EECE 576L

Advanced Topics in Deep Learning

Lecture 5: Large Language Models

Renjie Liao

University of British Columbia

2026/27 Winter Session, Term 1 (Fall 2026)

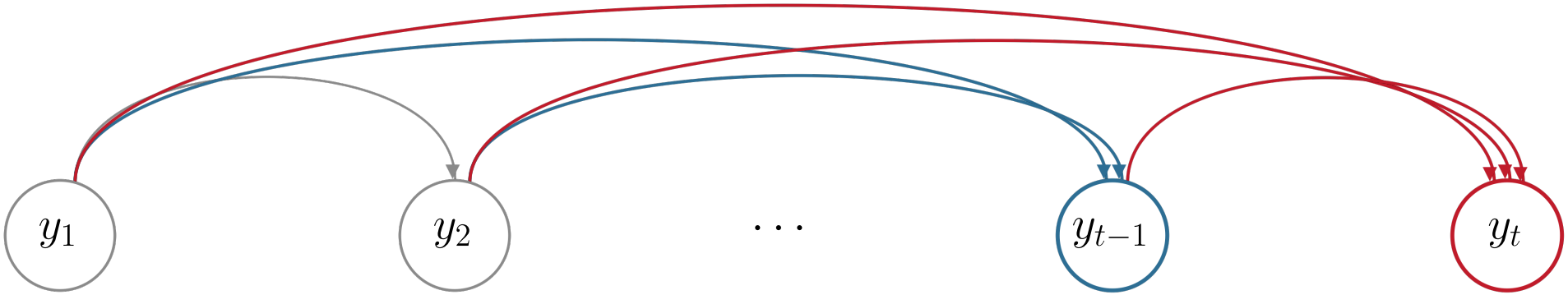
Outline

- **Pre-training**
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- Post-training
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- Inference
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

Autoregressive Language Modeling

One decoder predicts each token from its preceding prefix.

$$\pi_\theta(y) = \prod_{t=1}^{T} \pi_\theta(y_t \mid y_{<t})$$



Each arrow permits conditioning on an earlier token.

Intermediate tokens and their incident edges are omitted.

What Is a Token?

A token is a discrete vocabulary entry, represented by an integer ID.
A tokenizer converts text into these entries.

Split on spaces

Let’s	do	tokenization!

Split on punctuation

Let	’s	do	tokenization	!

Subwords can split a word. Here </w> marks a word ending.

Let’s </w>	do</w>	token	ization</w>	!</w>

[1] Hugging Face LLM Course.

BPE: Learning Merge Priorities

Training corpus: repeated words, with no merges across word boundaries.

$$V_0 = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\}$$

Chosen seed alphabet includes e and f for unseen inputs.

	“cd”: 300 copies	**“ab”: 100 copies**	**“abc”: 100 copies**
Seed	[c, d]	[a, b]	[a, b, c]
Rank 1 300 pairs	[cd]	[a, b]	[a, b, c]
Rank 2 200 pairs	[cd]	[ab]	[ab, c]
Rank 3 100 pairs	[cd]	[ab]	[abc]

Save merge order as rank. Recount pairs after every training merge.

[2] Sennrich et al., ACL 2016; [3] OpenAI, tiktoken educational implementation.

BPE: Vocabulary and Encoding

The vocabulary keeps base units and every learned merge product.

Vocabulary

$$V_0 = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{e}, \texttt{f}\}$$

$$V_1 = V_0 \cup \{\texttt{cd}\}$$

$$V_2 = V_1 \cup \{\texttt{ab}\}$$

$$V_3 = V_2 \cup \{\texttt{abc}\}, \quad |V_3| = 9$$

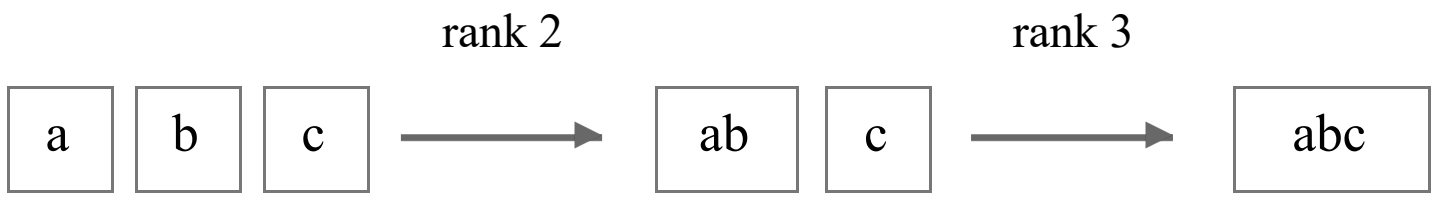
Saved merge ranks

$$1: \quad (\texttt{c}, \texttt{d}) \mapsto \texttt{cd}$$

$$2: \quad (\texttt{a}, \texttt{b}) \mapsto \texttt{ab}$$

$$3: \quad (\texttt{ab}, \texttt{c}) \mapsto \texttt{abc}$$

Encoding “abc”



Rank 1 has no match here. Apply rank 2, then the newly eligible rank 3.

[2] Sennrich et al., ACL 2016; [4] Hugging Face LLM Course, §6.5; [3] OpenAI, tiktoken educational implementation.

BPE Encoding: A New Input

New input: “abcdef”, treated as one pre-tokenized segment.

Smaller rank means higher priority. New-input pair counts do not set priority.

$$1 : (\texttt{c}, \texttt{d}) \mapsto \texttt{cd} \quad 2 : (\texttt{a}, \texttt{b}) \mapsto \texttt{ab} \quad 3 : (\texttt{ab}, \texttt{c}) \mapsto \texttt{abc}$$

Seed	[a, b, c, d, e, f]	Eligible: ab (rank 2), cd (rank 1).
Rank 1	[a, b, cd, e, f]	Next: ab (rank 2) is eligible.
Rank 2	[ab, cd, e, f]	No learned pair remains.

Stop: no remaining adjacent pair has a learned merge rule.

abc is in the vocabulary, but c has already joined cd.

Longest-prefix matching would instead give [abc, d, e, f].

[3] OpenAI, tiktoken educational implementation.

BPE Training Algorithm

Training learns a vocabulary and fixed merge priorities from corpus counts.

Input: segmented corpus, seed alphabet, maximum number of merges.

$$C, \qquad V \leftarrow V_0, \qquad \mathcal{R} \leftarrow [\,], \qquad M \in \mathbb{N}$$

For each training iteration: $j = 1, \ldots, M$

1. Count eligible adjacent pairs. Stop if none remain; otherwise select a maximum.

$$(a, b) \leftarrow \arg\max_{(u,v)} c_C(u, v)$$

2. Replace all nonoverlapping occurrences in the current corpus, left to right.
3. Add the new piece and record this pair with the next rank.

$$V \leftarrow V \cup \{ab\}, \qquad r(a, b) \leftarrow j, \qquad \mathcal{R}.\,\mathrm{append}((a, b))$$

Return the vocabulary and ranked rules. Freeze them before encoding new text.

[2] Sennrich et al., ACL 2016; [3] OpenAI, tiktoken educational implementation.

BPE Encoding and Decoding

Encoding reuses the learned ranks. Decoding reconstructs text from token IDs.

Input: a segment in seed units and the frozen merge-rank map.

$$S = (s_1, \ldots, s_n), \qquad E(S) = \{i : 1 \le i < n,\ (s_i, s_{i+1}) \in \mathcal{R}\}$$

1. Stop if no learned pair is eligible. Otherwise choose:

$$i^\star = \arg\min_{i \in E(S)} \big(r(s_i, s_{i+1}),\ i\big)$$

Lowest rank wins; equal-rank ties use the leftmost occurrence.

2. Merge this occurrence, recompute eligible pairs, and repeat step 1.
3. Map final pieces to their fixed vocabulary IDs.

Decoding (illustrative IDs) Look up pieces, then concatenate.

$$[7, 6, 4, 5] \mapsto [\texttt{ab}, \texttt{cd}, \texttt{e}, \texttt{f}] \mapsto \texttt{abcdef}$$

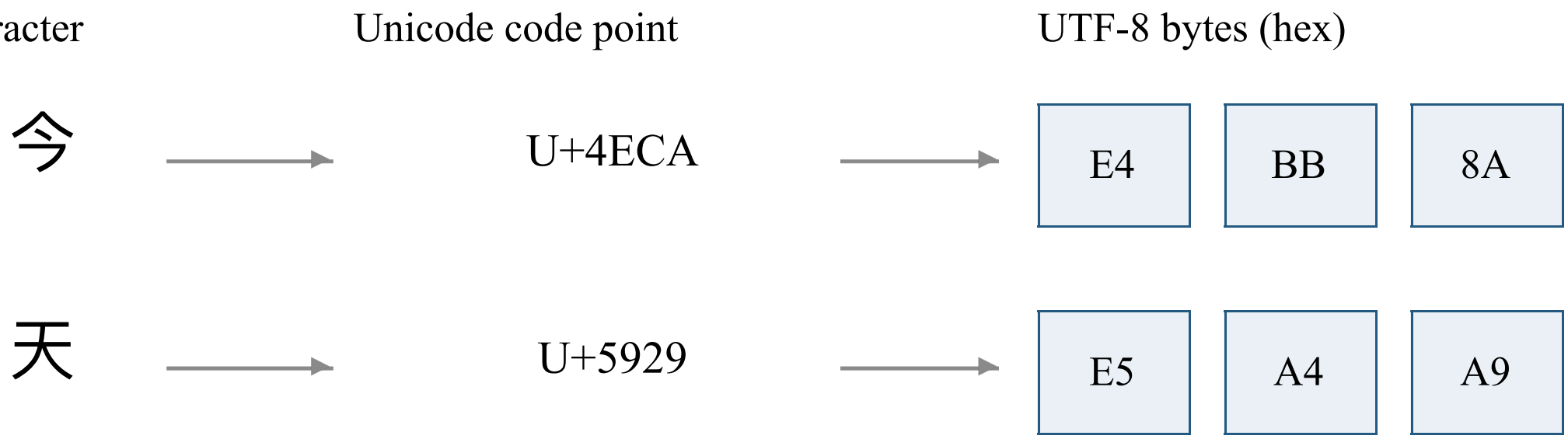
For byte-level tokenizers, concatenate token bytes before UTF-8 decoding.

[3] OpenAI, tiktoken educational implementation.

Unicode and UTF-8

Unicode gives characters code points. UTF-8 turns them into 1–4 bytes.

今天 “today” in Chinese

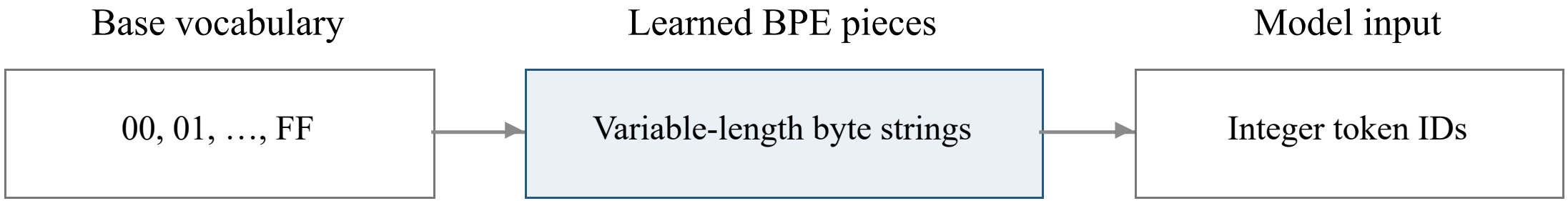


Two code points become six bytes; this does not determine the token count.

[5] Yergeau, RFC 3629, 2003, §3.

Byte-Level BPE: Coverage and Compression

Start with all 256 possible byte values (00–FF).
UTF-8 encodes Unicode text as sequences of these bytes.



- Coverage: any valid UTF-8 text remains representable.
- Compression: learned merges change token counts.
- A single byte need not be a valid UTF-8 character.

Representable does not mean the model understands the language.

Preprocessing can change the original text, e.g., normalization of café.

é: U+00E9 versus e + ◌́: U+0065 U+0301

[3] OpenAI, tiktoken educational implementation; [6] Unicode Standard Annex #15.

Tokenizer Families at a Glance

Vocabulary size is chosen per model, not fixed by the algorithm.

Family	Learning vocabulary	Encoding new text	Examples
BPE [2]	Merge most frequent adjacent pairs.	Follow saved merge priorities.	Qwen3 [7] GPT-4o [8]
WordPiece	Grow vocabulary using likelihood gains [9].	Greedy longest vocabulary match.	BERT [10]
Unigram	Fit piece probabilities; prune candidate pieces.	Most probable split (Viterbi), or sampling.	T5 [11]

SentencePiece [12] is a toolkit supporting BPE and Unigram.

[2] Sennrich et al., ACL 2016; [13] Kudo, ACL 2018; [12] Kudo & Richardson, EMNLP 2018.

Tokenizers in Closed Models

Public tokenizer details depend on the model and version.

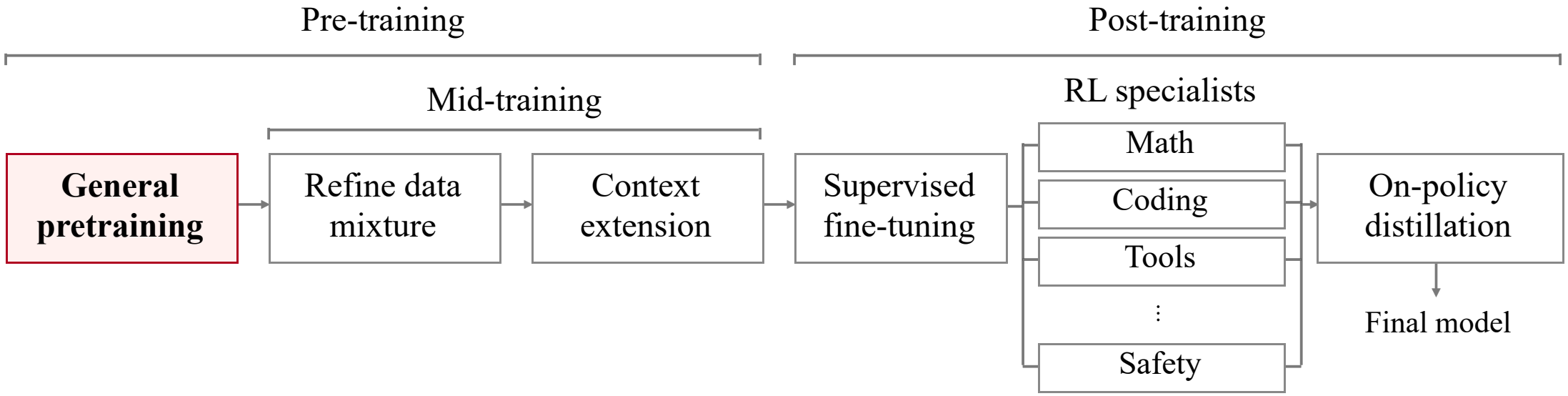
Models	**Documented tokenizer**
GPT-3.5 Turbo, GPT-4	cl100k_base [8] Byte-level BPE [14]
GPT-4o, GPT-4.1, GPT-5	o200k_base [8] Byte-level BPE [14]
Legacy Claude tokenizer utility	Public BPE utility [15] Exact model-version mapping unspecified.
Claude 3 and later	Legacy utility is inaccurate [15]. Use model-specific token counts [16].

