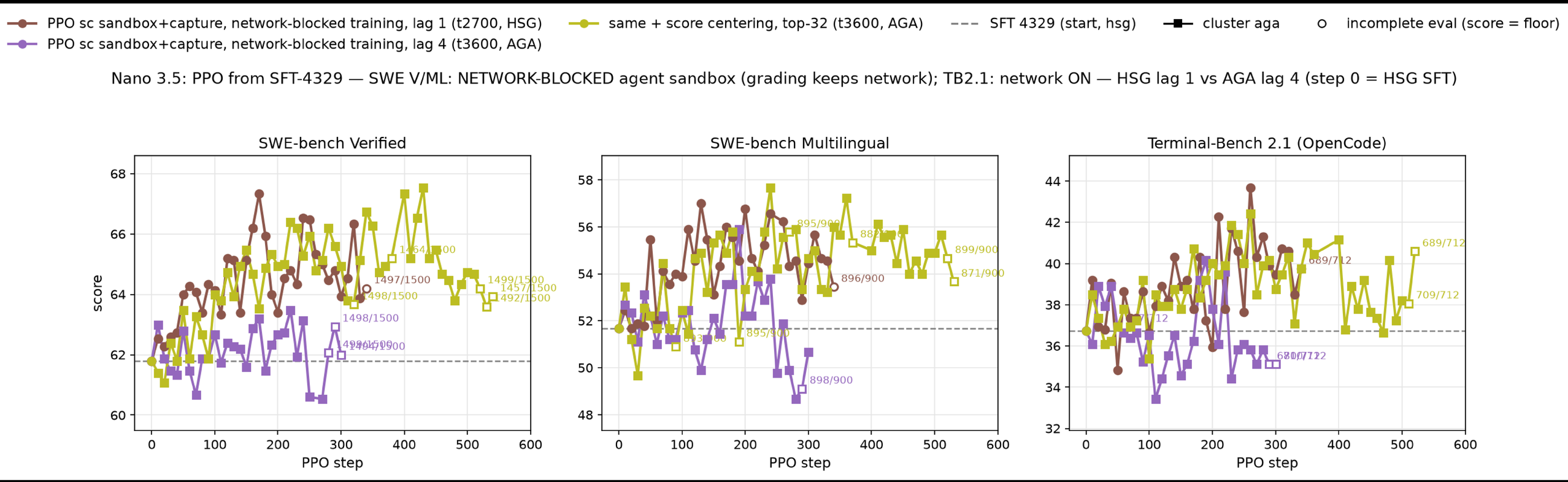
The worse the mismatch, the bigger the edge.

Marek & Ryabinin (Together AI), "Score Centering Stabilizes Off-policy Reinforcement Learning", arXiv:2609.20807, 2026

NVIDIA

Score centering

Better lag better GPU utilization



Agenda

- 1. Motivation and Evaluation: what do we want?
- 2. Mid-training, SFT and data
- 3. Alignment, RLHF and DPO
- 4. RL for reasoning
- 5. RL for agents and its infrastructure
- **6. Distillation**
- 7. Putting it all together: Nemotron-3.5-Super
- 8. Post-training open problems
- 9. Q&A



Distillation: capability flows downhill

Three main types

Sequence-level

$$\mathbb{E}_{y \sim \pi_T}[-\log \pi_S(y)]$$

Fine-tune the student on teacher outputs — every modern SFT set is this.

R1's 800K samples: R1-Distill-Qwen-32B hits 72.6 AIME vs 47.0 for direct RL on the same base — for small models, distillation beats RL.

Logit-level

$$\mathbb{E}_{x \sim \mathcal{D}}\, \mathrm{KL}[\pi_T(\cdot|x) \,\|\, \pi_S(\cdot|x)]$$

Match full next-token distributions on a fixed corpus — dense signal, any student architecture.

Gemma 3 distills every size (up to 14T tokens, 256 sampled logits per token); Minitron: prune + distill at up to 40× fewer tokens than from-scratch.

On-policy

$$\mathbb{E}_{y \sim \pi_S}\, \mathrm{KL}[\pi_S(\cdot|c_t) \,\|\, \pi_T(\cdot|c_t)]$$

Student samples; teacher grades every token (reverse KL) — dense reward, no verifier, trains on the student's own states.

