DeepLearning.AI

Production Considerations

A production post-training pipeline

Frontier lab pipelines

R1 & R1-Zero were first examples of open-source “reasoning” models

“DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning” [Jan 2025]

Deepseek R1 pipeline

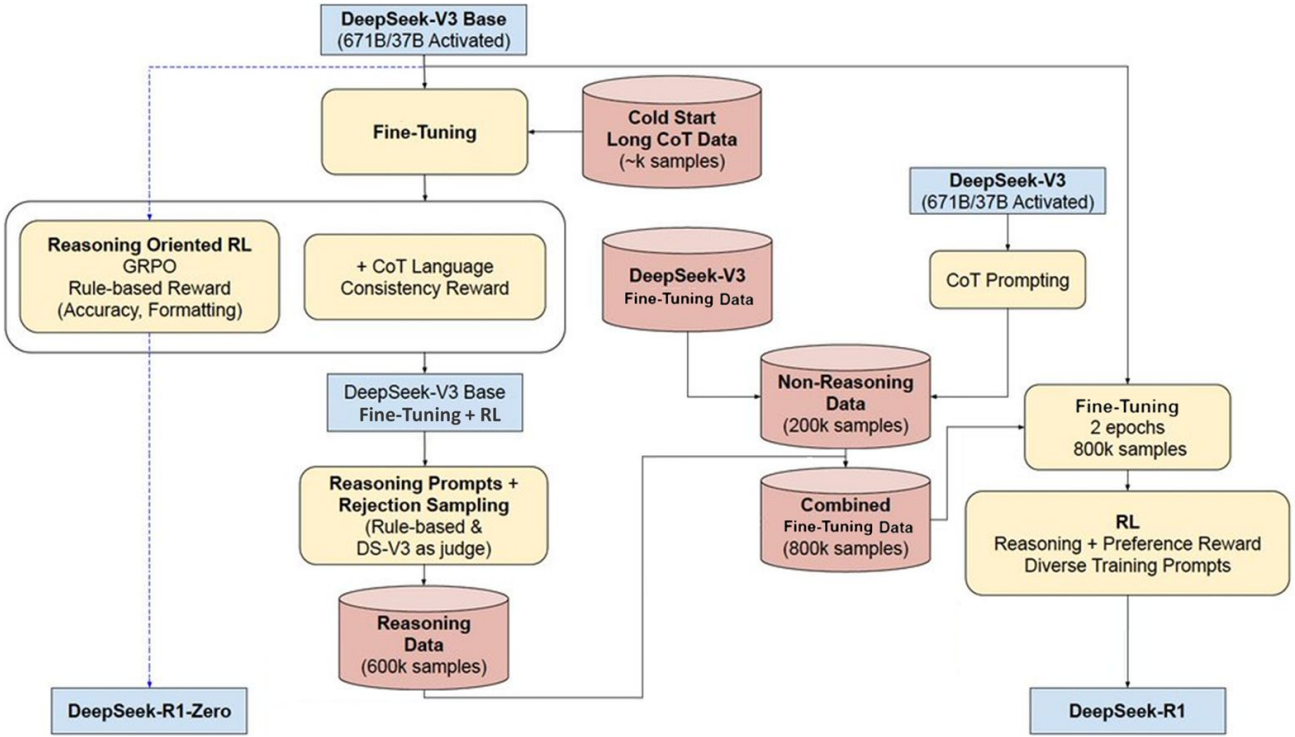


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

2 final models

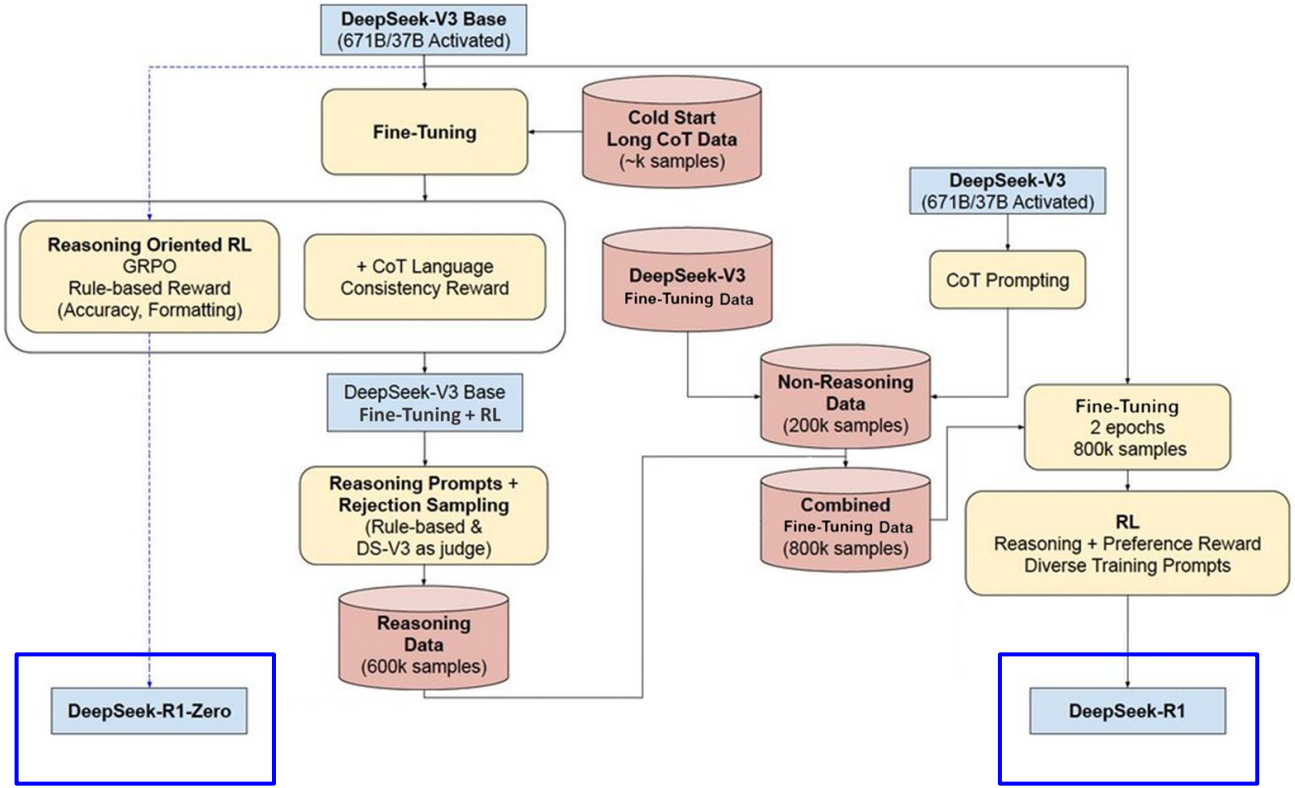


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Intermediate models

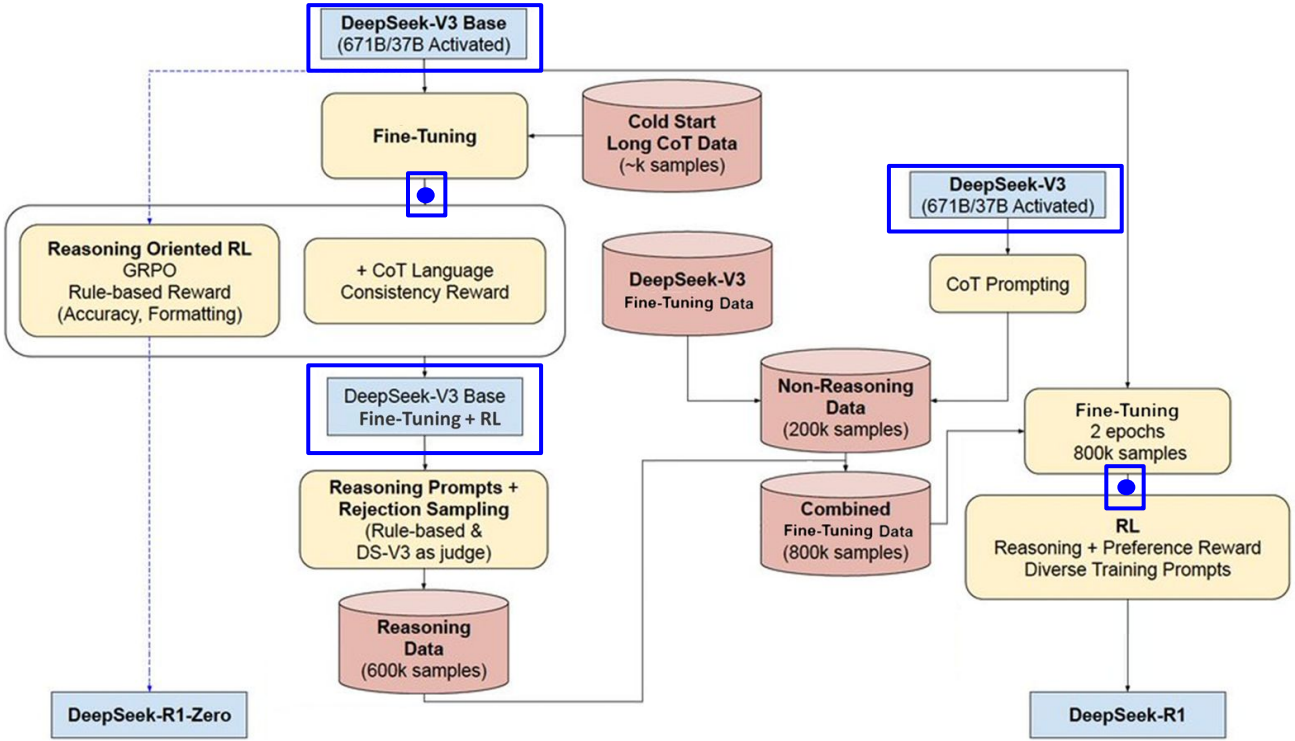


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Intermediate datasets