For ChatGPT, specify the underlying model when discussing tokenization.

[8] OpenAI mappings; [14] encoding definitions; [15] Anthropic legacy utility; [16] token counting.

LLM Training Pipeline

One reported recipe: MiMo-V2-Flash [17]. All scales are specific to this model.



1. General pretraining

Data: 22 trillion tokens (0–22T cumulative) [17, §3.1].

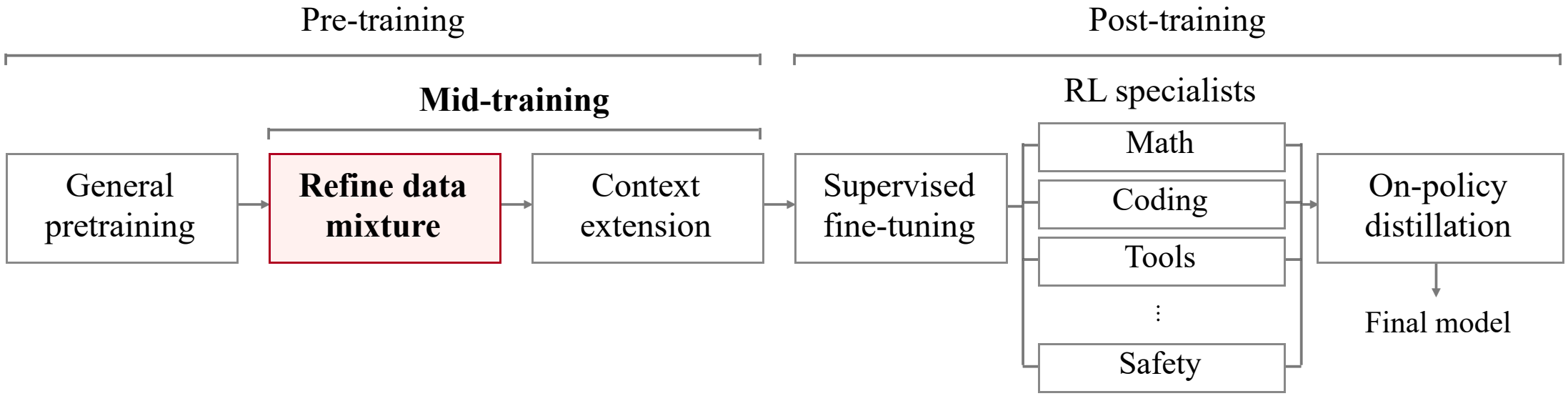
Training sequence length: 32K tokens [17, §3.1].

Expected behavior: broad language, knowledge and initial problem-solving.

[17] MiMo Team, MiMo-V2-Flash Technical Report, 2026, §§3.1–3.2 (the two mid-training substeps are separate phases in this report).

LLM Training Pipeline

One reported recipe: MiMo-V2-Flash [17]. All scales are specific to this model.



2a. Mid-training: change the training mix

Data: 4 trillion additional tokens (22–26T cumulative) [17, §3.1].

Training sequence length: 32K before extension (inferred from [17, §3.2]).

Goal: strengthen code and reasoning through more targeted examples.

[17] MiMo Team, MiMo-V2-Flash Technical Report, 2026, §§3.1–3.2 (the two mid-training substeps are separate phases in this report).

What Is a Data Mixture?

1. Data mixture: the proportions of training tokens sampled from different sources.

$$\mathcal{L}(\theta) = \sum_{d=1}^{D} \alpha_d \, \mathbb{E}_{(y_{<t}, y_t) \sim \mathcal{D}_d} [-\log \pi_\theta(y_t \mid y_{<t})]$$

D source count α_d token share $\mathcal{D}_d$ source distribution

$$\alpha_d \geq 0, \qquad \sum_{d=1}^{D} \alpha_d = 1$$

2. Upsampling a source means drawing from it more often.
More of the training loss then comes from its examples.

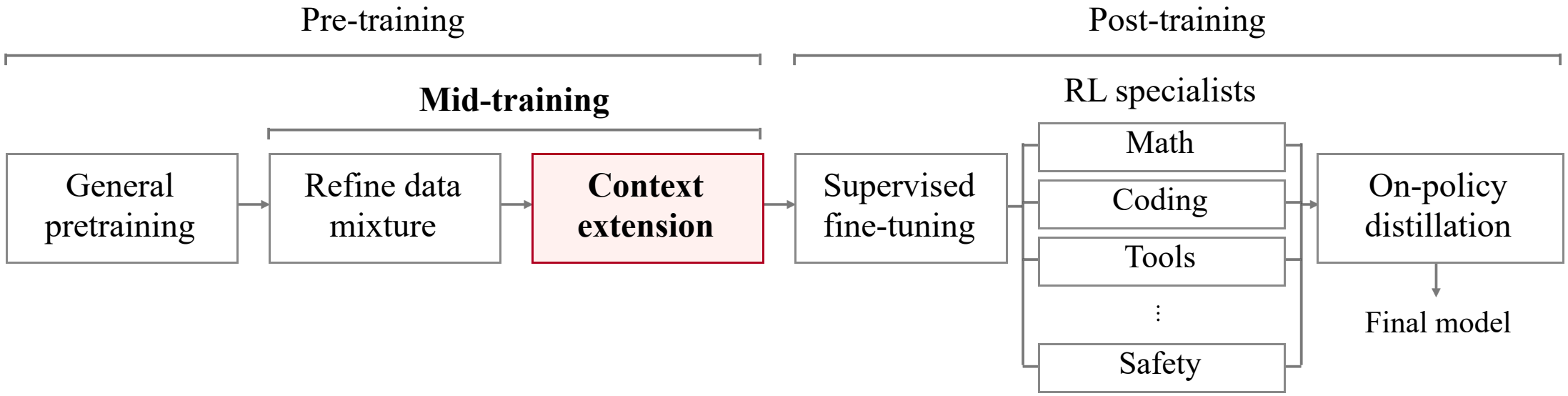
3. MiMo upsamples code and includes about 5% synthetic reasoning data [17, §3.1].

More practice can strengthen those skills. Measure gains and regressions.

[17] MiMo Team, 2026.

LLM Training Pipeline

One reported recipe: MiMo-V2-Flash [17]. All scales are specific to this model.



2b. Mid-training: context extension

Data: 1 trillion additional tokens (26–27T cumulative) [17, §3.1].

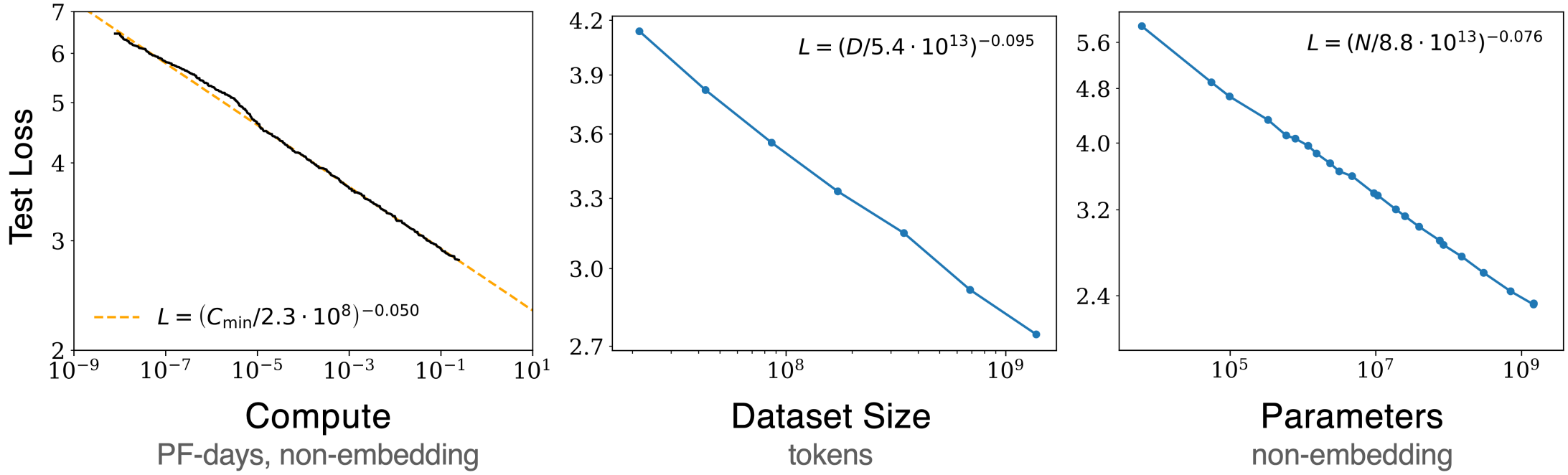
Training sequence length: 32K to 256K tokens [17, §3.2].

Expected behavior: use evidence and dependencies across longer documents.

[17] MiMo Team, MiMo-V2-Flash Technical Report, 2026, §§3.1–3.2 (the two mid-training substeps are separate phases in this report).

Pretraining Scaling Laws

Empirical fits for autoregressive pretraining on a fixed data recipe.



Prediction loss decreases systematically with pretraining scale.

Changing the mixture or context length requires new measurements.

[18] Kaplan et al., 2020, Fig. 1 and §2.

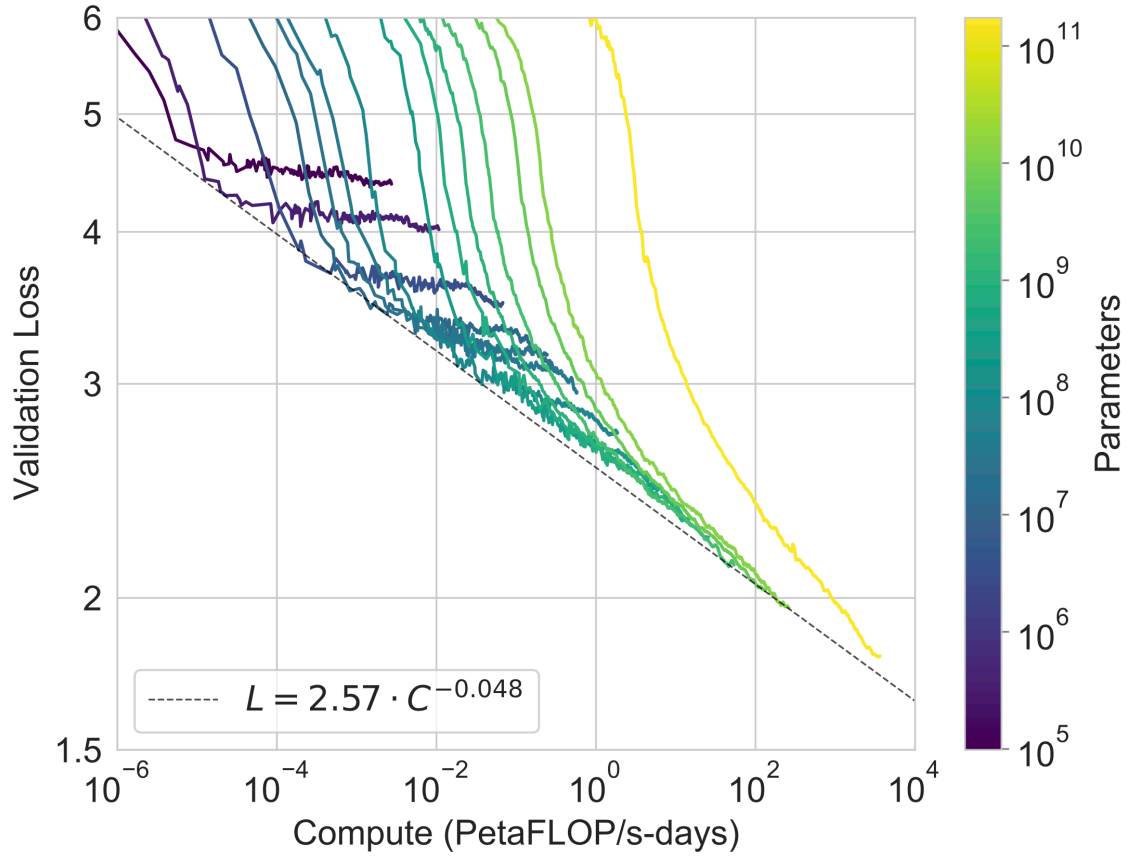
Compute-Efficient Training

A training run trades model size against training duration.

1. Each colored curve trains one fixed model size for longer.

2. More steps eventually yield diminishing improvements for that size.

Choose model size and training duration for the compute budget.



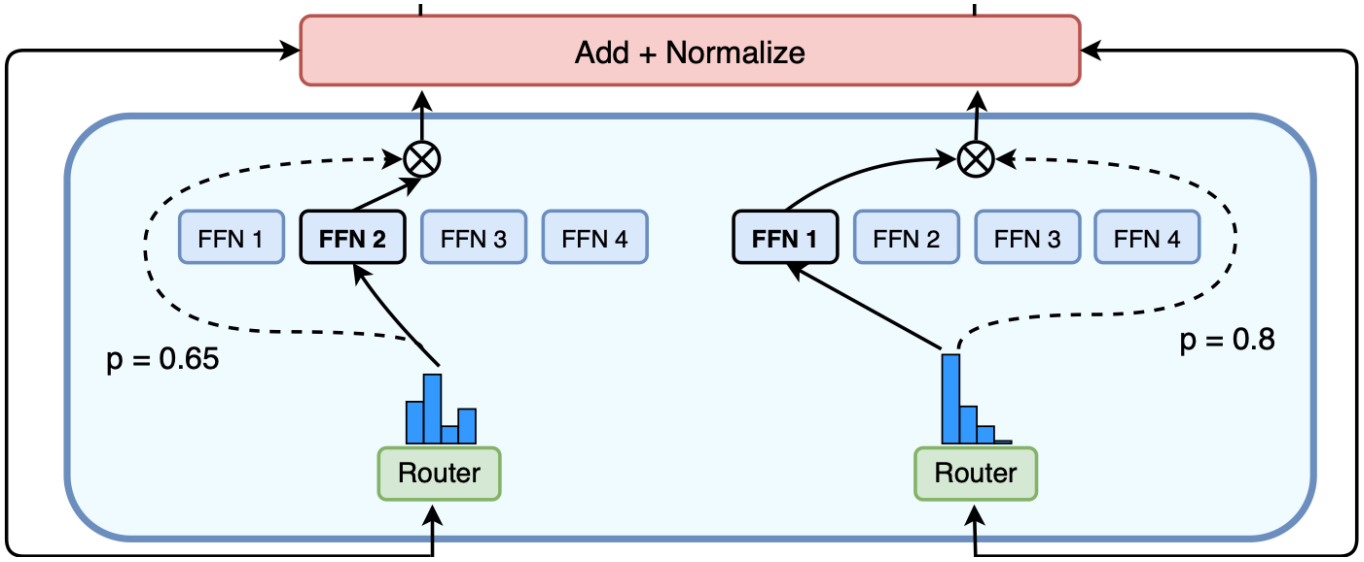
[19] Brown et al., NeurIPS 2020, Fig. 3.1.