Fixes exactly the off-distribution failure of SFT — and it is the engine of section 7's MOPD.

Ask of every method: whose samples, which distribution, which direction of KL — the three choices that define the whole design space.

DeepSeek-R1 (2025) · Gemma 3 (2025) · Minitron (2024) · GKD (2024)

Logit distillation at production scale

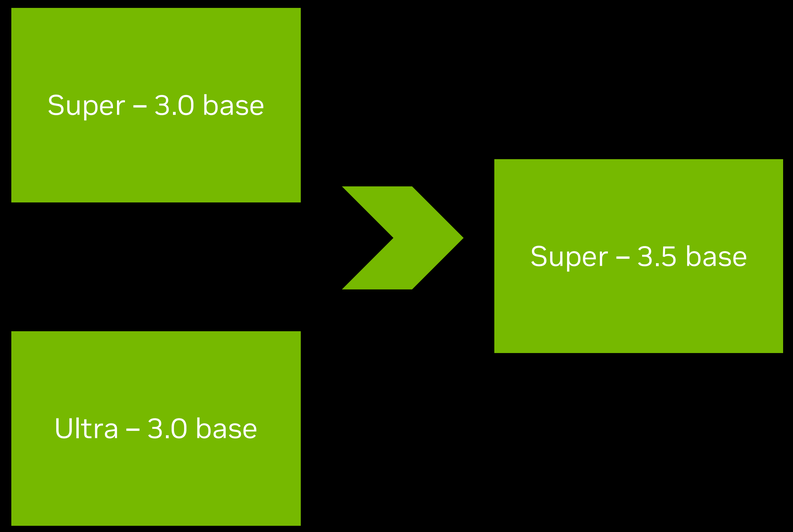
How the Nemotron 3.5 base models were born — 1T tokens each

The recipe

- **Teacher chain:** Lightning 3.5 Base ← Super 3 Base; Super 3.5 Base ← Ultra 3 Base — always the next-larger teacher, 1T tokens each
- **Top-k logits:** teacher's top-128 logits + a sink token carrying the residual tail mass — storage- and bandwidth-friendly
- **Losses:** 0.9 forward-KL + 0.1 cross-entropy (Lightning); skew reverse KL, alpha = 0.1 (Super)
- **LR trick:** start at the student's minimum LR, ramp slowly to 10% of its peak, decay — limits early drift from a good initialization

Why it wins

- a strong base for a fraction of from-scratch pretraining cost — capability inherited, not re-discovered
- architecture freedom: the student can be pruned, reshaped, hybridized (Minitron line: +16% MMLU vs training the same small model from scratch)
- resulting bases: Super 3.5 84.1 MMLU / Lightning 78.6 — before any post-training
- the same top-k-logits interface inference engines expose is what score centering uses — infrastructure converges



NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026 · Minitron (NVIDIA, 2024)

On-policy distillation: RL's cheaper sibling

Dense reward = teacher log-probs on the student's own tokens

Mechanics

- student generates; teacher scores every token; minimize reverse KL on the student's own states (GKD lineage)
- RL hands you one sparse scalar per episode; OPD hands you a signal at every token — O(N) bits instead of O(1)
- no verifier needed — the teacher is the reward model, dense and always on-distribution
- caveat (2026): a stronger teacher is not automatically a better teacher — OPD transfers thinking patterns, not raw capability

Economics (the headline numbers)

~10× cheaper

Qwen3: strong-to-weak distillation reached 74.4 AIME at 1,800 GPU-hours — direct RL got 67.6 at 17,920

9–30×

compute reduction vs continued SFT; 50–100× vs RL when a teacher policy already exists (Thinking Machines, Oct 2025)

85 → 45 → 83

IFEval through mid-training damage and OPD repair — restore behavior without losing the new knowledge

$$\mathbb{E}_{y \sim \pi_S} \, \mathrm{KL}\big[\pi_S(\cdot|c_t) \,\|\, \pi_T(\cdot|c_t)\big]$$

GKD (2024) · Qwen3 (2025) · Thinking Machines, "On-Policy Distillation" (Oct 2025)

NVIDIA

Agenda

- 1. Motivation and Evaluation: what do we want?
- 2. Mid-training, SFT and data
- 3. Alignment, RLHF and DPO
- 4. RL for reasoning
- 5. RL for agents and its infrastructure
- 6. Distillation
- **7. Putting it all together: Nemotron-3.5-Super**
- 8. Post-training open problems
- 9. Q&A