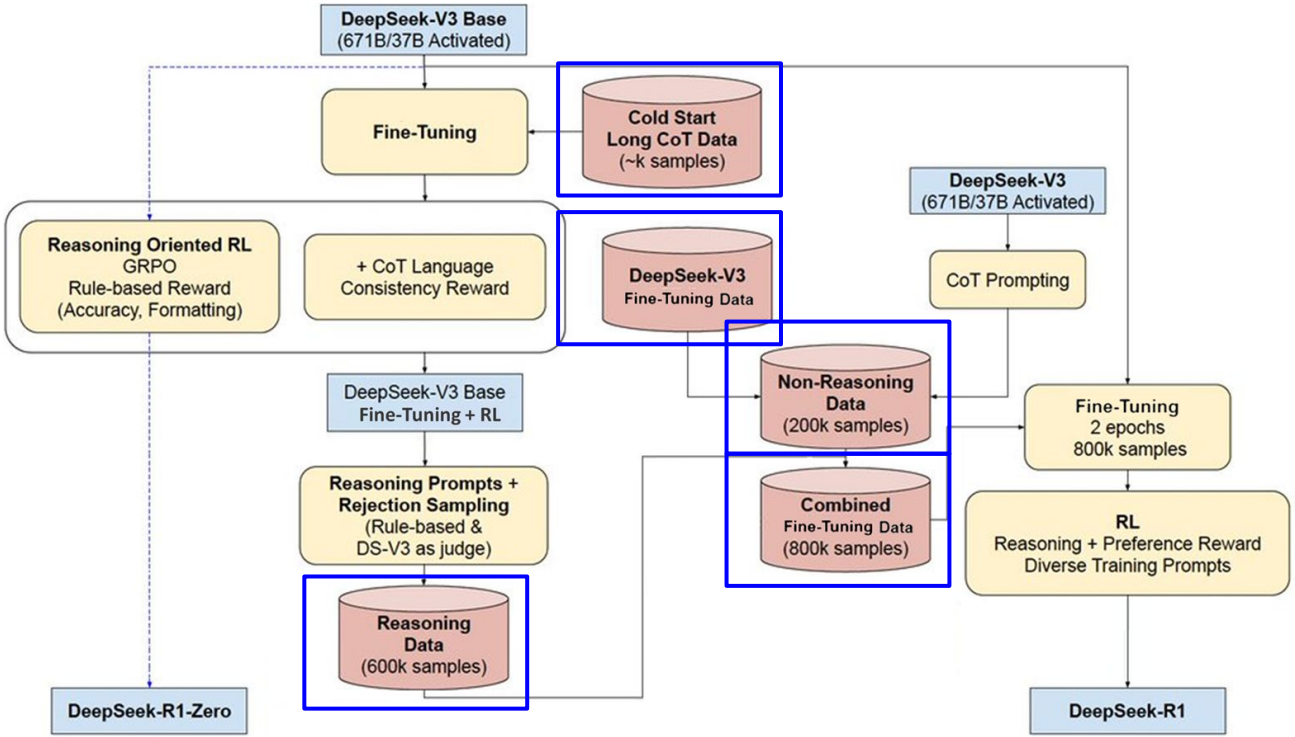


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Post-training stages

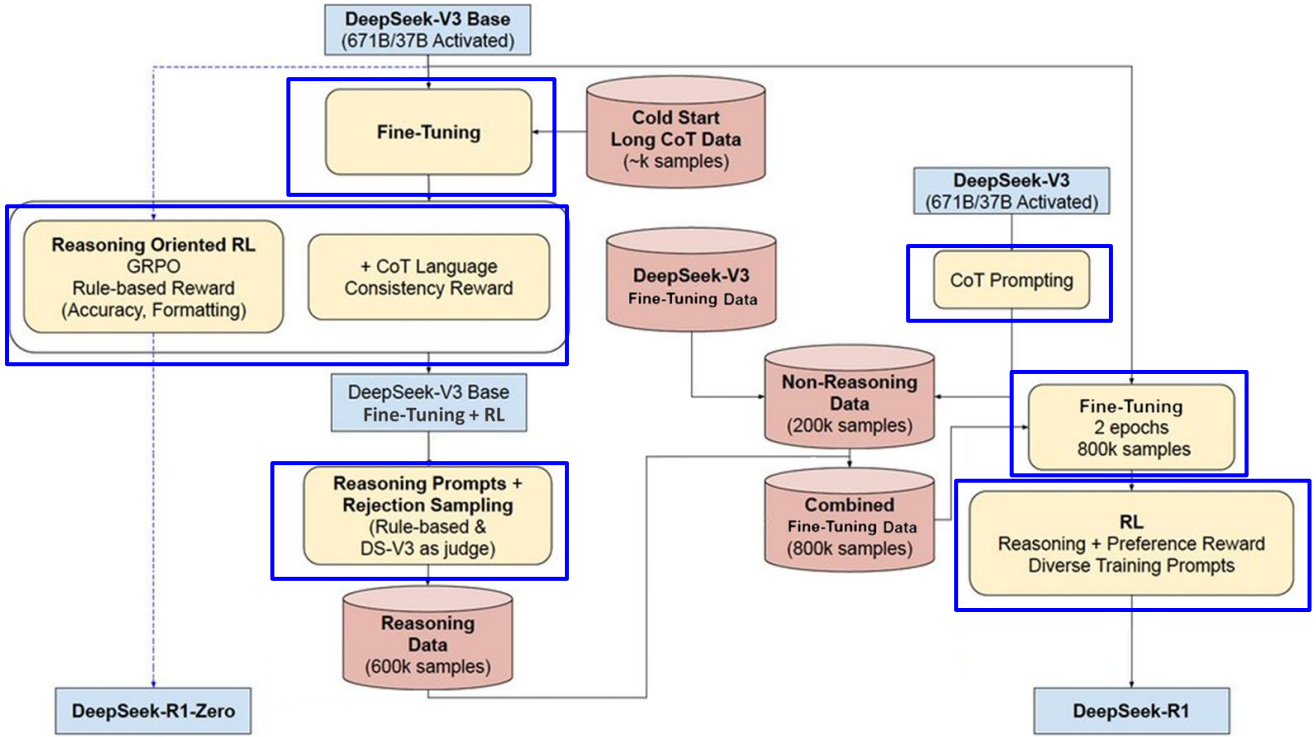


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

R1-Zero pipeline: Simple, only RL with verifiers

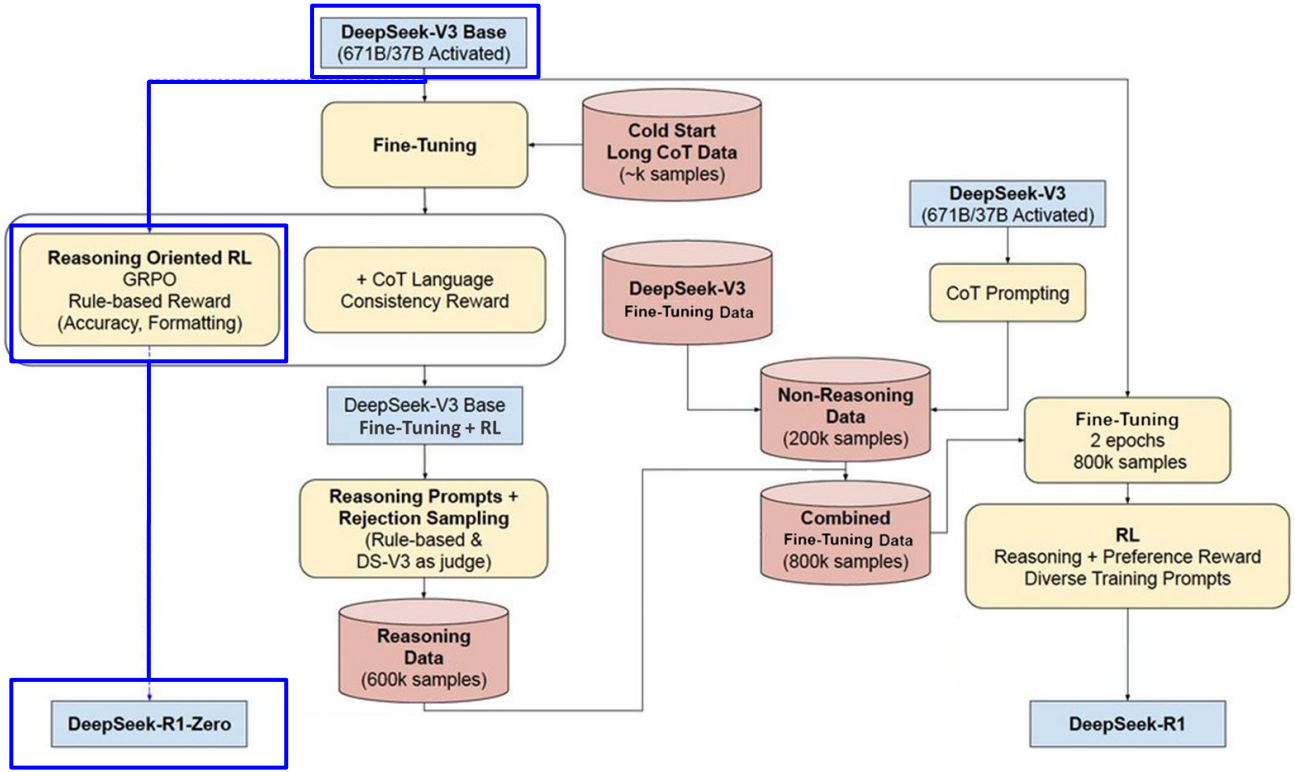
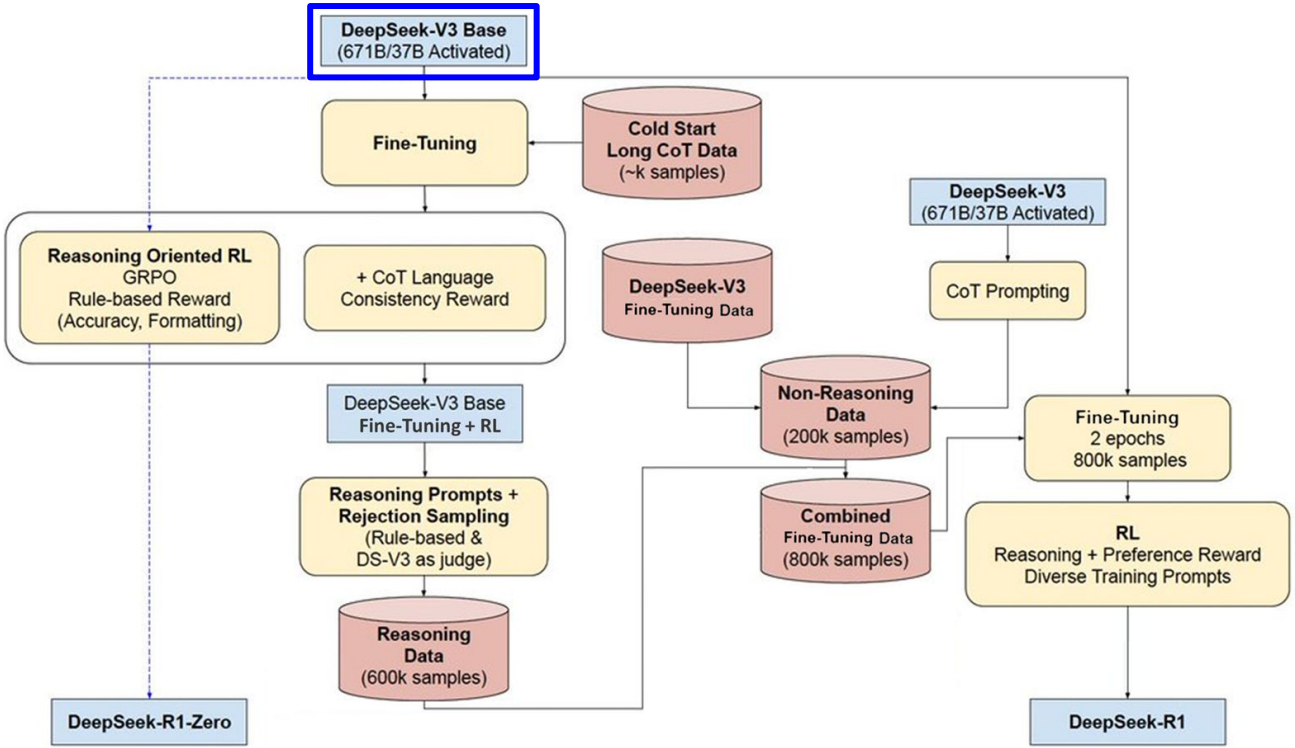


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

R1-Zero pipeline: Simple, only RL with verifiers



DeepLearning.AI

Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

AMD

Deepseek-v3 base

Pre-trained on 14.8 Trillion tokens.

"Only 180,000 GPU hours". Only $5.3M.

Benchmark (Metric)	# Shots	DeepSeek-V2 Base	Qwen2.5 72B Base	LLaMA-3.1 405B Base	DeepSeek-V3 Base
Architecture	-	MoE	Dense	Dense	MoE
# Activated Params	-	21B	72B	405B	37B
# Total Params	-	236B	72B	405B	671B
Pile-test (BPB)	-	0.606	0.638	**0.542**	0.548
BBH (EM)	3-shot	78.8	79.8	82.9	**87.5**
MMLU (EM)	5-shot	78.4	85.0	84.4	**87.1**
MMLU-Redux (EM)	5-shot	75.6	83.2	81.3	**86.2**
MMLU-Pro (EM)	5-shot	51.4	58.3	52.8	**64.4**
DROP (F1)	3-shot	80.4	80.6	86.0	**89.0**
ARC-Easy (EM)	25-shot	97.6	98.4	98.4	**98.9**
ARC-Challenge (EM)	25-shot	92.2	94.5	**95.3**	**95.3**

[From "DeepSeek-V3 Technical Report", DeepSeek-AI, 2024]