Mixture of Experts: Gating

Softmax scores every expert. Top-k selects which feed-forward networks run.

$$\mathcal{T}_k(v) = \mathrm{TopK}(\mathrm{softmax}(W_{\mathrm{r}} v),\ k)$$

$$z(v) = \sum_{e \in \mathcal{T}_k(v)} \left[\mathrm{softmax}(W_{\mathrm{r}} v)\right]_e f_e(v)$$



W_{r} learned router matrix

$\mathcal{T}_k(v)$ selected expert indices

$f_e(v)$ expert FFN output

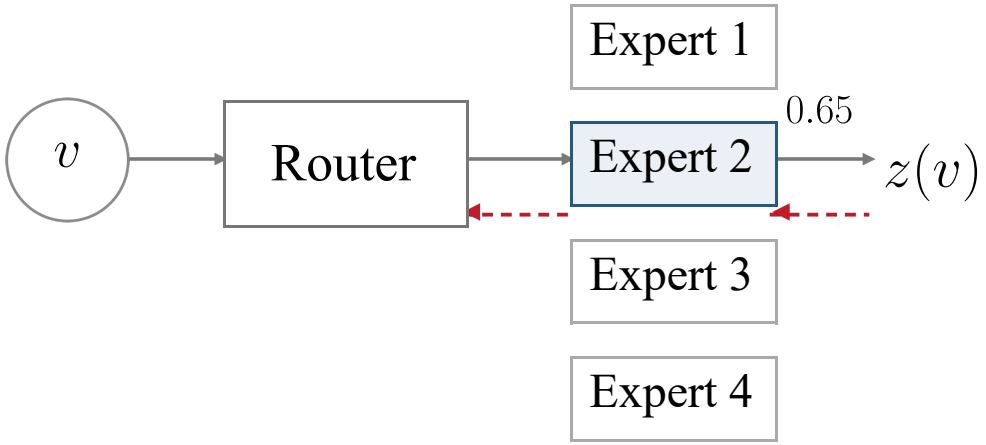
[20] Fedus et al., Switch Transformers, JMLR 2022, §2.1, Eqs. (1)–(2), Fig. 2.

MoE: Training and Inference

Switch-style top-1: hard selection with continuous gate weights in both phases.

$$p(v) = \mathrm{softmax}(W_r v) = (0.10, 0.65, 0.15, 0.10)$$

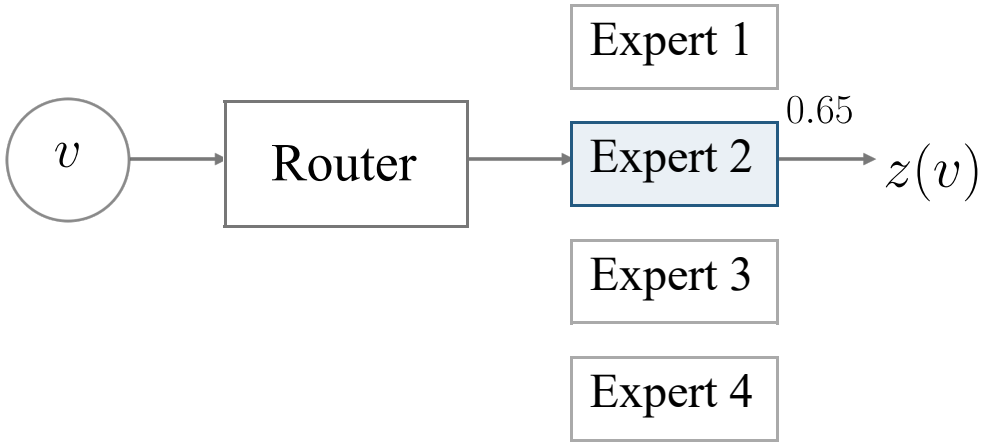
Training



task gradients

Update the router and selected expert.
Differentiate through gate values.

Inference



fixed parameters

Use the same hard top-1 choice.
Keep learned parameters fixed.

Same weighted output in both phases:

$$z(v) = 0.65 f_2(v)$$

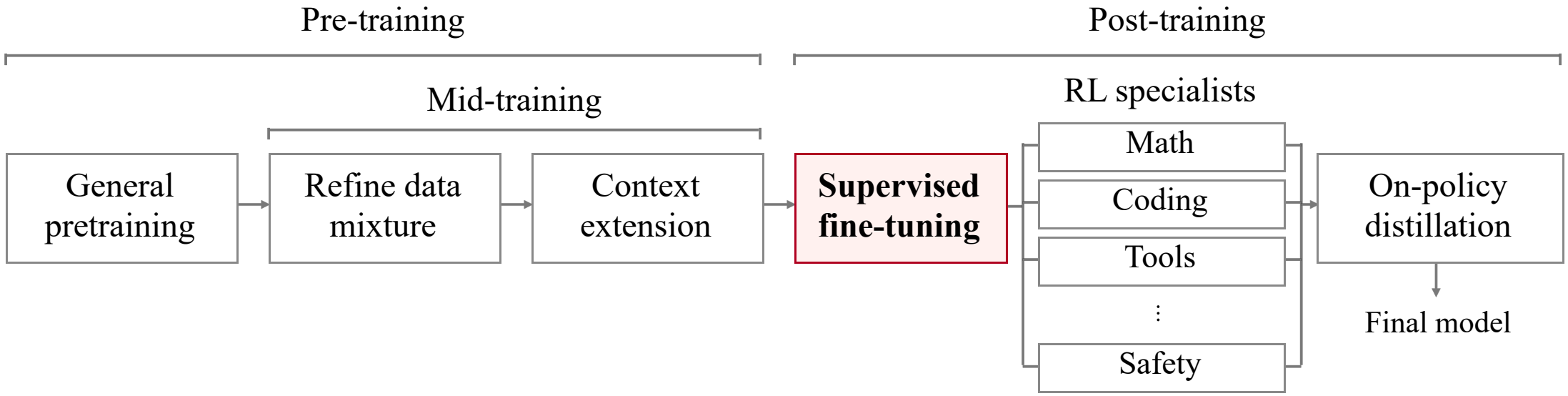
[20] Fedus et al., Switch Transformers, JMLR 2022, §2.1, Eqs. (1)–(2).

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- **Post-training**
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- Inference
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

LLM Training Pipeline

One reported recipe: MiMo-V2-Flash [17]. All scales are specific to this model.



3. Supervised fine-tuning (SFT)

Data: millions of demonstration samples; an exact count is not reported.

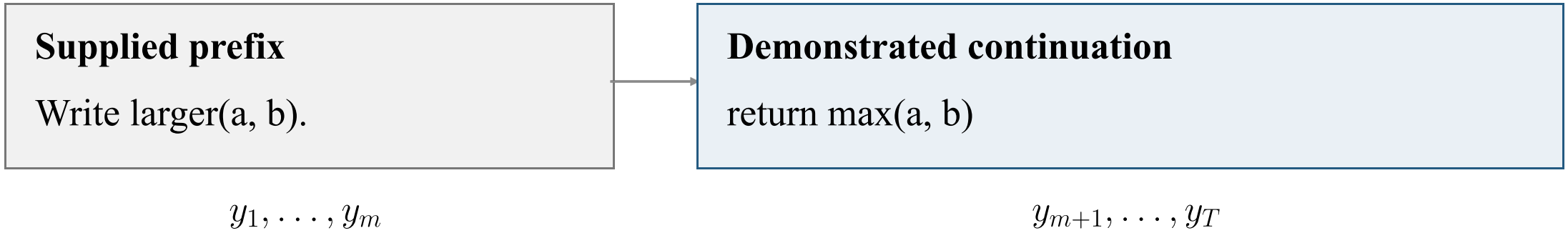
Training sequence length: no uniform SFT cap reported.

Expected behavior: follow requests and demonstrated response/tool formats.

[17] MiMo Team, MiMo-V2-Flash Technical Report, 2026, §§4.1–4.4.

Supervised Fine-Tuning

Use demonstrations of desired assistant responses.



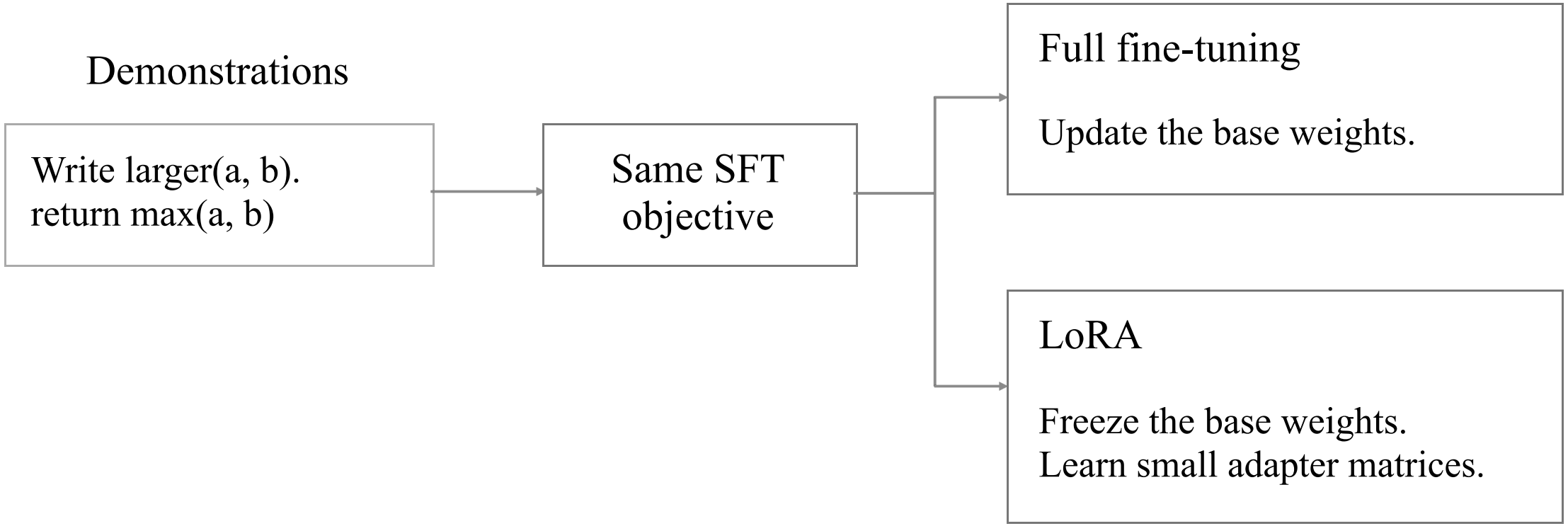
$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(y,m)\sim\mathcal{D}_{\text{demo}}} \sum_{t=m+1}^{T} \log \pi_\theta(y_t \mid y_{<t})$$

Train on response tokens. The prefix supplies context.

[21] Ouyang et al., NeurIPS 2022.

Parameter-Efficient Fine-Tuning (PEFT)

SFT defines the learning objective. PEFT limits which parameters are trained.

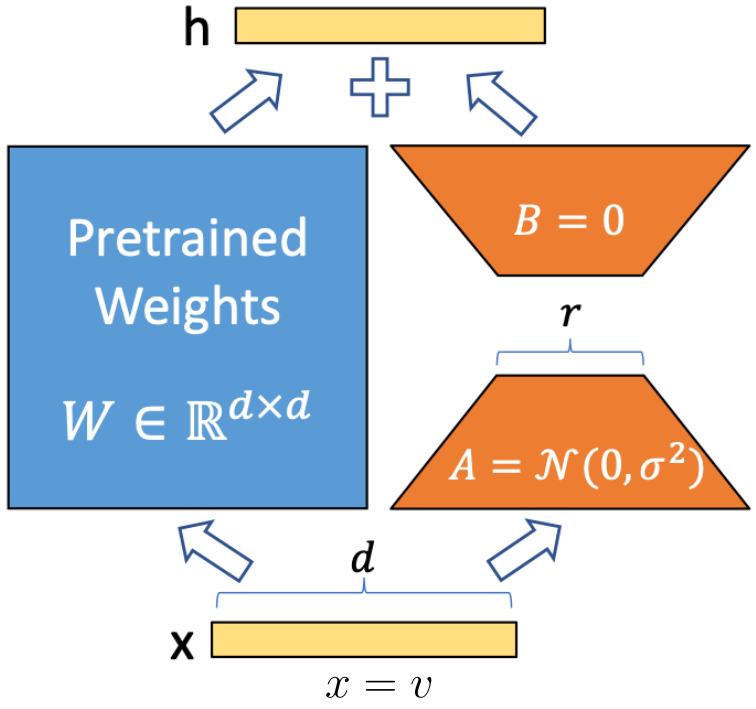


PEFT can also use DPO or RL objectives.

[22] Hu et al., ICLR 2022.

Low-Rank Weight Updates

LoRA freezes the base matrix and learns two small adapter matrices.



$$h = \left(W + \frac{\alpha}{r} BA\right) v$$

$$W \in \mathbb{R}^{d \times d}, \quad A \in \mathbb{R}^{r \times d}, \quad B \in \mathbb{R}^{d \times r}$$

r adapter rank α update scaling

The base matrix is frozen.

The adapter matrices are trained.