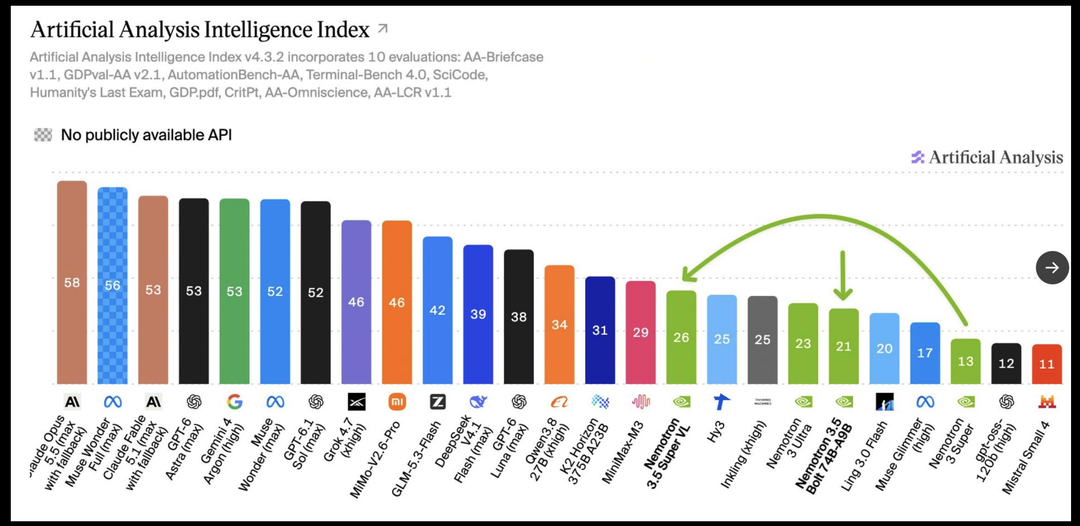
Nemotron 3.5: open, efficient, agentic

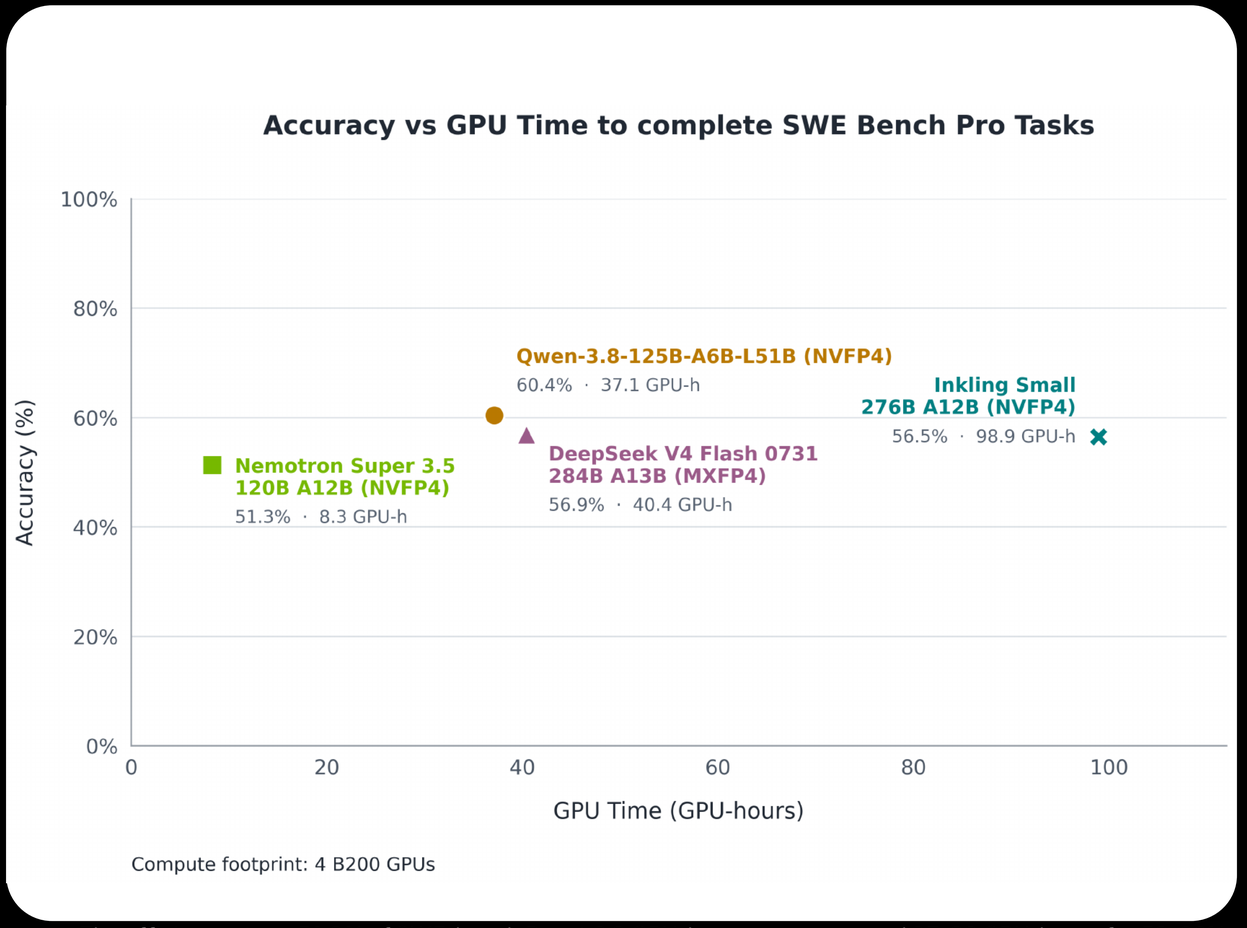
Super-3.5 120B-A12B (vision-language, 1M context) + Bolt-3.5 all released