R1-Zero pipeline: RL stage

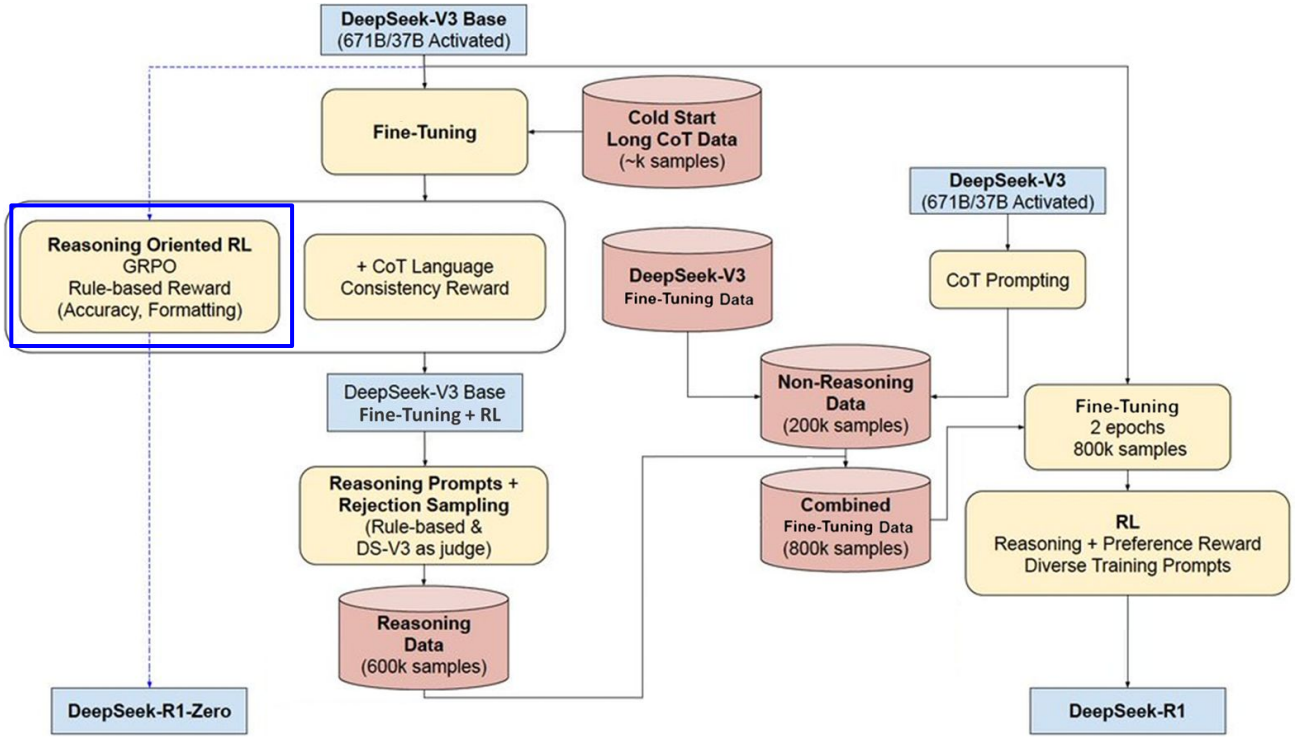
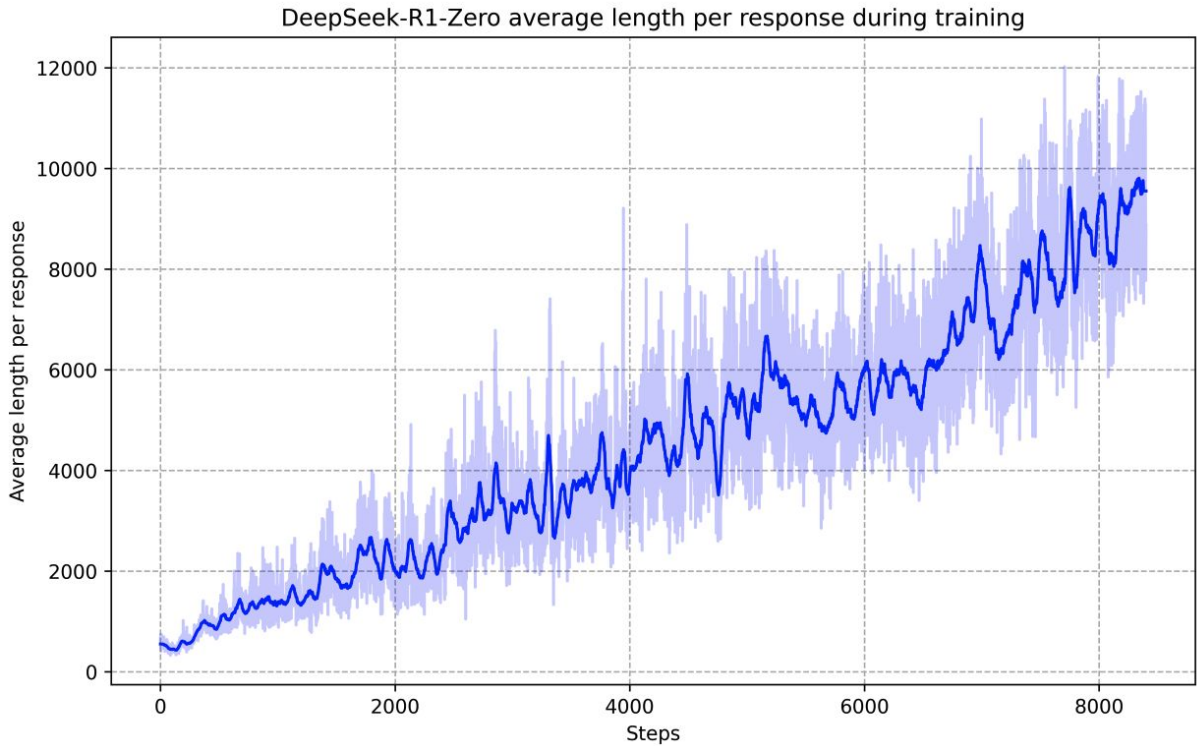


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

RL for reasoning

- Train data: math & coding
- GRPO with 2 verifiers:
  - Accuracy
  - Format
- From 15.6% to 86.7% on AIME (math)
  - Competitive with OpenAI's o1 model



[From "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning", DeepSeek-AI, 2025]

RL for reasoning

"Behaviors such as reflection... the exploration of alternative approaches to problem-solving arise spontaneously."

RL for reasoning

"Behaviors such as reflection... the exploration of alternative approaches to problem-solving arise spontaneously."

"DeepSeek-R1-Zero struggles with challenges like poor readability, and language mixing."

Time to upgrade from R1-Zero to R1!