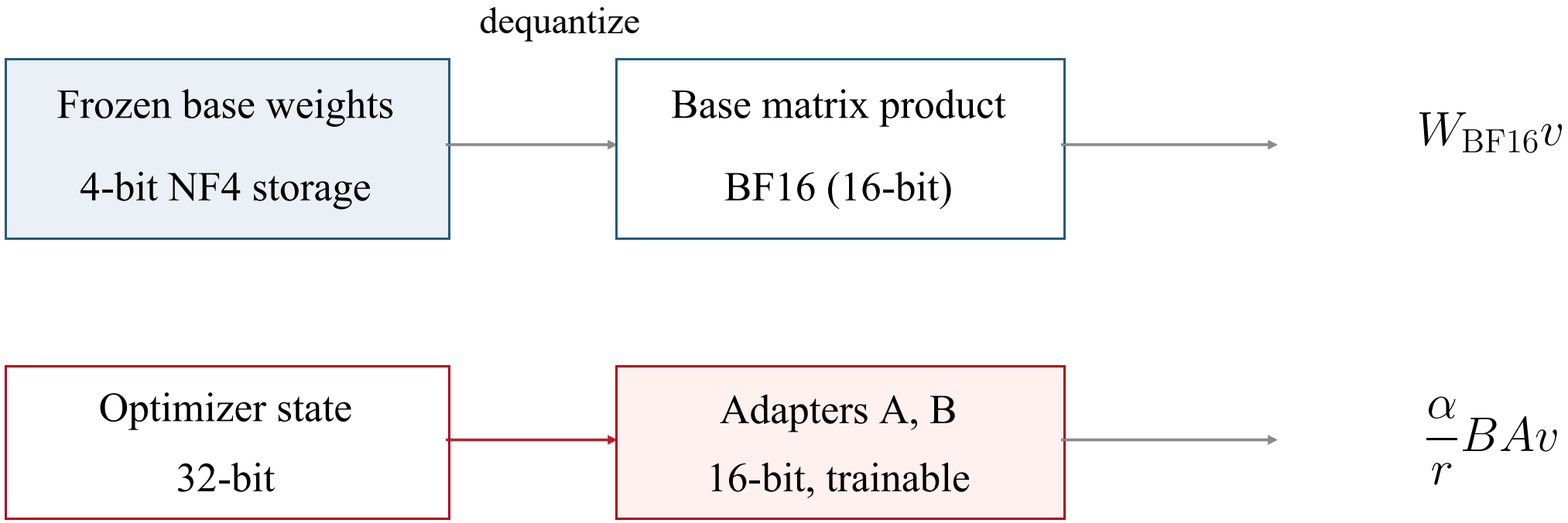
The SFT loss stays unchanged.

Low rank limits adaptation capacity while reducing trainable parameters.

[22] Hu et al., ICLR 2022.

QLoRA: Storage and Computation

The base uses 4-bit storage and 16-bit computation.



- **Training:** add both outputs and backpropagate; update only adapters.
- **Inference:** add both outputs with fixed weights; no optimizer state.

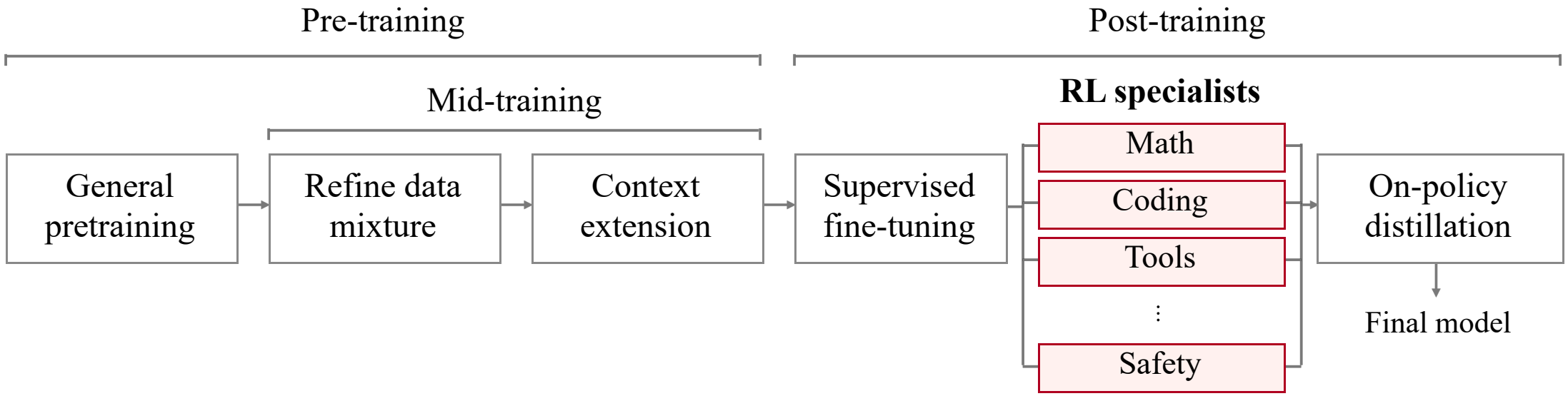
[23] Dettmers et al., NeurIPS 2023.

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- **Post-training**
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- Inference
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

LLM Training Pipeline

One reported recipe: MiMo-V2-Flash [17]. All scales are specific to this model.



4. Domain-specialized reinforcement learning

Data examples: 120K coding, 150K search and 50K general-agent tasks.

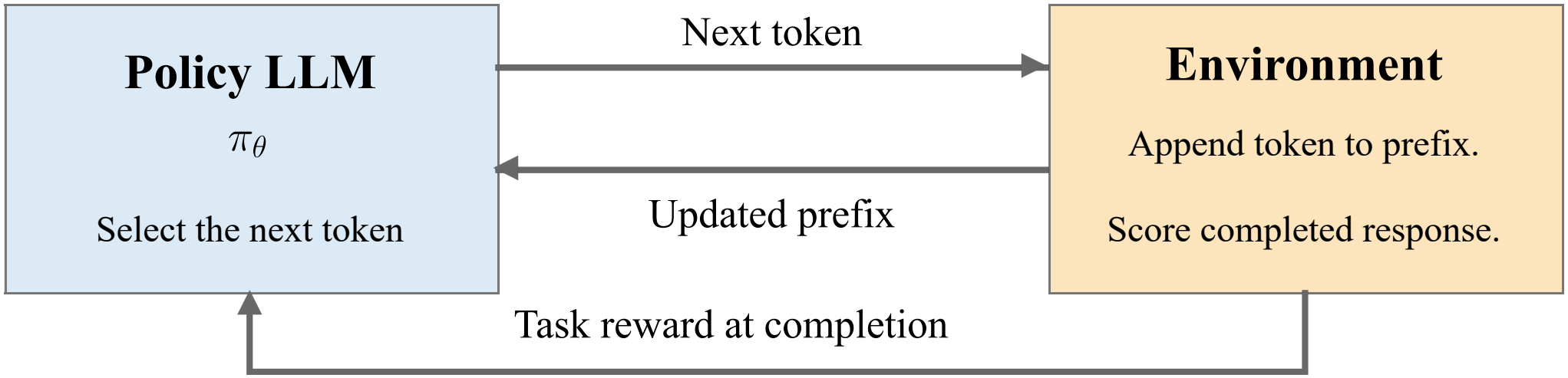
Training length and total sampled-response budget: not reported.

Expected behavior: improve task success in each specialist domain.

[17] MiMo Team, MiMo-V2-Flash Technical Report, 2026, §§4.1–4.4.

Text Generation as an RL Interaction

An episode generates one response, one token at a time.



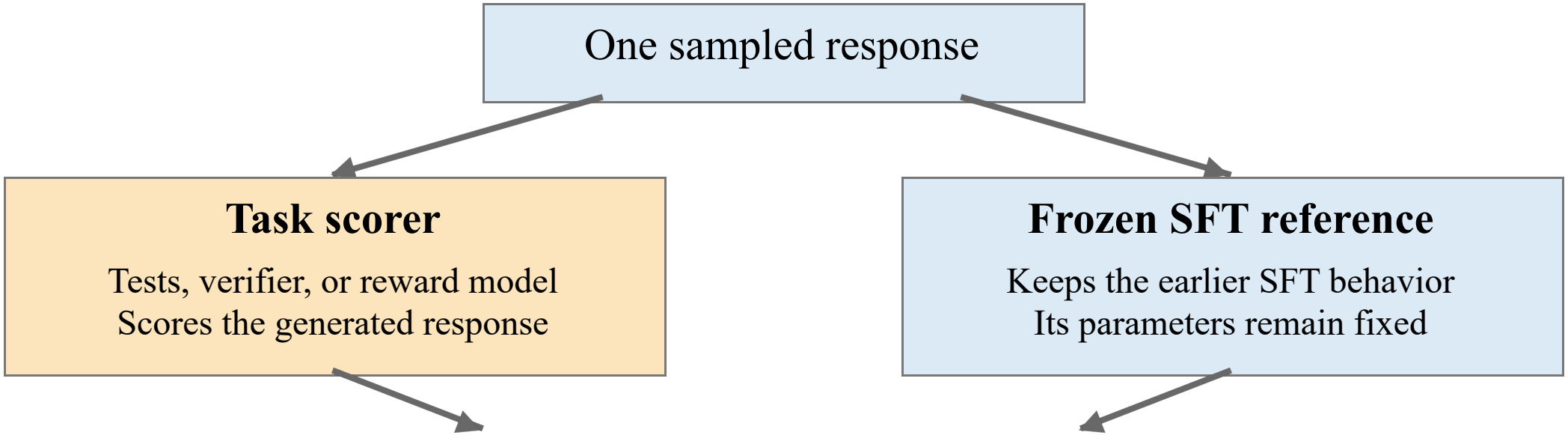
$$s_t = y_{<t}, \quad a_t = y_t \sim \pi_\theta(\cdot \mid s_t), \quad s_{t+1} = (s_t, y_t), \quad t > m$$

The supplied prompt is the initial prefix. Generated tokens are actions.

[21] Ouyang et al., NeurIPS 2022; [24] Schulman et al., 2017.

Expected Reward Minus Reference KL

Maximize task success while limiting drift from a frozen SFT policy.



$$J(\theta) = \mathbb{E}_{y_{m+1:T} \sim \pi_\theta}[R(y)] - \beta\, D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})$$

The KL divergence compares distributions over complete responses to one prompt.

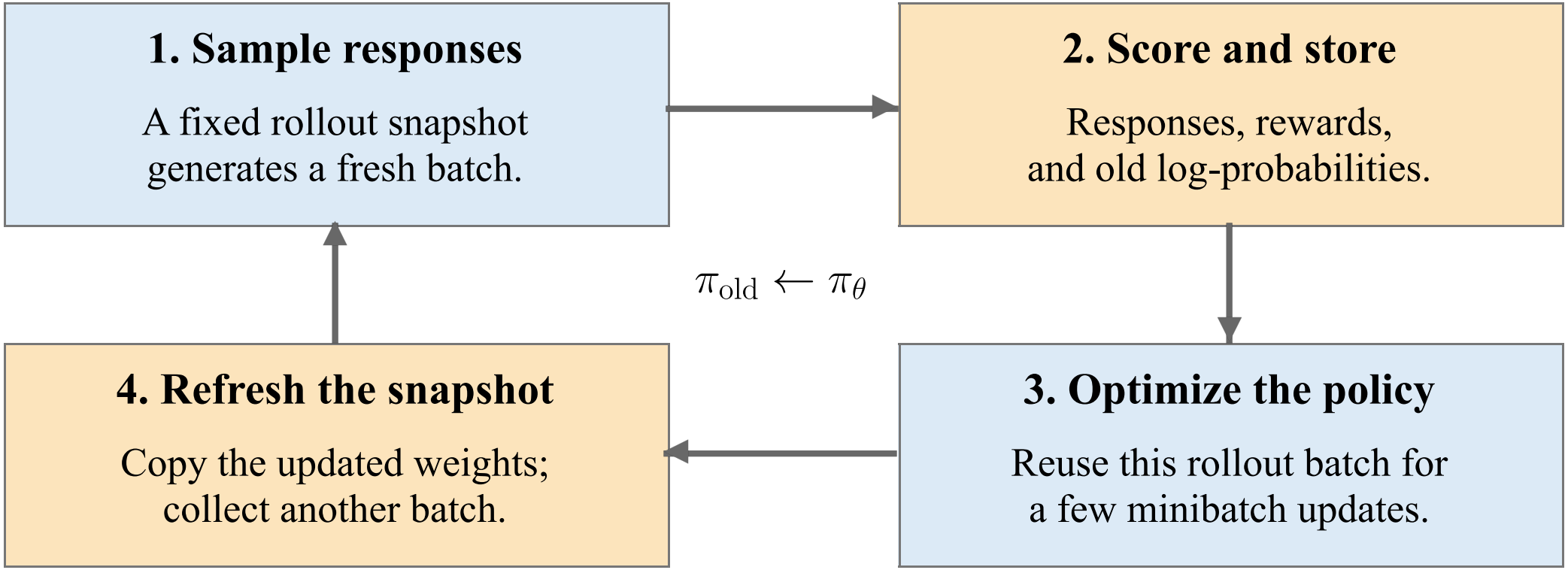
It is zero when they agree. Greater regularization discourages policy drift.

$R(y)$ terminal task score $\beta \geq 0$ regularization strength

[21] Ouyang et al., NeurIPS 2022, §3.5.

Rollouts, Learning, and Weight Synchronization

Separate the sampling snapshot from the frozen SFT reference.



Old policy: collected this batch. Reference: stays fixed as the KL anchor.

[24] Schulman et al., 2017, Algorithm 1.

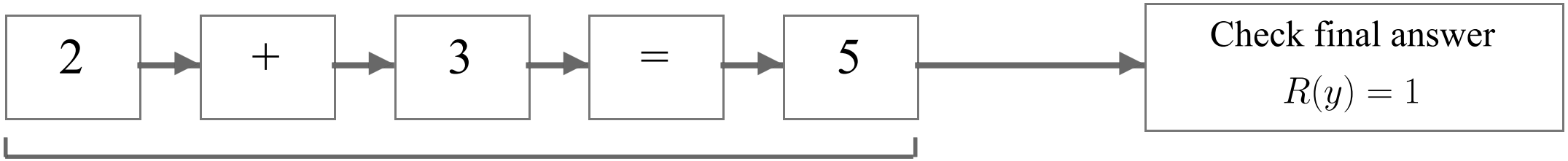
A Final Score Leaves Credit Assignment Unresolved

One final score rewards the whole response, not individual token decisions.

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R(y)] = \mathbb{E}_{\pi_\theta}\left[R(y) \sum_{t=m+1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid y_{<t})\right]$$

Prompt: What is 2 + 3?

Illustrative generated tokens



The same score weights all token log-gradients.

Which token caused the success? The final score does not tell us.

A baseline will make this learning signal less noisy.

[24] Schulman et al., 2017; score-function policy-gradient estimator.

Return, Value, and Advantage

The state is the generated prefix. The action is the next token.

$$s_t = y_{<t}, \qquad a_t = y_t, \qquad m < t \le T$$

Return

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k, \qquad 0 \le \gamma \le 1$$

Value

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s]$$

Action value

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a]$$

Advantage

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

$\gamma = 1$ Finite responses often use no discount, as in InstructGPT.

r_k per-token reward, including the terminal score and any KL shaping.

[21] Ouyang et al., NeurIPS 2022; [24] Schulman et al., 2017; [25] Hugging Face TRL, PPO Trainer.