51.78%
SWE-bench Pro — surpassing Nemotron 3 Ultra (550B-A55B) with ~4.6× fewer active parameters

60.7 / 89.5 / 88.2
Terminal-Bench 2.1 / GPQA-Diamond / LiveCodeBench v6

1420 ± 12
Text Arena Elo with style control — chat quality from an agent-first model



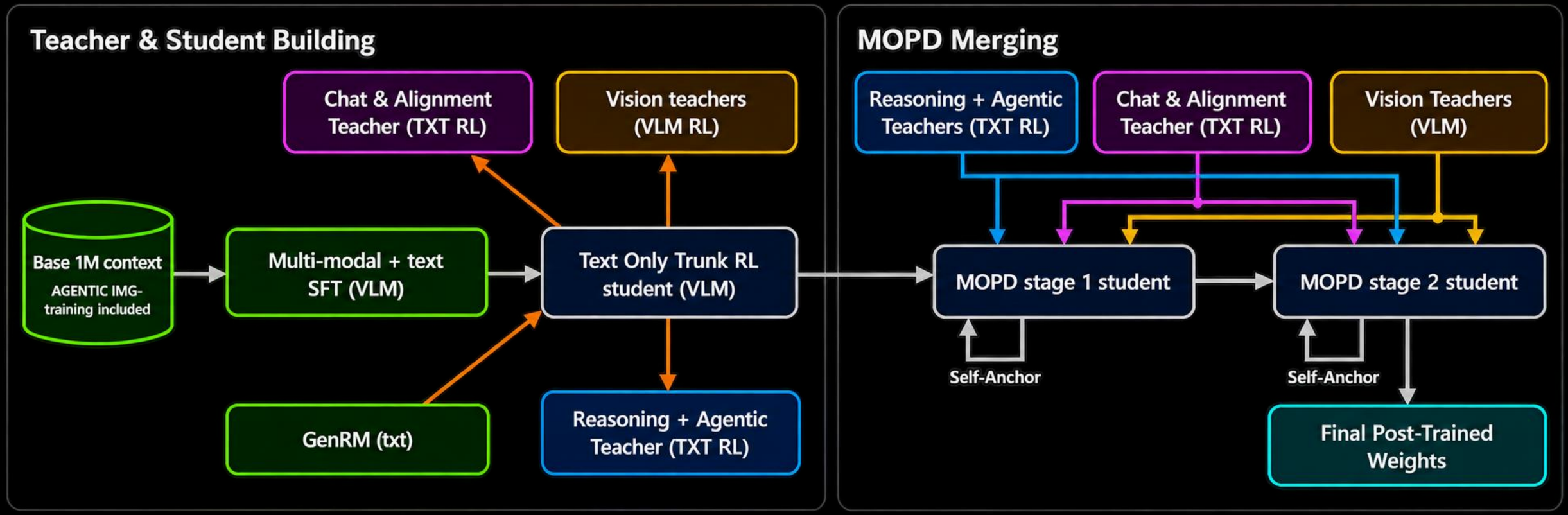


The efficiency story: 51.3% of SWE-bench Pro at 8.3 GPU-hours — peers spend 37–99 GPU-hours for 56–60%.

NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026 · hybrid Mamba-Transformer latent MoE · native MTP speculative decoding · NVFP4 release

Nemotron-3.5-Super-VL

The post-training pipeline



- two-stage SFT turns the text-only base into a VLM while adding reasoning, chat, agentic, and long-context skills
- one RL stage runs RLVR and RLHF jointly — each sample routed to its verifier or reward model; specialist teachers then branch from the RL checkpoint
- MOPD merges the teachers back into one student; a frozen-backbone MTP boosting pass finishes the model; Lightning takes the light path: text SFT + RLVR only

NVIDIA, “Nemotron 3.5 Super and Lightning” technical report, October 2026

Specialist teachers, then merge

Push each capability as far as it goes — in a model that doesn’t have to be good at anything else

Agentic teacher

end-to-end GRPO on fully executable Terminal, SWE, and Search sandboxes — no single-step environments; 64 prompts × 16 rollouts, 2-hour timeouts, trajectory age ≤ 1

Reasoning teacher

25,600-example curriculum (90% science / 5% math / 5% code), generation capped at 131,072 tokens; answer verification by an LLM judge for math & STEM, execution for code; teacher taken after 80 updates

Alignment teacher

RLHF with the principle-following GenRM: responses compared to a fixed strong reference instead of each other; mean of per-rubric scores; reasoning-length penalty; abstention training

Vision teachers

video reasoning and tracking, vision tool use, vision RLVR — all on deterministic verifiable rewards; a malformed think block gets its advantage overridden to -5, and concise correct reasoning keeps a small (at most 5%) bonus

Why specialists: capabilities with conflicting data diets and inference profiles stop fighting over one model’s capacity — the merge comes next.

NVIDIA, “Nemotron 3.5 Super and Lightning” technical report, October 2026

NVIDIA

MOPD: merging the teachers without forgetting

Multi-teacher on-policy distillation — reverse KL on the student's own rollouts

$$\hat{A}_t = \mathrm{sg}[\log \pi_T(y_t|c_t) - \log \bar{\pi}_S(y_t|c_t)]$$ dense, per-token, verifier-free

Mechanics

- every student token gets a dense advantage from one teacher forward pass — no verifier, no teacher sampling; zero where they already agree
- async with one-step-stale rollouts: importance-weighted and masked outside [0.2, 5]
- domains without a specialist route to a self-anchor — a frozen copy of the starting student that regularizes departures

Two stages on 128 GB200 nodes

- stage 1 (30 steps): agentic 35% · reasoning 25% · alignment 20% · vision tools 15% · self-anchor 5% — the fast-improving domains first
- stage 2 (40 steps): chat + vision domains that move slowly; agentic & reasoning teachers retire into the self-anchor
- trust-region teacher in stage 2: mix alpha = 0.2 of teacher with the frozen student — bounds the negative advantage at log(0.8) ≈ −0.22 and stabilizes training once the teachers are absorbed

Why staged: benchmarks improve at different rates under distillation — some peak within a few steps and then decline. Merge order is a real hyperparameter.

NVIDIA, "Nemotron 3.5 Super and Lightning" technical report, October 2026 · teachers served on independent TP8/EP8 inference fleets, pipelined in NeMo-RL

NVIDIA

Ship it: efficiency is post-training too

The last mile — speculative decoding, 4-bit quantization, and test-time scaling

MTP boosting

A frozen-backbone pass trains only the speculative-decoding head: hidden-state mixing + a distillation loss against the backbone.

Draft acceptance length 3.79 → 4.61 (temperature 1); the largest gains at the deepest draft positions — where train–inference mismatch is worst. TIM, one last time.

NVFP4 + QAD

4-bit weights with quantization-aware distillation — 200 iterations of SFT-based recovery.