R1 pipeline: General purpose, not just reasoning

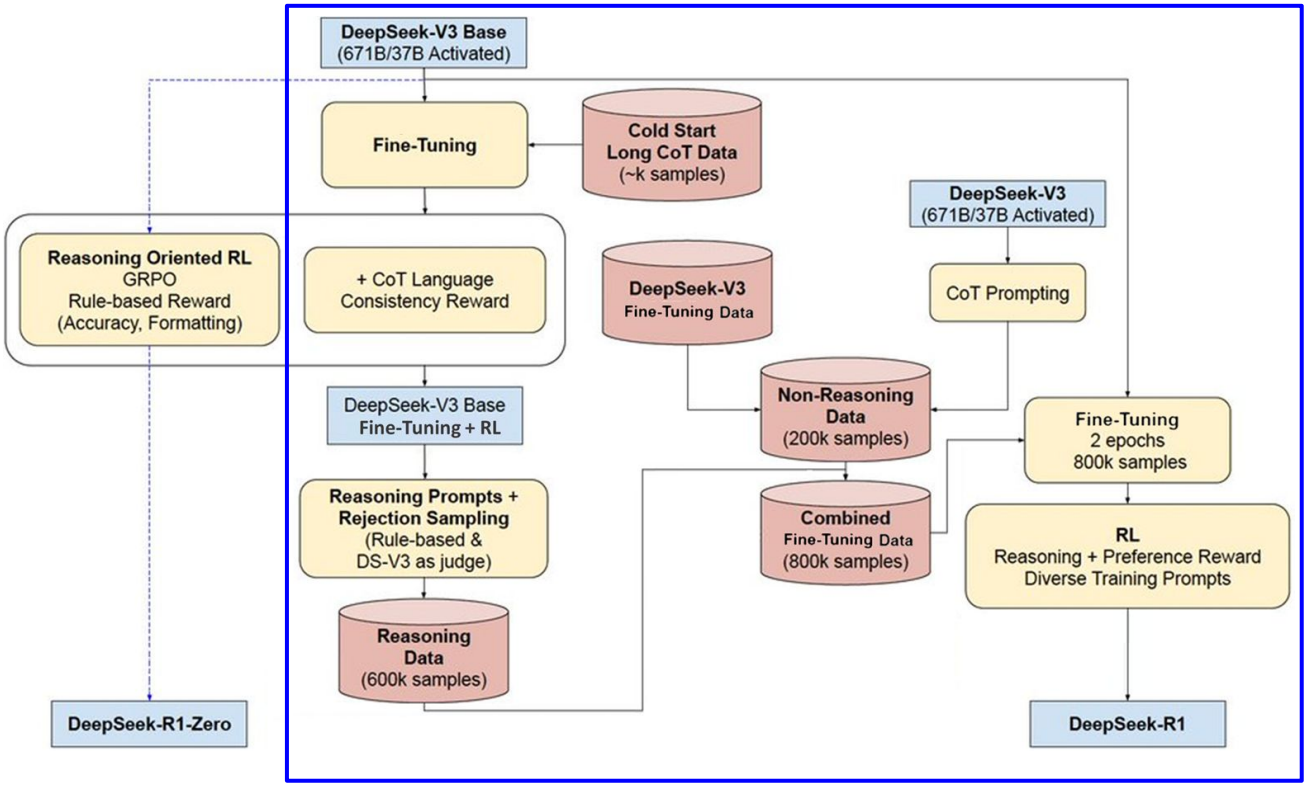


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

R1: Generating reasoning data

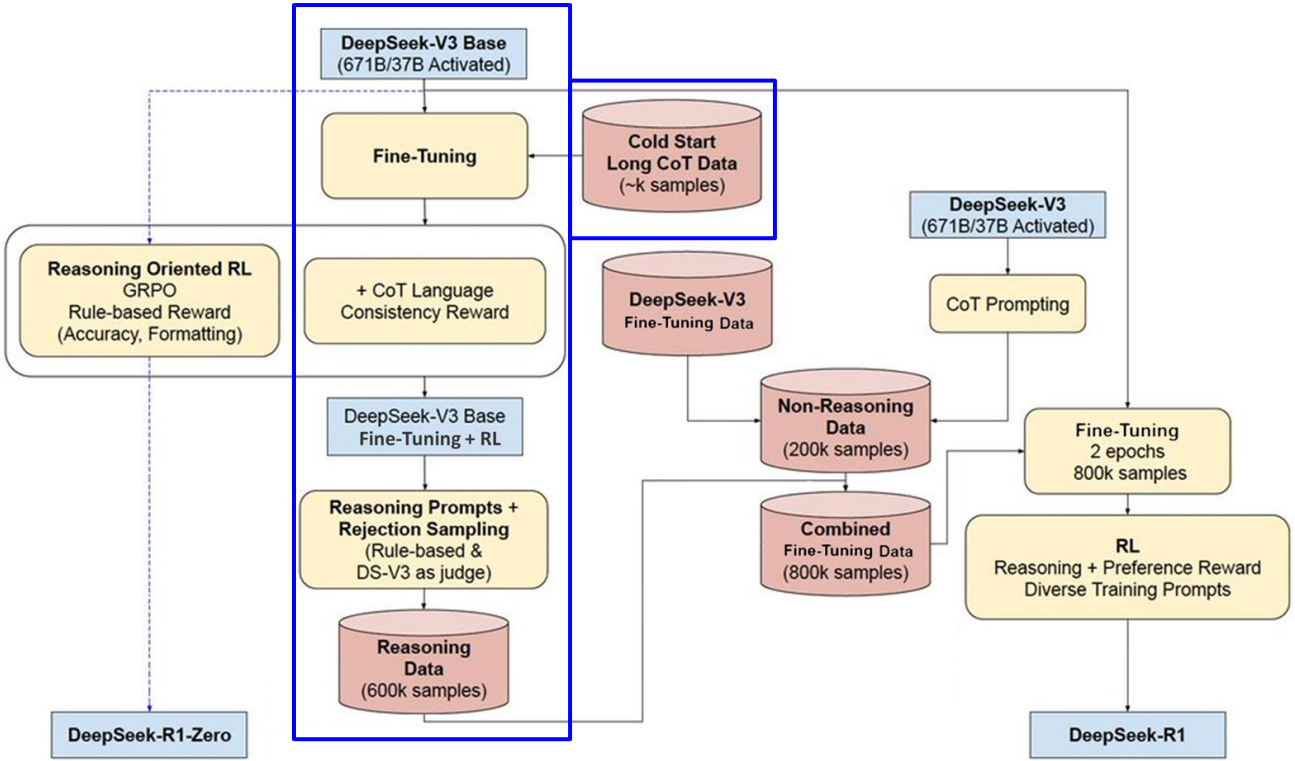


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Generating reasoning data

Input

Alice has 3 apples and buys 2 more. **How many now?**

Target Output

<think>
Start with 3.
Buying more things should be additive.
If so, then 2 ⇒ 3+2
...
</think>

~k examples = Small "seed" dataset

Long: very detailed reasoning

"Cold-start": bootstraps reasoning style and format.

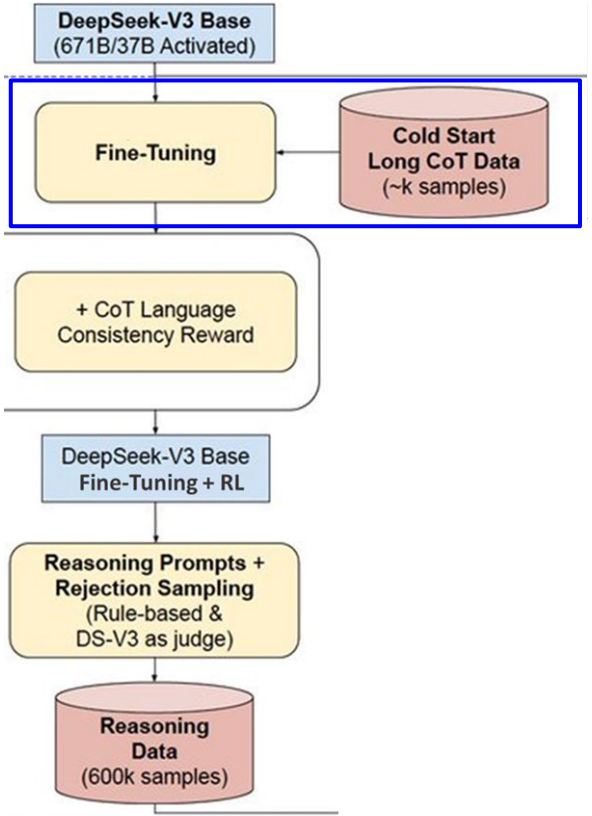


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Generating reasoning data

Intermediate fine-tuned model

Now to be trained with RL!

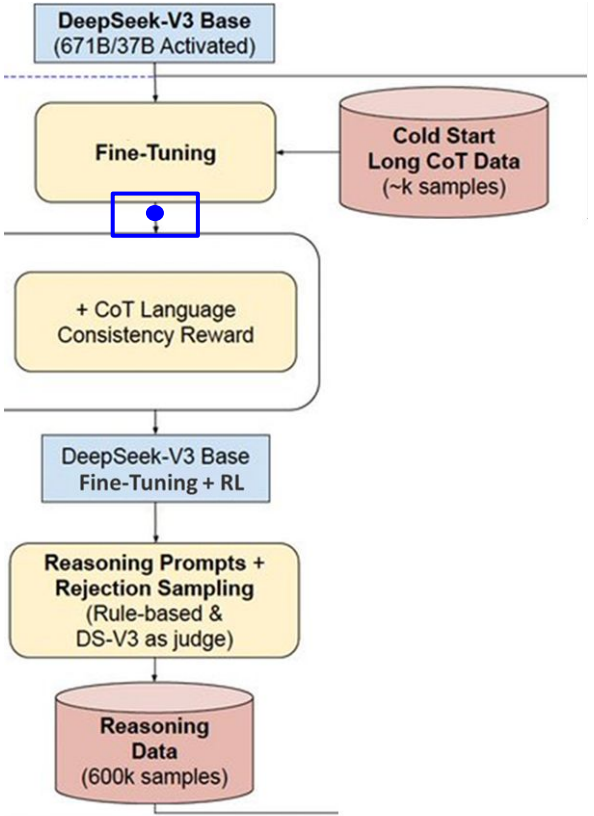
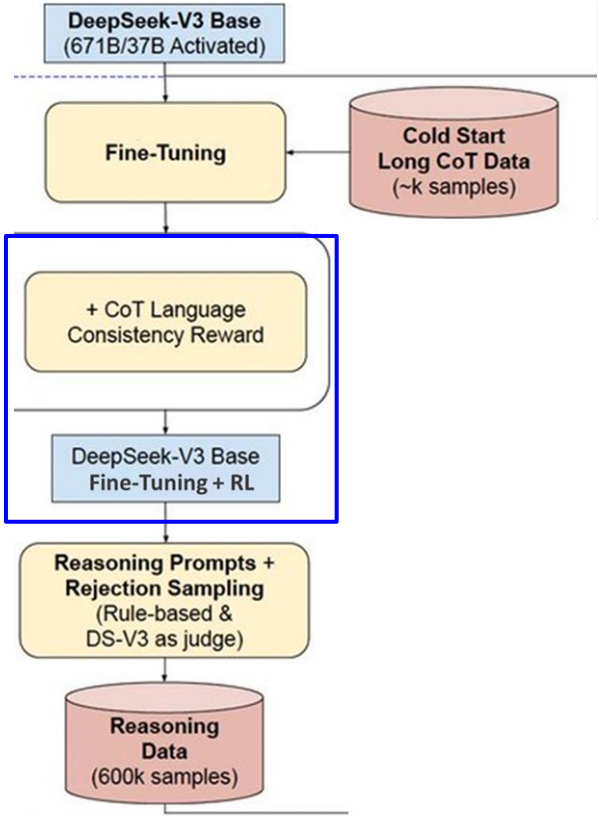
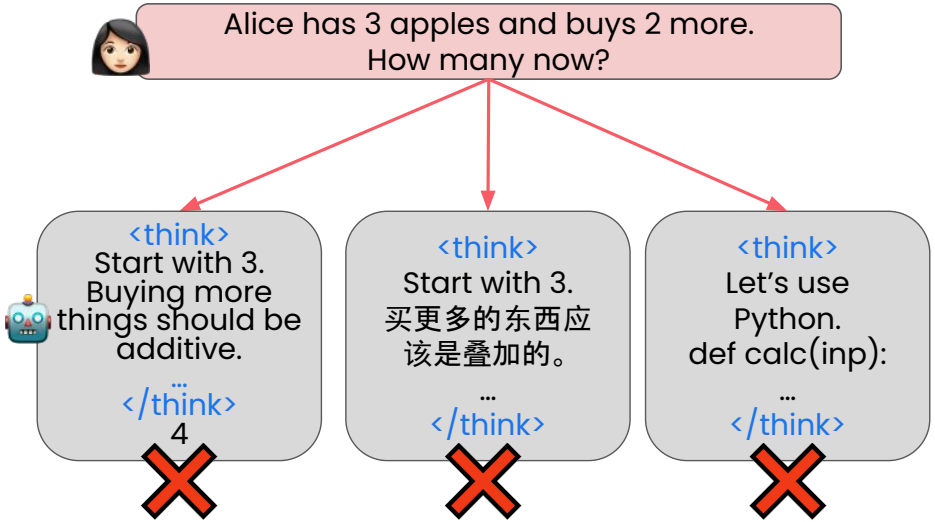