A Baseline Reduces Gradient Variance

For the undiscounted finite-response objective:

1. Estimate the expected return from each prefix.

$$V_\phi(s_t) \approx V^{\pi_{\text{old}}}(s_t)$$

2. Center the sampled return around that baseline.

$$\widehat{A}_t = G_t - V_\phi(s_t)$$

3. Weight the policy-gradient estimate by this centered signal.

$$\widehat{g} = \sum_{t=m+1}^{T} \widehat{A}_t \, \nabla_\theta \log \pi_\theta(y_t \mid s_t)$$

A suitable baseline lowers gradient variance without changing the expected gradient.

[21] Ouyang et al., NeurIPS 2022; [24] Schulman et al., 2017.

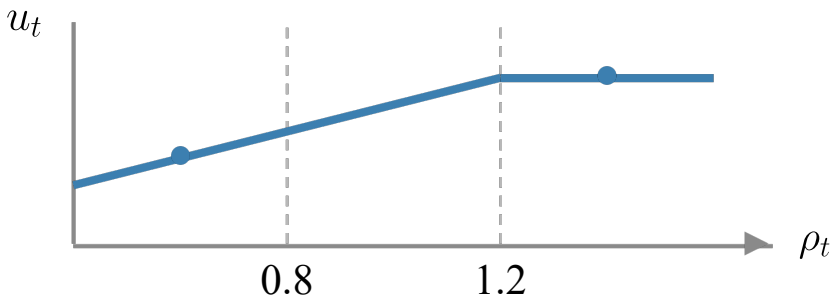
PPO: Clipping Limits Update Incentives

Maximize the surrogate. The old policy and sampled advantages stay fixed.

$$\rho_t = \frac{\pi_\theta(y_t \mid s_t)}{\pi_{\text{old}}(y_t \mid s_t)}$$

$$u_t = \min\left\{\rho_t \widehat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\widehat{A}_t\right\}$$

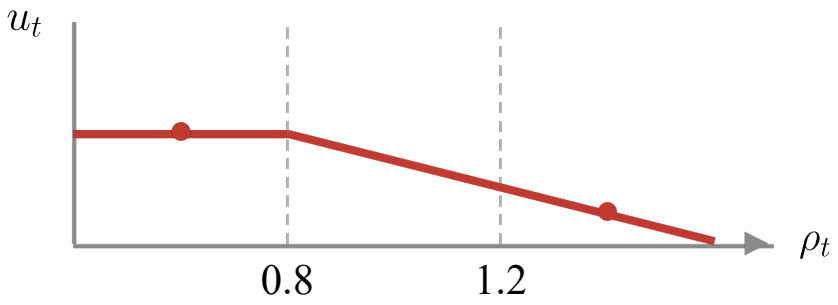
$\widehat{A}_t = +1, \quad \epsilon = 0.2, \quad u_t = \min(\rho_t, 1.2)$



$\rho_t = 0.6 : \quad u_t = 0.6$

$\rho_t = 1.4 : \quad u_t = 1.2$

$\widehat{A}_t = -1, \quad \epsilon = 0.2, \quad u_t = -\max(\rho_t, 0.8)$



$\rho_t = 0.6 : \quad u_t = -0.8$

$\rho_t = 1.4 : \quad u_t = -1.4$

Beneficial changes saturate. Changes in the wrong direction still lower the objective.

[24] Schulman et al., Proximal Policy Optimization Algorithms, 2017, Eq. 7 and Fig. 1.

GRPO: Relative Rewards Without a Critic

Sample G responses from the old policy for each shared prompt.

$$\bar{R} = \frac{1}{G}\sum_{j=1}^{G} R_j, \quad s_R = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(R_j - \bar{R})^2}, \quad \widehat{A}_i = \frac{R_i - \bar{R}}{s_R + \delta}, \quad \delta > 0$$

Each response gets one outcome score and one advantage for all its tokens.

$$\rho_{i,t} = \frac{\pi_\theta(y_{i,t} \mid s_{i,t})}{\pi_{\text{old}}(y_{i,t} \mid s_{i,t})}, \quad s_{i,t} = y_{i,<t}, \quad N_i = T_i - m$$

$$\mathcal{J}_{\text{group}}(\theta) = \mathbb{E}_{\mathcal{B}}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{N_i}\sum_{t=m+1}^{T_i}\Big(\min\{\rho_{i,t}\widehat{A}_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_i\} - \beta D_{\text{KL}}(\pi_\theta(\cdot \mid s_{i,t}) \,\|\, \pi_{\text{ref}}(\cdot \mid s_{i,t}))\Big)\Big].$$

A group supplies the baseline. The reference KL limits policy drift.

The expectation averages prompt groups in the rollout batch.

[26] Shao et al., DeepSeekMath, 2024, §4.1. Teaching form with full per-prefix KL.

Why Groups: Learn From Relative Outcomes

The same prompt gives a shared context for comparing response quality.

Too hard for this policy

0 0 0 0

Centered task signal



No successful alternative in this sampled group.

Already solved here

1 1 1 1

Centered task signal

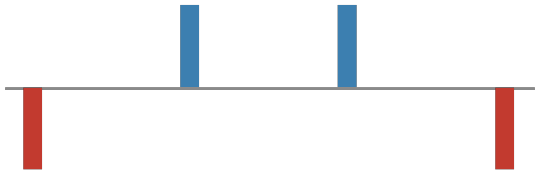


No relative ranking from equal rewards.

Mixed outcomes

0 1 1 0

Centered task signal



Positive and negative weights distinguish responses.

Group scores replace a critic. Useful relative updates need varied outcomes.

Equal scores give zero centered task signal. KL and other losses may remain.

[26] Shao et al., DeepSeekMath, 2024. [27] Yu et al., DAPO, 2025.

One Prompt, Four Responses

Write larger(a,b). Tests: (1,2) → 2 and (2,1) → 2.

return a

Reward 0 $\widehat{A}_1 \approx -1$

return a if a >= b else b

Reward 1 $\widehat{A}_2 \approx +1$

return max(a, b)

Reward 1 $\widehat{A}_3 \approx +1$

return b

Reward 0 $\widehat{A}_4 \approx -1$

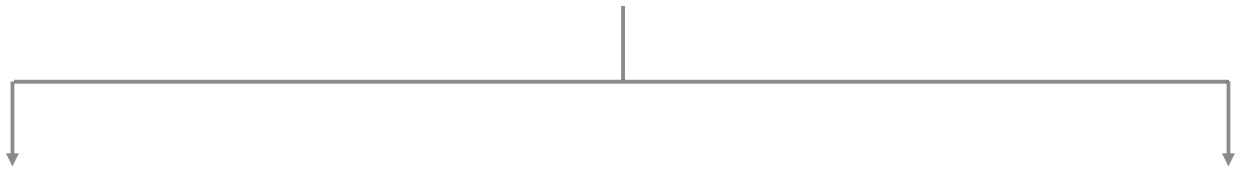
$$\bar{R} = \tfrac{1}{2}, \qquad s_R = \tfrac{1}{2}$$

Increase successful responses relative to unsuccessful ones.

[26] Shao et al., 2024.

Additional Option: Preference Data

Some goals are easier to rank than to score automatically.



Send me the report.

Less preferred: abrupt.

Could you send the report, please?

Preferred: polite request.

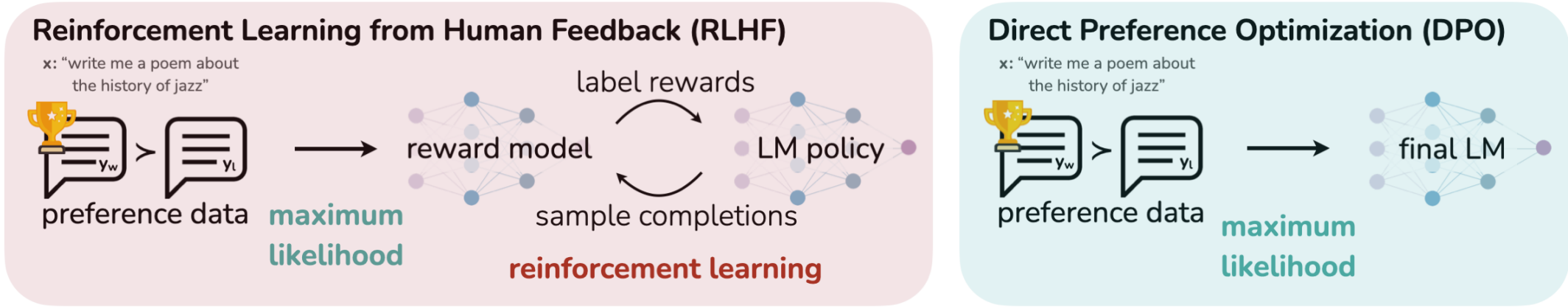
$$y^{(w)}_{\leq m} = y^{(l)}_{\leq m}$$

$$\ell_\theta(y; m) = \sum_{t=m+1}^{T} \log \pi_\theta(y_t \mid y_{<t})$$

[21] Ouyang et al., NeurIPS 2022; [28] Rafailov et al., NeurIPS 2023.

Direct Preference Optimization (DPO)

Additional option: fit preferences when online rewards are costly.



$$\Delta_\theta = \left[\ell_\theta(y^{(w)}; m) - \ell_{\text{ref}}(y^{(w)}; m)\right] - \left[\ell_\theta(y^{(l)}; m) - \ell_{\text{ref}}(y^{(l)}; m)\right]$$

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}_{\text{pref}}} \log \sigma(\beta \Delta_\theta)$$

Fixed preference pairs are required; this update uses no fresh rollouts.

$\beta > 0$ preference scale; σ logistic sigmoid

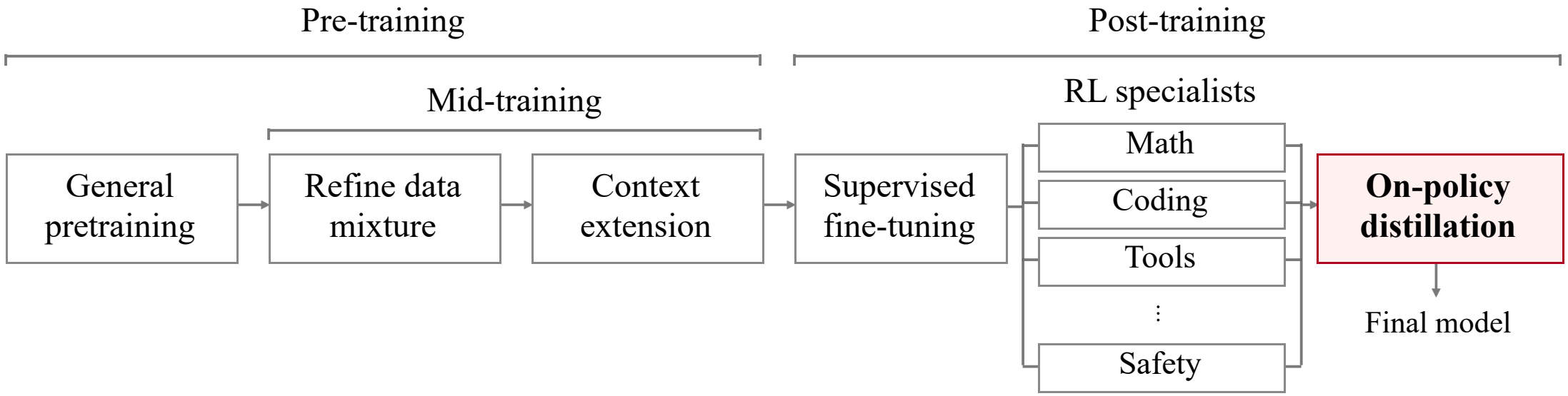
[28] Rafailov et al., NeurIPS 2023, Fig. 1.

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- **Post-training**
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- Inference
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

LLM Training Pipeline

One reported recipe: MiMo-V2-Flash [17]. All scales are specific to this model.



5. Multi-teacher on-policy distillation

Data: fresh student-generated responses; total budget is not reported.

Training sequence length: no uniform distillation cap reported.

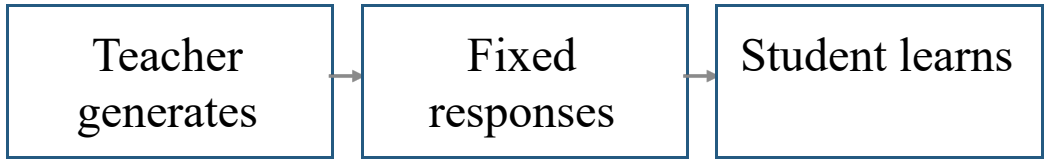
Expected behavior: integrate specialist skills into one student model.

[17] MiMo Team, MiMo-V2-Flash Technical Report, 2026, §§4.1–4.4.

Offline and On-Policy Distillation

The distinction is who generates the training responses.

Offline distillation



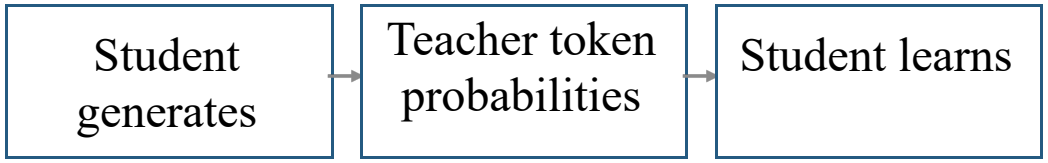
Benefit

Reusable demonstrations and ordinary supervised training.

Cost

Teacher examples may miss the prefixes created by student mistakes.

On-policy distillation



Benefit

Fit teacher token probabilities on student-generated prefixes.

Cost

Fresh rollouts and teacher scoring add training cost.

[17] MiMo Team, 2026; [29] Thinking Machines Lab, 2025.

Model Behavior After Training Stages

Constructed teaching examples, not measured model outputs.

Shared request: solve $x^2 - 5x + 6 = 0$

Base checkpoint	Instruction checkpoint	Reasoning checkpoint
After pre-training	After SFT	After reasoning SFT / RL
May continue a worksheet:	May answer directly:	May show a solution:
$x^2 - 7x + 12 = 0$	$x \in \{2, 3\}$	$(x - 2)(x - 3) = 0$

Distillation can transfer reasoning behavior into a consolidated student.

A written solution trace alone does not guarantee correctness.