99.62% median accuracy recovery across 36 benchmarks; 2.43× the throughput of BF16; up to 4,856 tok/s/GPU on GB200 while holding ≥ 50 tok/s per user.

Test-time scaling

N parallel agents attempt the task; the same checkpoint then judges candidate trajectories pairwise in a pivot tournament (LLM-as-a-Verifier).

Terminal-Bench 62.7 → 71.9 · DeepSWE 24.6 → 34.5 · BrowseComp 71.5 → 79.8 — with zero parameter updates.

Everything is open: checkpoints (base / BF16 / NVFP4), the post-training datasets (Nemotron-Post-Training-Dataset-v2, RL-Super-Training-Blends on Hugging Face), NeMo-RL + NeMo Gym, and a step-by-step reproducibility tutorial.

Takeaways

Three things to remember

1

Post-training turns compute into behavior

demonstrations → preferences → verifiable rewards → environments — with distillation moving the result anywhere, for 10–100× less than you'd re-learn it.

2

Data, rewards, and evals are the moat — algorithms mostly set the pace

curate hard prompts, engineer un-hackable rewards, decontaminate everything; every eval becomes a reward eventually, and every reward gets attacked.

3

Infrastructure is strategy

async multi-environment RL, train–inference numerics, distillation economics, and serving efficiency decide what is trainable at all — the lecture's deepest bugs lived in kernels, not in loss functions.

Materials: Nemotron 3.5 report · huggingface.co/nvidia (datasets & weights) · github.com/NVIDIA-NeMo/RL & /Gym · kuchaev.com · x.com/kuchaev

NVIDIA

Agenda

- 1. Motivation and Evaluation: what do we want?
- 2. Mid-training, SFT and data
- 3. Alignment, RLHF and DPO
- 4. RL for reasoning
- 5. RL for agents and its infrastructure
- 6. Distillation
- 7. Putting it all together: Nemotron-3.5-Super
- **8. Post-training open problems**
- 9. Q&A



Open problems

There is plenty to research

Accuracy & efficiency tradeoff

More reasoning yield better answers. But how do we make models reason efficiently?
How does verbosity relate to model size? Loops?

How to make good base model for post-training team?

What makes base model good starting point for post-training?

How base model improvements translate to post-training improvements?

How to make a model great for further customization?

How entropy and model “feel” related? What makes it easy or hard to further post-training the model on a new use-case?

Reward specification beyond verifiers

rubrics and GenRMs still get gamed; scalable oversight for tasks no single human can check remains open

Long-horizon credit assignment

which of 300 tool calls mattered? turn-level advantages, counterfactual credit, single-rollout methods — all early

Does RL create capability?

pass@k-robust metrics, entropy mechanics, contamination-proof protocols — the elicit-vs-teach question wants rigor

Continual post-training

every stage regresses something (RULER 79.9 → 33.5); merging, replay, and interference lack a usable theory

Off-policy RL at scale

staleness, quantized rollouts, TIM-robust estimators — score centering is one datapoint, not a solved field

SFT vs Mid-training?

Do we need SFT at all? Can we push all that data into mid-training?

What you can do without 10,000 GPUs

The bottlenecks are environments, data, and evals

- **Build environments and verifiers:** the single highest-leverage artifact: a clean task + tools + verifier package. Contribute to NeMo Gym or the Environments Hub — community environments already feed real training runs.
- **Small-model RL science:** the score-centering paper ran ~6,180 H100-hours total; s1 fine-tuned in 26 minutes on 16 H100s; one-shot RLVR needs a single example. Rigorous 0.6B–8B studies change what the frontier does.
- **Data-centric work:** difficulty estimation, reward profiling, curriculum design, decontamination tooling — data levers transfer to any scale.
- **Analyze open checkpoints:** Nemotron ships weights and the post-training datasets; OLMo and Tülu are fully open — interpret what RL actually changed in the weights and the entropy.
- **Build evals:** expert-written, contamination-resistant, reliability-aware — a good benchmark steers the entire field for two years.

github.com/NVIDIA-NeMo/Gym · huggingface.co/nvidia · Environments Hub (Prime Intellect) · Ai2 OLMo / Tülu

NVIDIA

Super-3.5-VL

Bolt-3.5

NeMo-RL

NeMo-Gym



NVIDIA