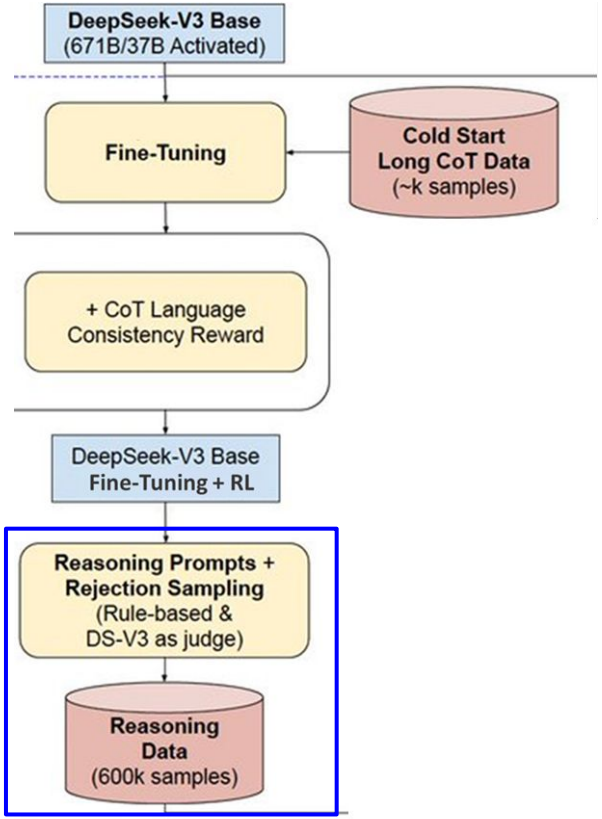
Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Generating reasoning data



Generating reasoning data





R1: Generating non-reasoning data

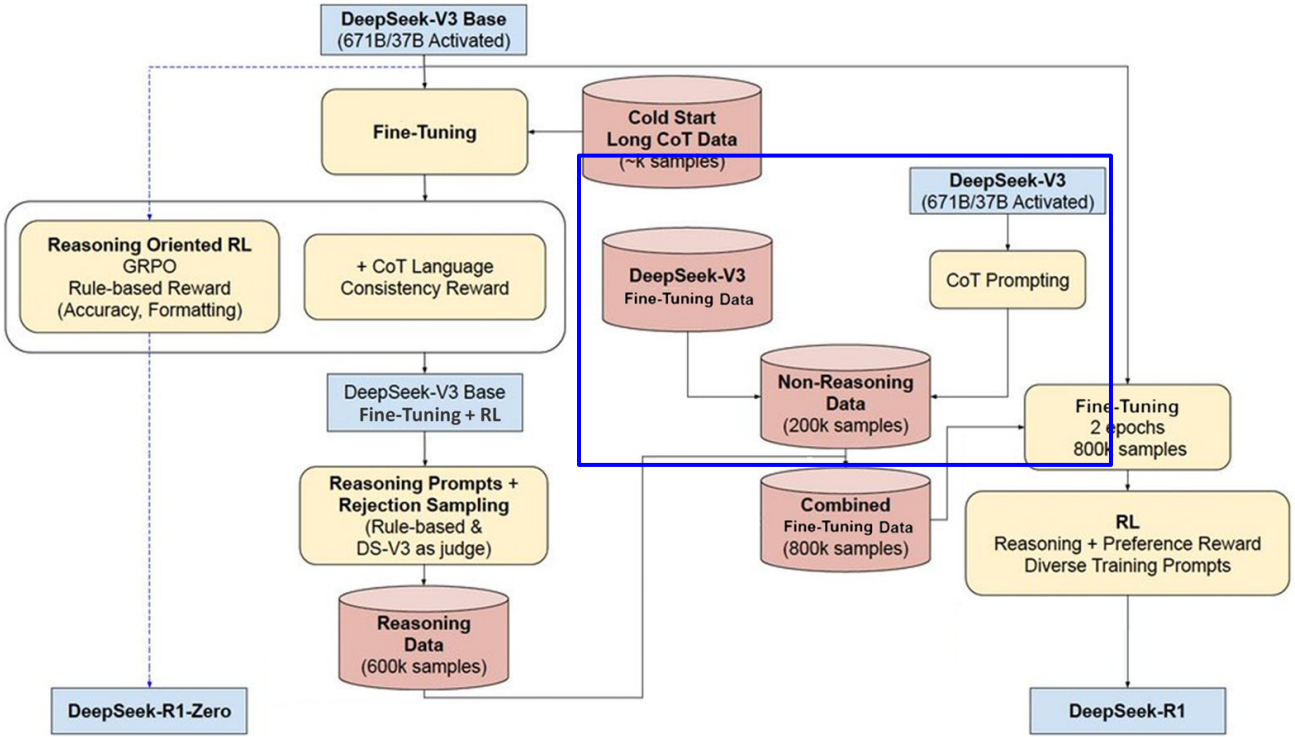


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Generating non-reasoning data

Input

Write a story about a 1800s whale hunting

Model Output

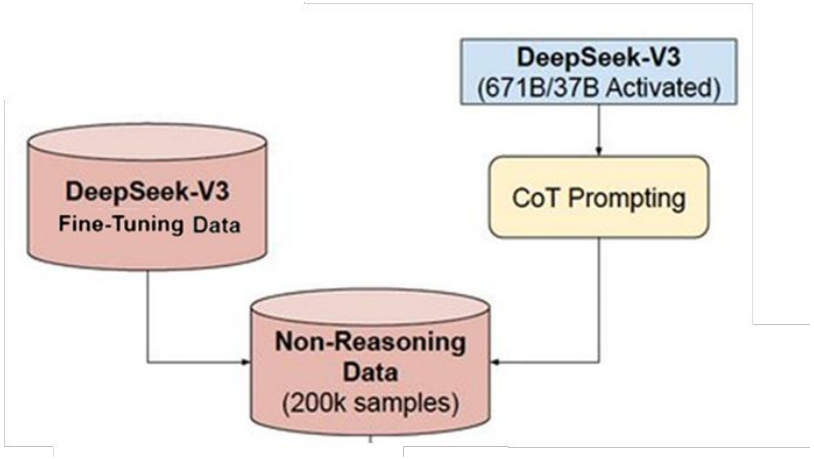
There are not many written records, but Moby Dick could serve as some inspiration. It was written before the Civil War, so...

...

Here is a story about captain Ahab...

Target Output

Here is a story about captain Ahab...



Generating non-reasoning data

Input

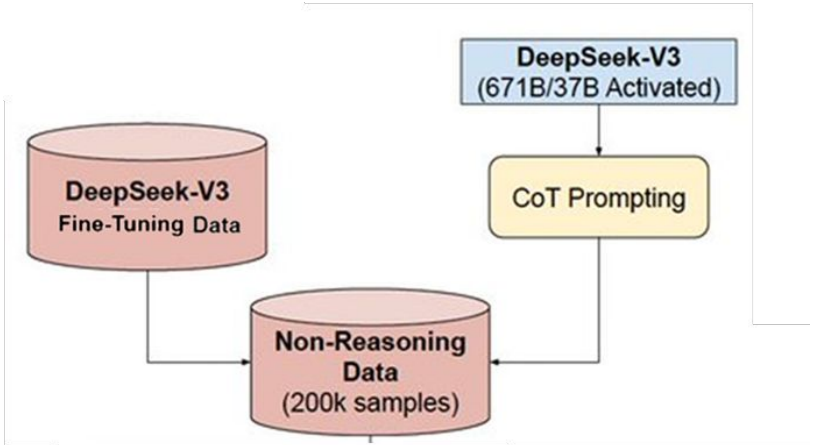
Hello!

Model Output

Hi! Happy to be of assistance.

Target Output

Hi! Happy to be of assistance.