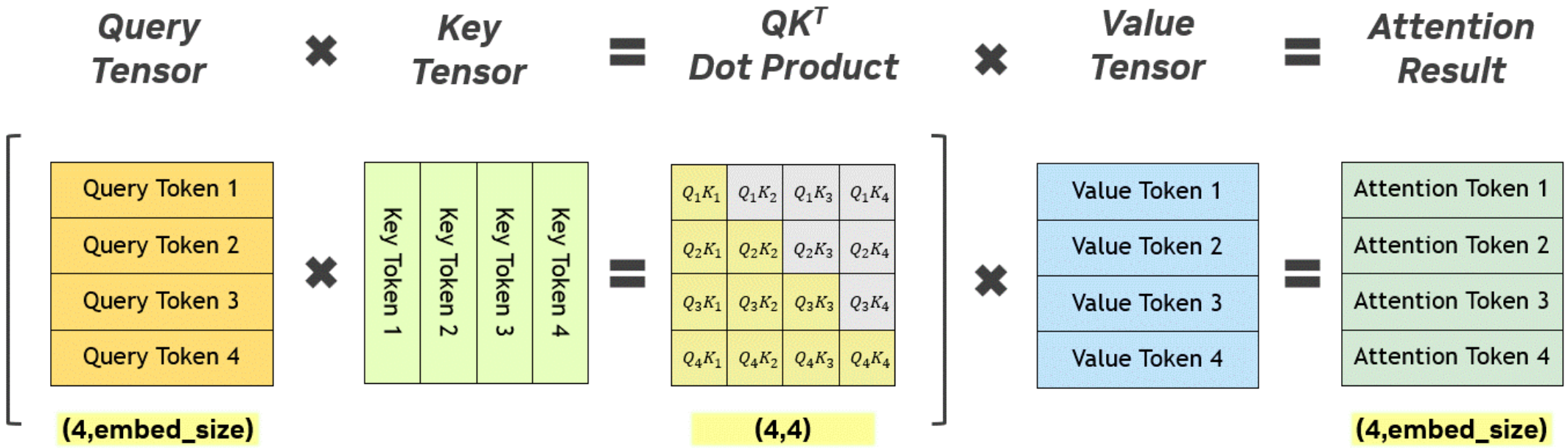
[7] Qwen Team, 2025; [21] Ouyang et al., NeurIPS 2022.

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- Post-training
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- **Inference**
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

KV Cache: Prefill

Compute the whole prefix once, then retain K/V at every decoder layer.



d_h head width

M causal mask

$$H = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}} + M\right) V$$

[30] Alvarez et al., NVIDIA, 2025, Fig. 1.

KV Cache Example: Prefill

Prompt: “I am sure this project” (7 illustrative tokens).

Reused cache Computed this call One layer and one head shown

	y_1	y_2	y_3	y_4	y_5	y_6	y_7
Tokens	<bos>	I	am	sure	this	pro	ject
Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7
K	k_1	k_2	k_3	k_4	k_5	k_6	k_7
V	v_1	v_2	v_3	v_4	v_5	v_6	v_7

$$Q_{1:7}, K_{1:7}, V_{1:7} \in \mathbb{R}^{7\times d_h}$$

Cache: 7 positions. The model predicts “will”.

“will” is not cached until it is processed on the next call.

[31] Hugging Face, Continuous batching, 2025; token example and cache mechanism.

KV Cache Example: Decoding “will”

Feed “will” at position 8; reuse the seven cached positions.

Reused cache Computed this call One layer and one head shown

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
Tokens	<bos>	I	am	sure	this	pro	ject	will
Q	Old queries are not recomputed or stored.							q_8
K	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8

$$h_8 = \underbrace{\mathrm{softmax}\left(q_8 K_{1:8}^{\top}/\sqrt{d_h}\right)}_{1\times 8} \underbrace{V_{1:8}}_{8\times d_h}$$

Cache: 8 positions. The model predicts “be”, which is not cached yet.

[31] Hugging Face, Continuous batching, 2025; token example and cache mechanism.

KV Cache Example: Decoding “be”

Feed “be” at position 9; reuse the eight cached positions.

Reused cache Computed this call One layer and one head shown

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
Tokens	<bos>	I	am	sure	this	pro	ject	will	be
Q	Old queries are not recomputed or stored.								q_9
K	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9
V	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9

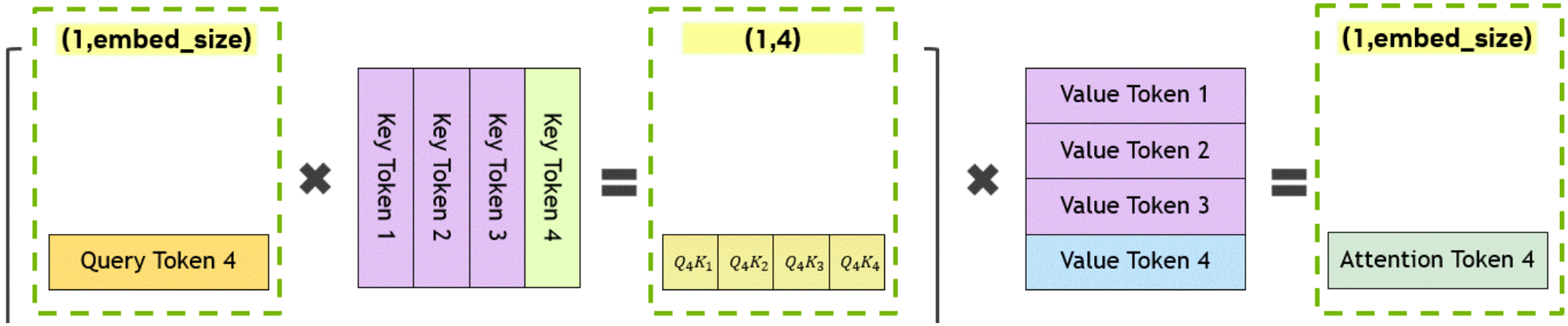
$$h_9 = \underbrace{\mathrm{softmax}\left(q_9 K_{1:9}^{\top}/\sqrt{d_h}\right)}_{1\times 9} \underbrace{V_{1:9}}_{9\times d_h}$$

Cache: 9 positions. The model predicts another token; repeat the same process.

[31] Hugging Face, Continuous batching, 2025; token example and cache mechanism.

KV Cache: Decode Compute

Only the current query attends to the cached prefix.



Reused keys and values

a_t attention weights

$$\underbrace{q_t}_{1\times d_h}\underbrace{K_{1:t}^{\top}}_{d_h\times t} \quad \text{and} \quad \underbrace{a_t}_{1\times t}\underbrace{V_{1:t}}_{t\times d_h}$$

One sequence, one head: matrix–vector products replace full-prefix attention.
Batched decoding combines query rows. Cache reads still grow with context.

[30] Alvarez et al., NVIDIA, 2025, Fig.1 (cropped).

KV Cache: Memory Budget

Two stored vectors per token, per KV head, at every layer.

$$M_{\mathrm{KV}} \approx 2\, n_{\mathrm{batch}}\, n_{\mathrm{layers}}\, n_{\mathrm{ctx}}\, n_{\mathrm{KV}}\, d_h\, b_{\mathrm{el}}$$

n_{batch}	simultaneous sequences	n_{layers}	decoder layers
n_{ctx}	cached tokens per sequence	n_{KV}	KV heads per layer
d_h	width of one head vector	b_{el}	bytes per stored value

Illustrative FP16/BF16 cache: 1 sequence, 32 layers, 8,192 tokens,
8 KV heads, head width 128, and 2 bytes per stored value.

$$2 \times 1 \times 32 \times 8192 \times 8 \times 128 \times 2 = 1\,\mathrm{GiB}$$

Model weights and runtime overhead require additional memory.

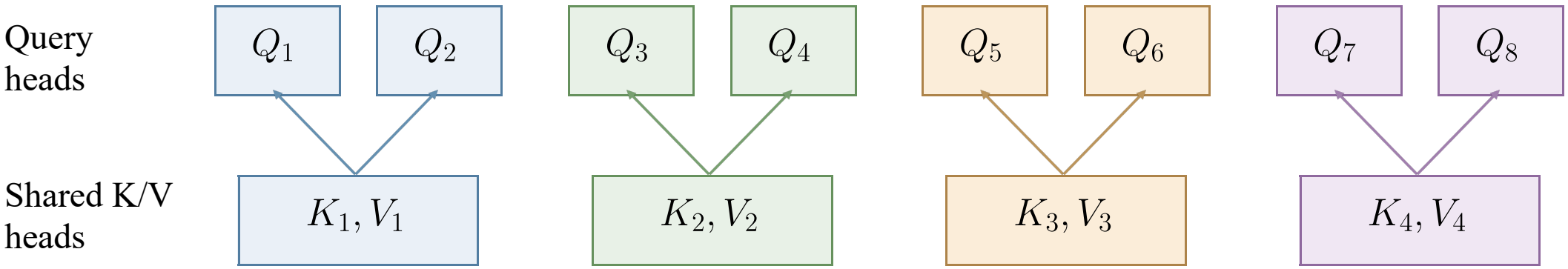
[30] Alvarez et al., NVIDIA, 2025. Numerical configuration is illustrative.

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- Post-training
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- **Inference**
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

Grouped-Query Attention: Fixed Groups

Eight query heads share four K/V heads through four fixed pairs.



$$\text{head}_h = \text{softmax}\left(Q_h K_{\gamma(h)}^{\top} / \sqrt{d_h} + M\right) V_{\gamma(h)}$$

$\gamma(h) = \lceil h/2 \rceil$ maps each query head to its shared K/V head.

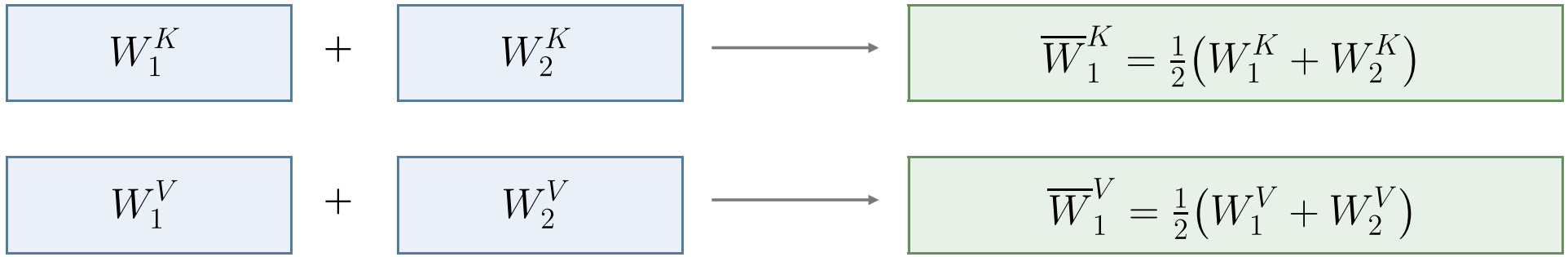
Every query head still attends across the full token prefix.

[32] Ainslie et al., GQA, EMNLP2023, §2.2.

Grouped-Query Attention: Conversion and Quality

Converting a trained MHA model requires adaptation to shared projections.

Example: query heads 1 and 2 share one new K/V head.



Continue pre-training so the shared heads can adapt.

Sharing reduces KV memory but can reduce model capacity.

If quality drops: use more KV groups or adapt further, then evaluate.

Reported GQA quality was close to MHA after uptraining, not guaranteed identical.

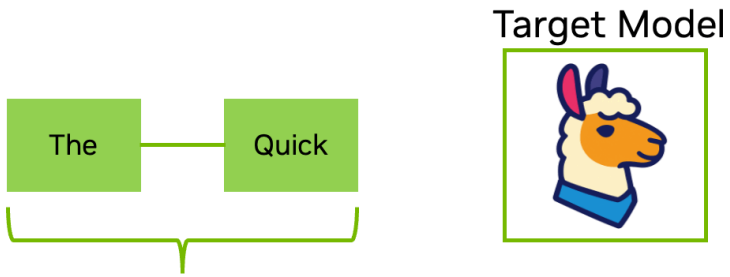
[32] Ainslie et al., GQA, EMNLP2023, §§2.1–2.2,3.3.

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- Post-training
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- **Inference**
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

Speculative Decoding: Draft and Verify

Draft a block → verify all positions together → accept a prefix and correct.

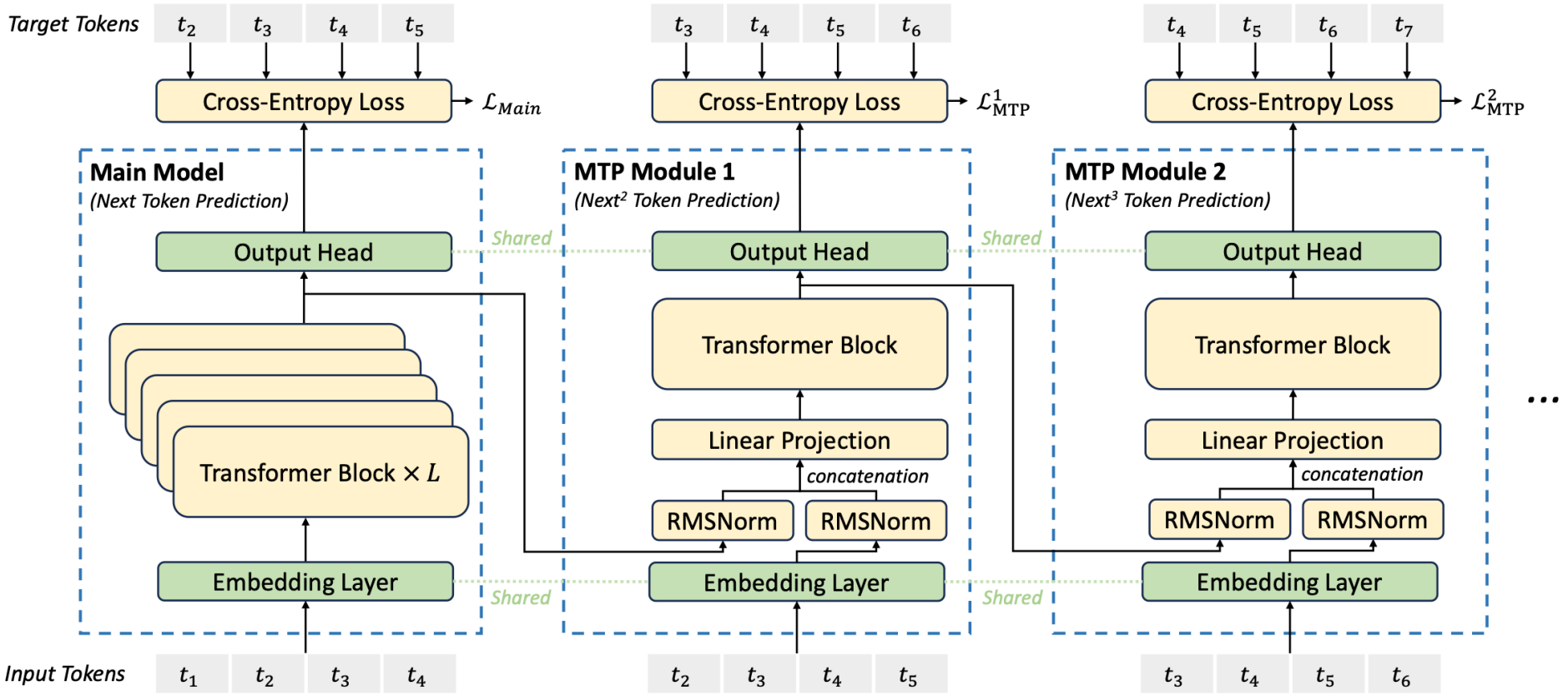


After the first rejection, discard the remaining draft tokens.