R1: Post-training

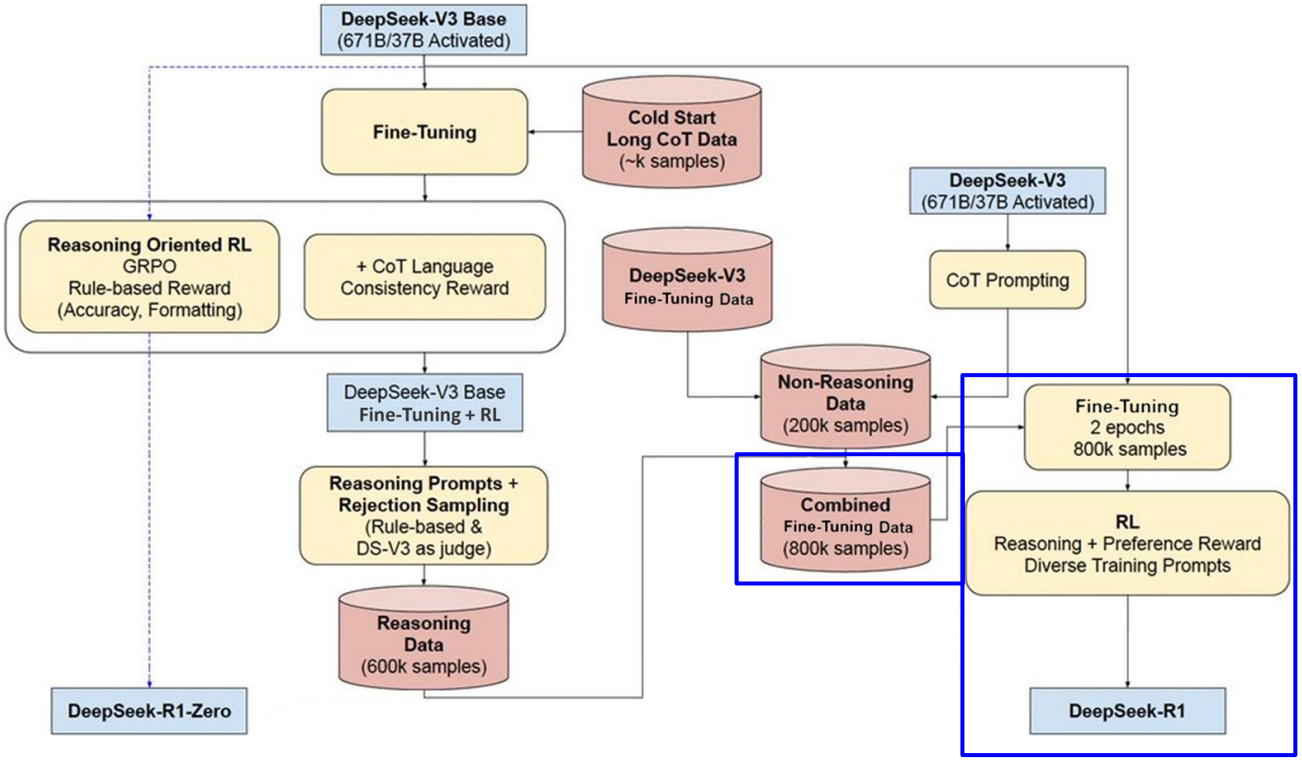
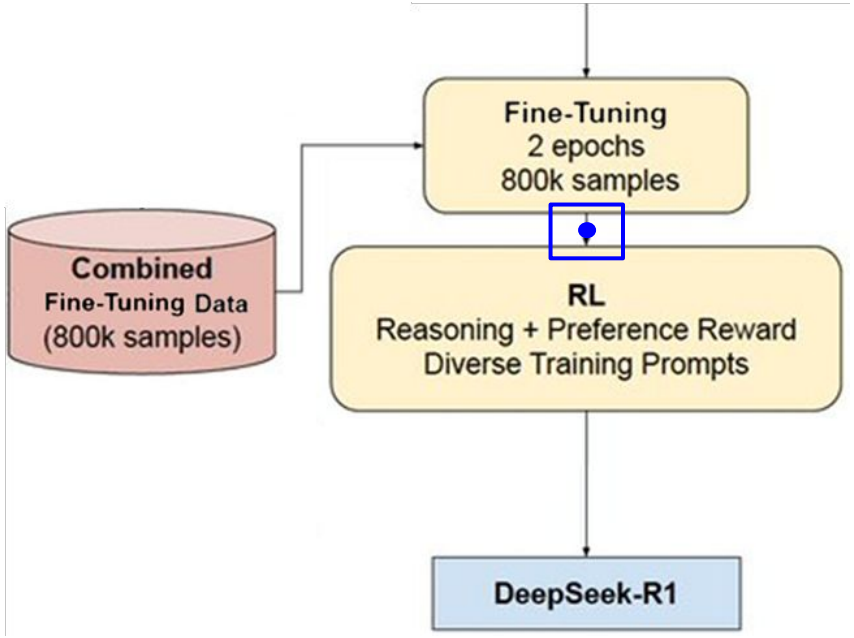


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

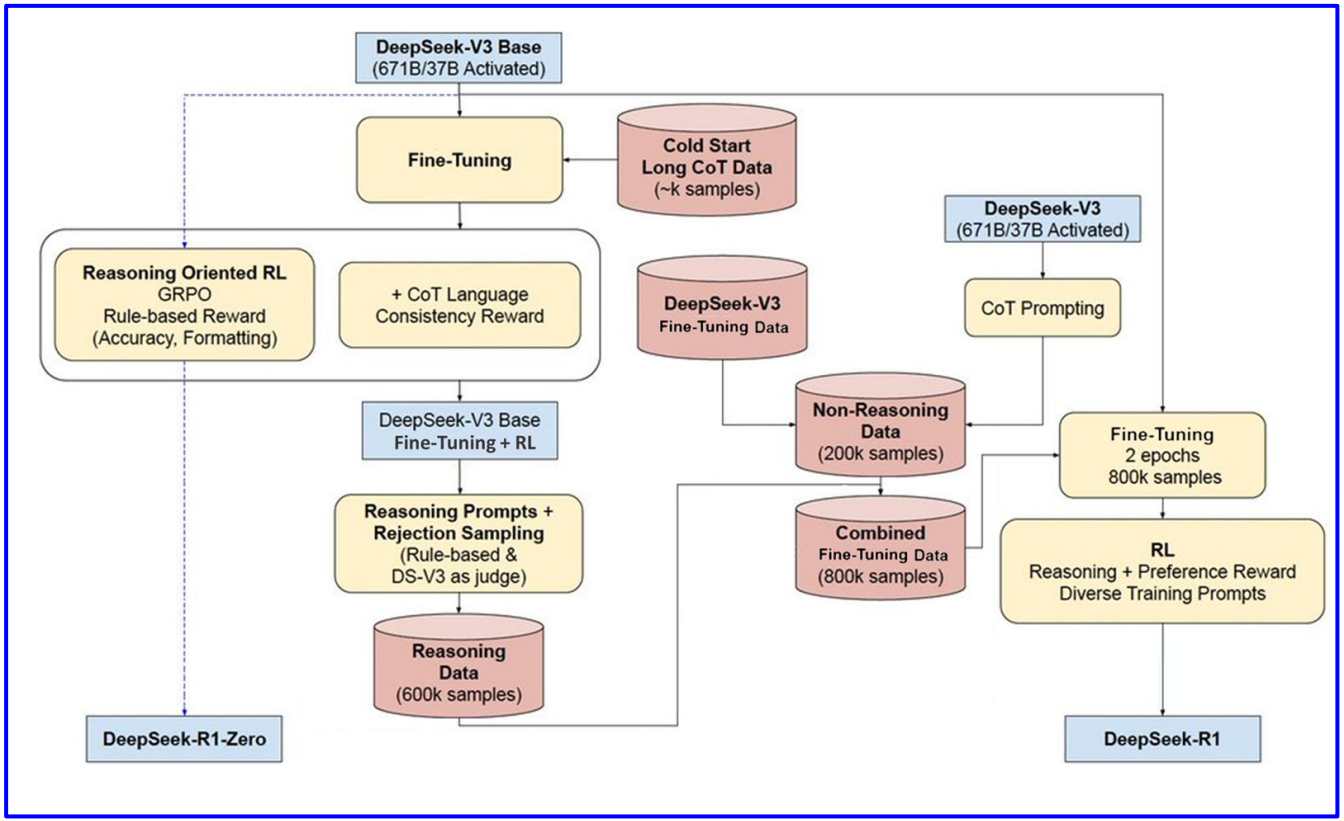
R1: Post-training

- Fine-tune for 2 epochs on synthetic data (800k)
- Train fine-tuned model with RL