[33] NVIDIA, 2025, Fig. 1 (embedded GIF); [34] Leviathan et al., ICML 2023.

Multi-Token Prediction as a Drafter

Training adds auxiliary losses for predicting more future tokens.



At inference, the extra modules can draft tokens for target verification.

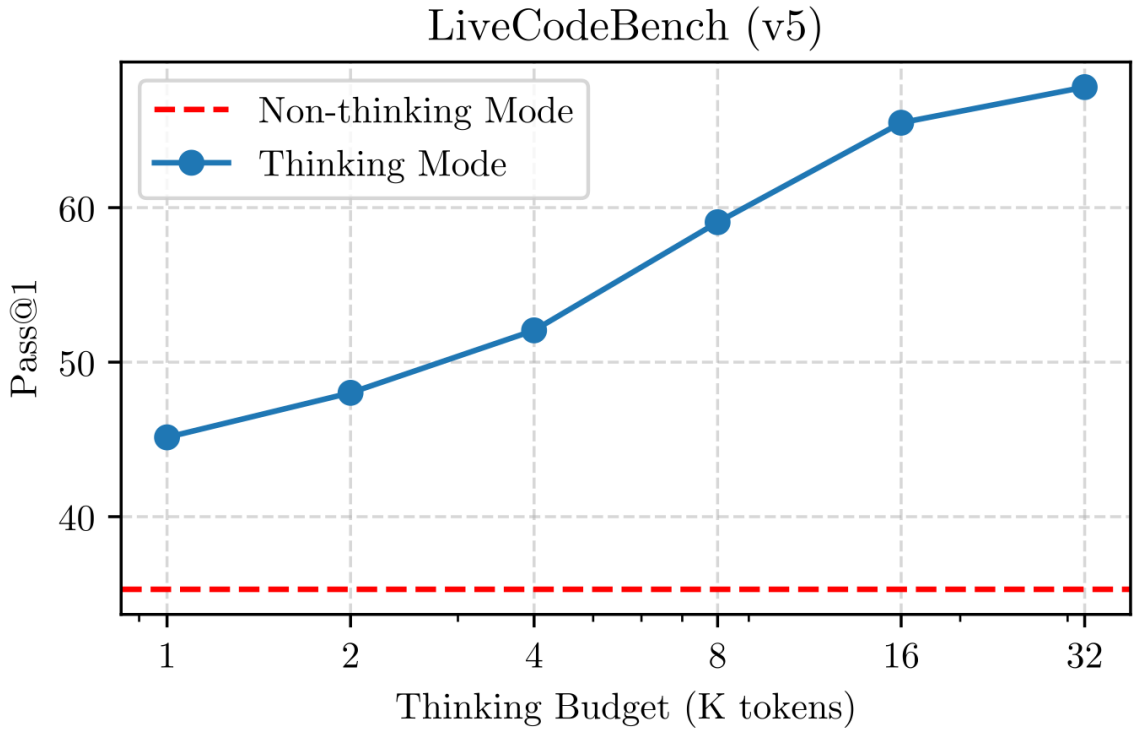
[35] DeepSeek-AI, DeepSeek-V3 Technical Report,2024, §2.2, Fig.3.

Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- Post-training
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- **Inference**
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- Evaluation
  - Sampling budgets, reliable comparisons, and checkpoint progress

Test-Time Scaling

A trained checkpoint can spend more computation on a single request.



Longer trajectory

Generate more tokens before the final answer.

More candidates

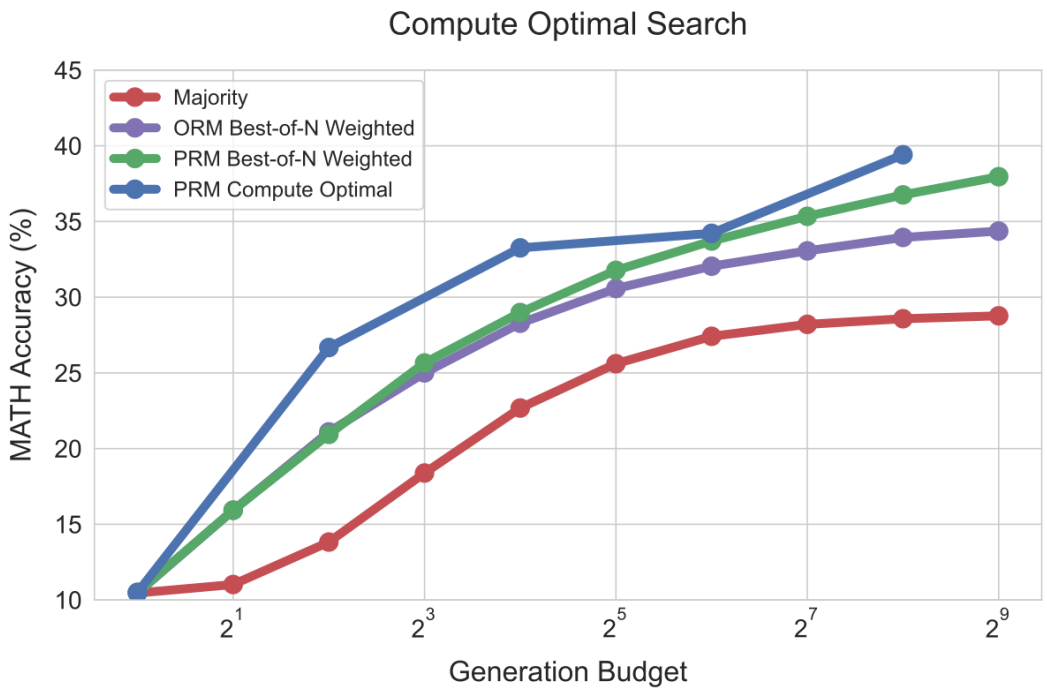
Sample alternative solutions and spend compute selecting one.

Compare accuracy at matched token budgets; also report latency.

[7] Qwen Team, 2025; [36] Snell et al., ICLR 2025.

Generation, Verification, and Selection

Sample solutions, score them, then select an answer.



Outcome reward model (ORM)

Scores a completed response.

Process reward model (PRM)

Scores intermediate reasoning steps.

Compute-optimal allocation

Choose the search strategy and budget for the question.

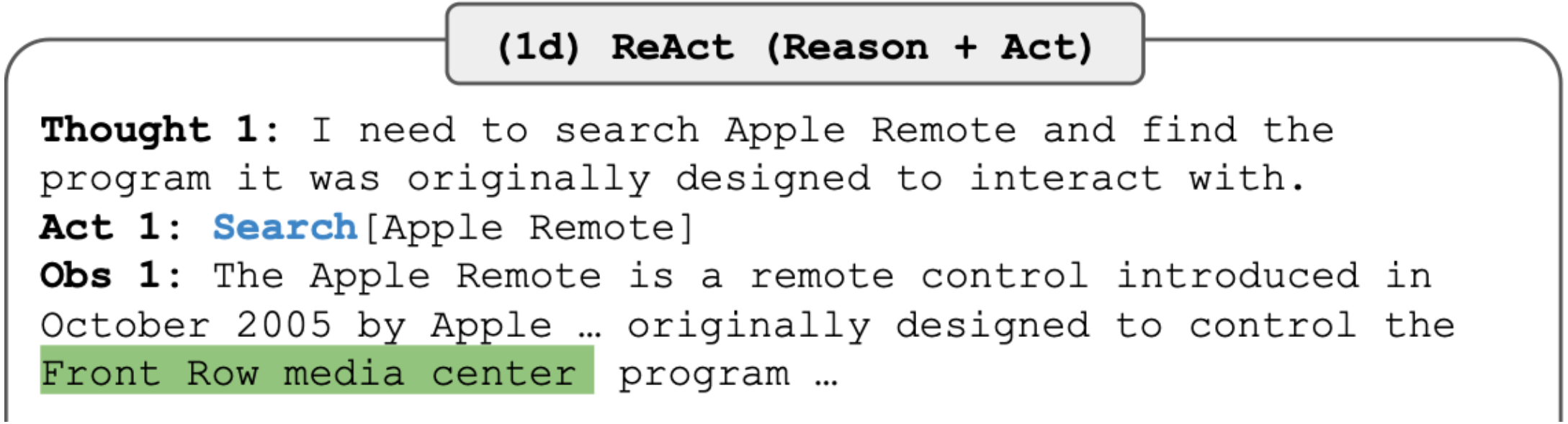
Weighted Best-of-N: sum verifier scores for each final answer; choose the largest.

[36] Snell et al., ICLR 2025, Fig. 1 (cropped).

Tools and Agents

Use tools when a task requires external information or actions.

Example: search for software controlled by Apple Remote.



Tool use: call an external function, such as search or a code interpreter.

Agentic loop: choose an action → observe its result → choose the next action.

[37] Yao et al., ICLR 2023, Fig. 1 (cropped).

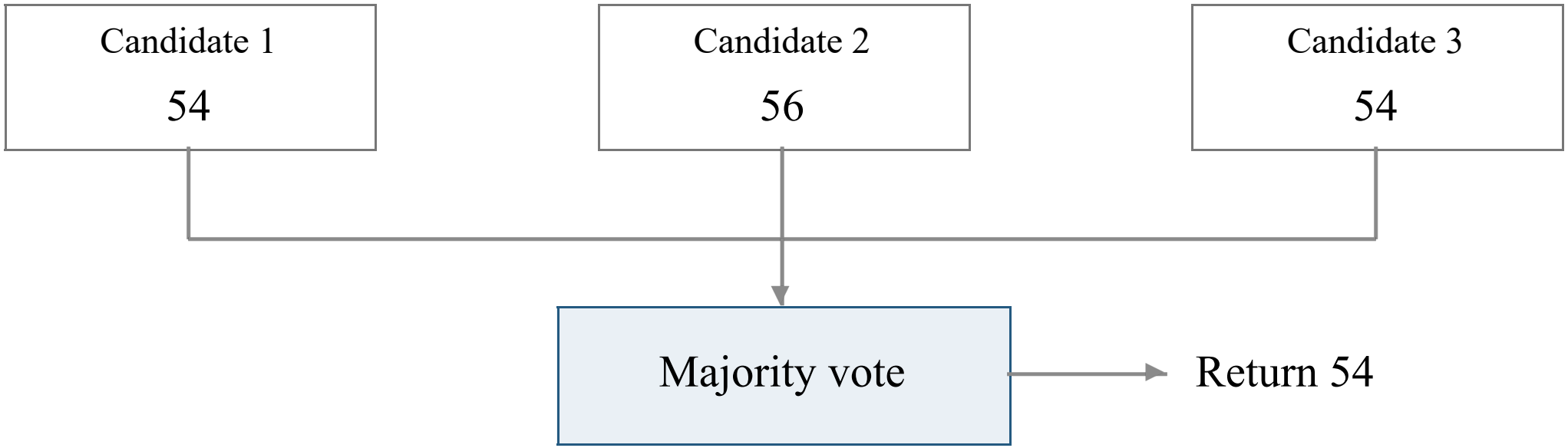
Outline

- Pre-training
  - Tokenization, data curriculum, context extension
  - Scaling laws and mixture-of-experts architectures
- Post-training
  - Supervised fine-tuning (SFT), with PEFT
  - Reinforcement learning (RL)
  - On-policy distillation (OPD)
- Inference
  - KV cache, grouped-query attention, speculative decoding
  - Test-time scaling, verification, and tools
- **Evaluation**
  - Sampling budgets, reliable comparisons, and checkpoint progress

Generate-and-Select Paradigm

Example: three sampled solutions to the same question.

$$7 \times 8 = ?$$



Evaluation answer: 56. The pool contains it; majority voting returns 54.

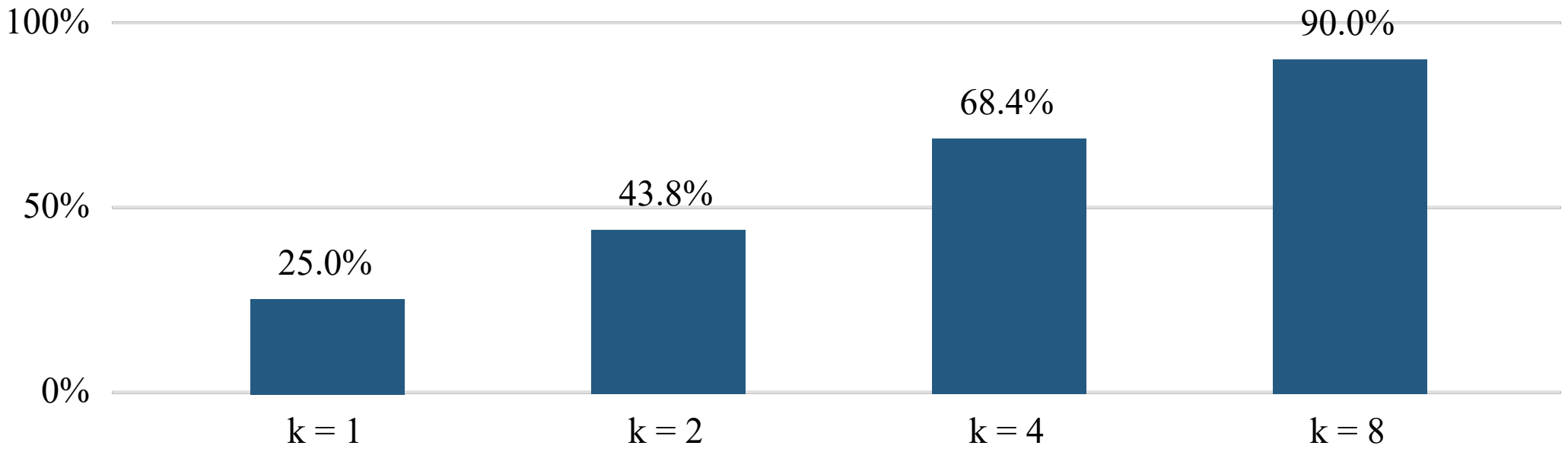
A selector restricted to this pool cannot exceed its any-correct coverage.

[36] Snell et al., ICLR 2025; [38] Chen et al., 2021.

Success Under a Sampling Budget

For one problem, let each independent attempt succeed with probability 0.25.

$$\text{pass@}\, k = 1 - (1 - p)^k$$



Coverage counts any successful candidate.

A deployed selector may still return a wrong candidate.

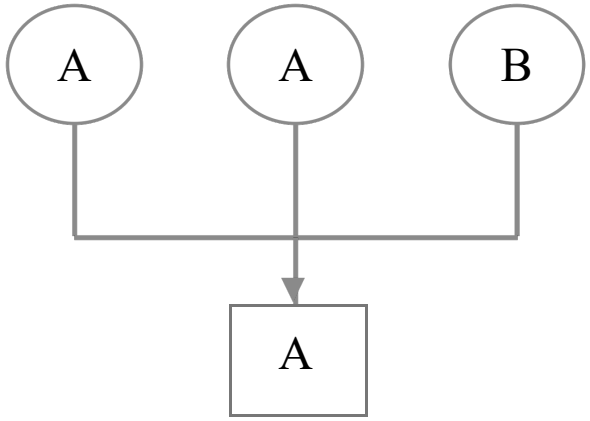
[38] Chen et al., 2021.

Modern Generation-and-Selection Practices

Choose among candidate answers, or use feedback to improve a candidate.

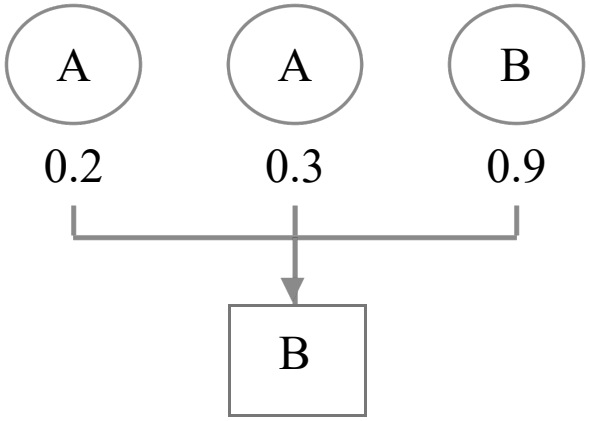
A and B denote different final answers; verifier scores are illustrative.

Majority vote



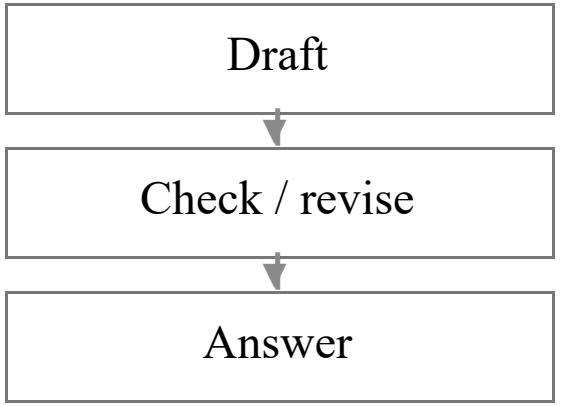
Sample diverse trajectories; vote on their final answers.

Verifier Best-of-N



Score complete responses; keep the highest-scored one.

Verify and refine



Use feedback to revise a draft, then return an answer.

Final success depends on the generator, selector, and compute budget.

[26] Shao et al., DeepSeekMath, 2024; [36] Snell et al., ICLR 2025.

Reliable Evaluation

Keep evaluation data separate and compare at a reported compute budget.

Training	Development	Final test
Fit model parameters.	Choose the recipe.	Evaluate fixed choices.

Leakage

Hold final prompts and solutions out of training and recipe selection.

Matched comparisons

Report starting checkpoint, token budget, candidate count and tool costs.

Measure accuracy together with latency and memory.

[39] Team Olmo, 2025; revised 2026; [36] Snell et al., ICLR 2025.

Reward and Generalization

Two programs can earn the same reward on an incomplete checker.

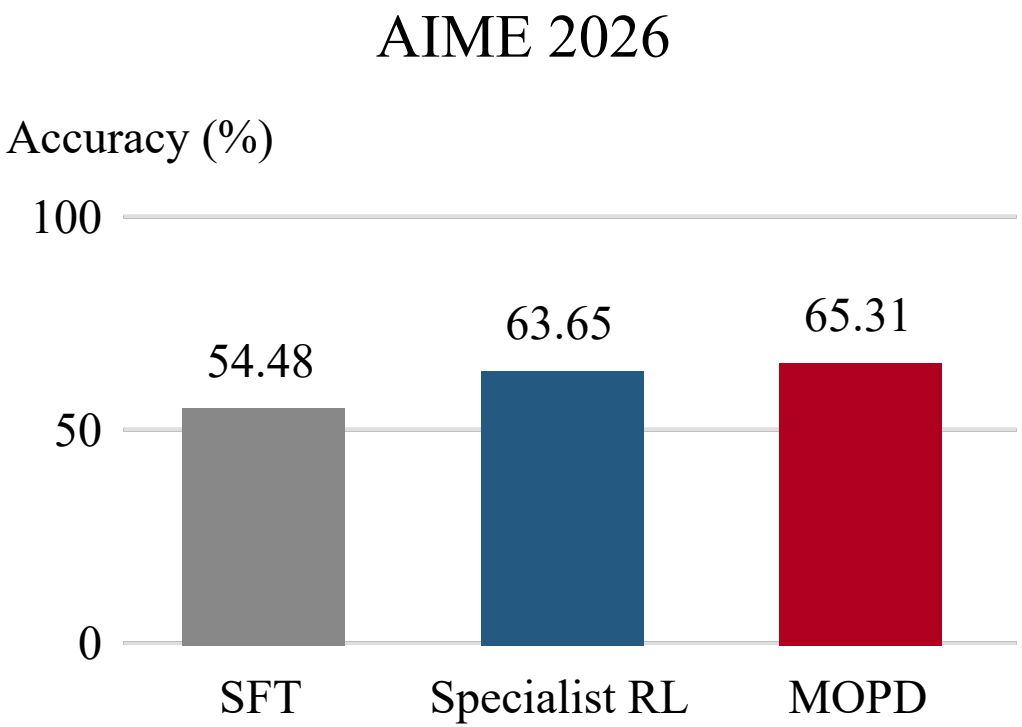
Task: return the larger integer for arbitrary inputs.

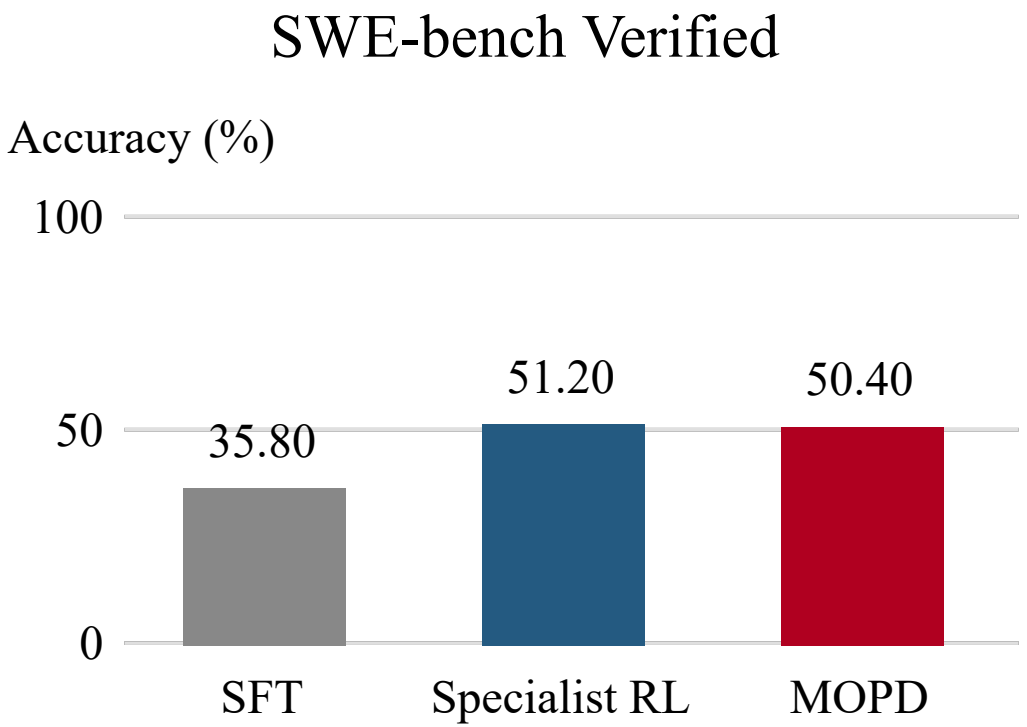
return 2	return max(a, b)
Visible tests	Visible tests
$(1, 2) \mapsto 2, \quad (2, 1) \mapsto 2$	$(1, 2) \mapsto 2, \quad (2, 1) \mapsto 2$
Both pass.	Both pass.
Held-out test: $(7, 3) \mapsto 7$	
Fails	Passes

[39] Team Olmo, 2025; revised 2026.

Evaluation Across Training Stages

One reported Qwen3-30B-A3B experiment [40, Table 2]. Higher is better.



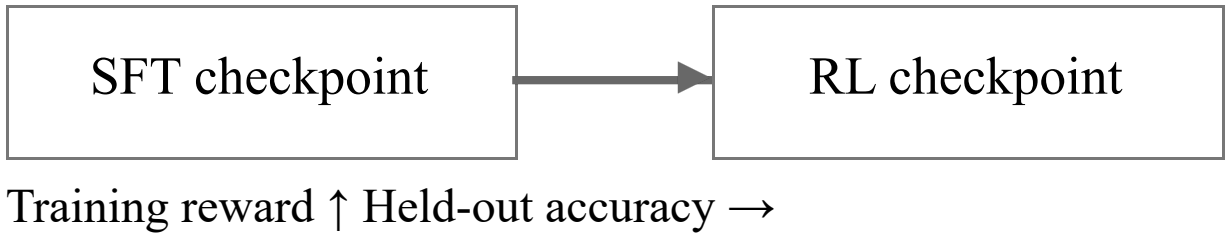


Check each skill for gains and regressions after consolidation.

[40] Ma et al., 2026, Table 2.

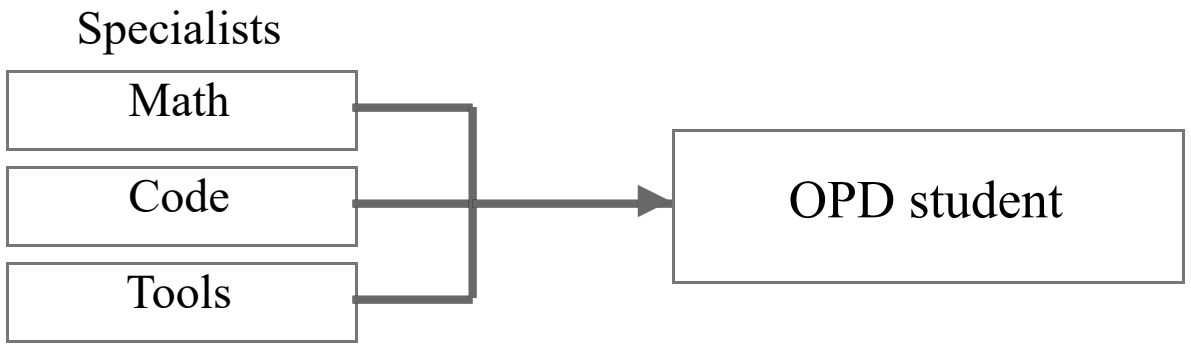
Questions and Discussion

1. Reward rises after RL, but held-out accuracy stays flat. What would you investigate?



Check reward exploitation, prompt overfitting and distribution shift with independent held-out tests.

2. OPD improves the aggregate score. How would you test whether each specialist's capabilities are preserved?



Compare teacher and student per domain, at matched sampling, token and tool budgets.

References

[1] Hugging Face LLM Course. Tokenizers.

[2] Sennrich et al. (2016). Neural Machine Translation of Rare Words with Subword Units. ACL.

[3] OpenAI. tiktoken educational implementation: byte-level BPE training and encoding.

[4] Hugging Face LLM Course. Byte-Pair Encoding Tokenization. Chapter 6.5.

[5] F. Yergeau. UTF-8, a transformation format of ISO 10646. RFC 3629, 2003.

[6] Unicode Consortium. Unicode Standard Annex #15: Unicode Normalization Forms.

[7] Qwen Team (2025). Qwen3 Technical Report.

[8] OpenAI. tiktoken: model-to-encoding mappings for GPT model names.

[9] Wu et al. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

[10] Devlin et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL.

[11] Google. SentencePiece project documentation and pretrained Unigram models, including T5-base.

References

[12] T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. EMNLP System Demonstrations, 2018.

[13] T. Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. ACL, 2018.

[14] OpenAI. tiktoken: public encoding definitions, vocabularies and pre-tokenization patterns.

[15] Anthropic. Legacy TypeScript tokenizer utility and warning about Claude 3 and later.

[16] Anthropic. Claude token counting: model-specific input-token counts.

[17] MiMo Team (2026). MiMo-V2-Flash Technical Report.

[18] Kaplan et al. (2020). Scaling Laws for Neural Language Models.

[19] Brown et al. (2020). Language Models are Few-Shot Learners. NeurIPS.

[20] Fedus et al. (2022). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. JMLR.

[21] Ouyang et al. (2022). Training Language Models to Follow Instructions with Human Feedback. NeurIPS.

References

[22] Hu et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models. ICLR.

[23] Dettmers et al. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. NeurIPS.

[24] Schulman et al. (2017). Proximal Policy Optimization Algorithms.

[25] Hugging Face (2025). TRL PPO Trainer documentation, v0.24.0. PPOConfig: gamma = 1.0.

[26] Shao et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models.

[27] Yu et al. (2025). DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv:2503.14476.

[28] Rafailov et al. (2023). Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. NeurIPS.

[29] Thinking Machines Lab (2025). On-Policy Distillation.

[30] Alvarez et al. (2025). Optimizing Inference for Long Context and Large Batch Sizes with NVFP4 KV Cache. NVIDIA.

References

[31] Hugging Face (2025). Continuous batching.

[32] Ainslie et al. (2023). GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. EMNLP.

[33] NVIDIA (2025). An Introduction to Speculative Decoding for Reducing Latency in AI Inference.

[34] Leviathan et al. (2023). Fast Inference from Transformers via Speculative Decoding. ICML.

[35] DeepSeek-AI (2024). DeepSeek-V3 Technical Report.

[36] Snell et al. (2025). Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Model Parameters. ICLR.

[37] Yao et al. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. ICLR.

[38] Chen et al. (2021). Evaluating Large Language Models Trained on Code.

[39] Team Olmo (2025). Olmo 3.

[40] Ma et al. (2026). MOPD: Multi-Teacher On-Policy Distillation for Capability Integration in LLM Post-Training.