Reasoning & non-reasoning data



Deepseek-R1-Zero and -R1 pipelines



DeepLearning.AI

Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

AMD

DeepLearning.AI

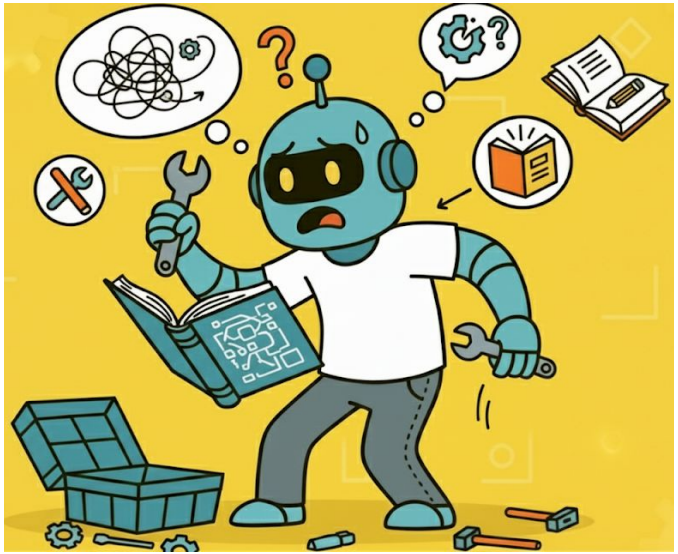
Production Considerations

Agents

A clumsy apprentice, not an agent!

📚 Assistant: learn to chat with post-training

🛠️🔑 Agent: learn to use tools, plan, and coordinate with different post-training



How does post-training create agents?

Chatbots

More post-training

Agents

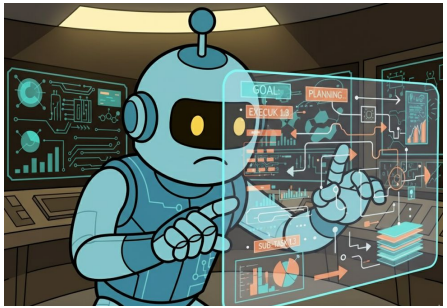
- Respond to queries
- Holds great conversation
- Can handle chat history

- Tool & API use
- Strategic reasoning & multi-step planning
- Fewer hallucinations, loops, and brittle execution.

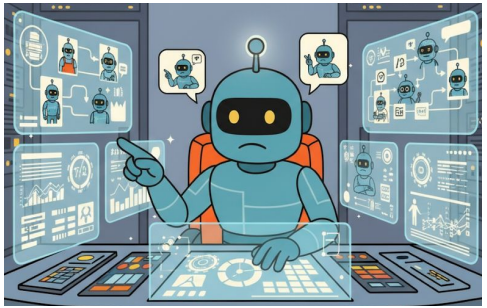
Agentic behavior

Tool Use

Planning



Coordination



Fine-tuning for tool use

Input

What is today's weather?

Target Output

<datetime tool <get_current_date()>
<weather API <today>>
<display_weather>☀️</display_weather>

RL for tool use

Input

Alice has 3 apples and buys 2 more.
How many now?

Model Output

<Use calculator <3+2=>>
<answer>5</answer>

Reward

+1

RL for tool use

Input

Debug this issue for me...

Model Output

Looking at my_codebase, and getting the latest with the search_api...

Reward

Correct: **+1**

Up to date: **+1**

***Total reward (score)*: +2**

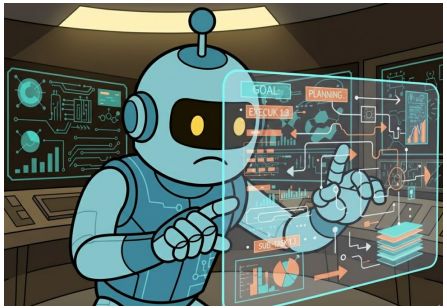
search_api Tools

my_codebase Files

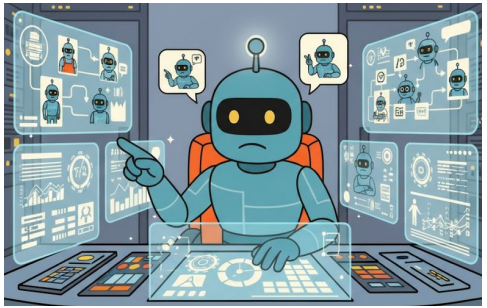
Agentic behavior

Tool Use

Planning



Coordination



Fine-tuning for reasoning

Input

Alice has 3 apples and buys 2 more.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>

RL for reasoning

Input

Alice has 3 apples and buys 2 more.
How many now?

Model Output

<think>
Start with 3.
Buys 2 ⇒ 3+2.
</think>

<answer>5</answer>

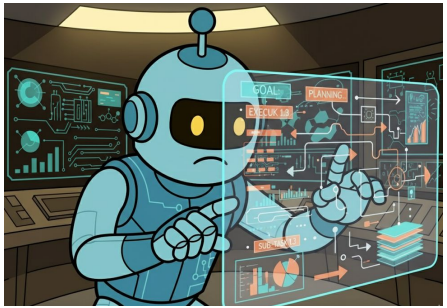
Reward

+1

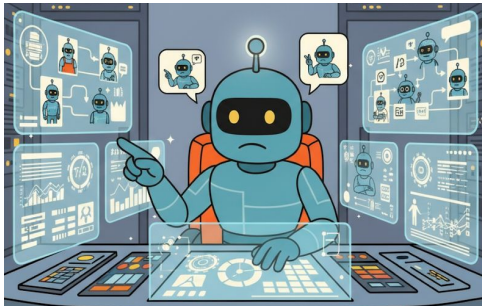
Agentic behavior

Tool Use

Planning



Coordination



Fine-tuning for coordination

Input

Alice has 3 apples and buys 2 more.
How many now?
<Agent A>Start with 3. Buys 2 ⇒ 3+2</Agent A>
<Agent B><calculator tool>3+2=5</Agent B>

Target Output

<think>
Agent A turns the word problem into an equation, hands off to Agent B who can use a calculator.
</think>

<answer>5</answer>

RL for coordination

Input

Find refund status for Order #123...

Model Output

<Agent A>Refund pending.</Agent A>

<answer>Didn't get order info.</answer>

Grader

Missing info handoff: **-1**
Wrong final response: **-1**
Total reward: **-2**

RL for coordination

Input

Refund Order #123...

Model Output

<Agent A>Refund pending (error no refund found).</Agent A>
<answer>Refunded.</answer>

Grader

Missing info handoff: **-1**
Wrong final response: **-1**
Total reward: **-2**

RL for coordination

Input

Refund Order #123...

Model Output

<Agent A>Order #123 belongs to customer 42.</Agent A>

<answer> Received order #123; refund complete ✅ </answer>

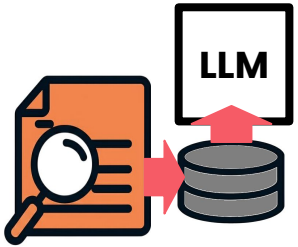
Grader

Info passed correctly: **+1**
Role execution correct: **+1**
Total reward: **+2**

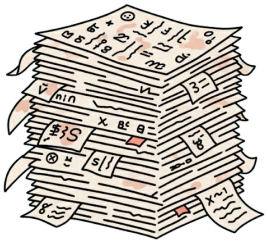
Live agents

Updated state

New context/RAG*



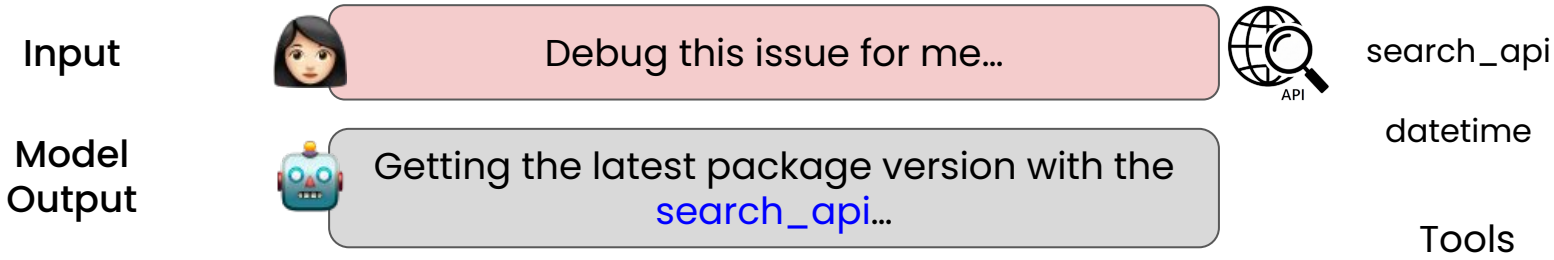
Messy, wrong data



*RAG: Retrieval Augmented Generation

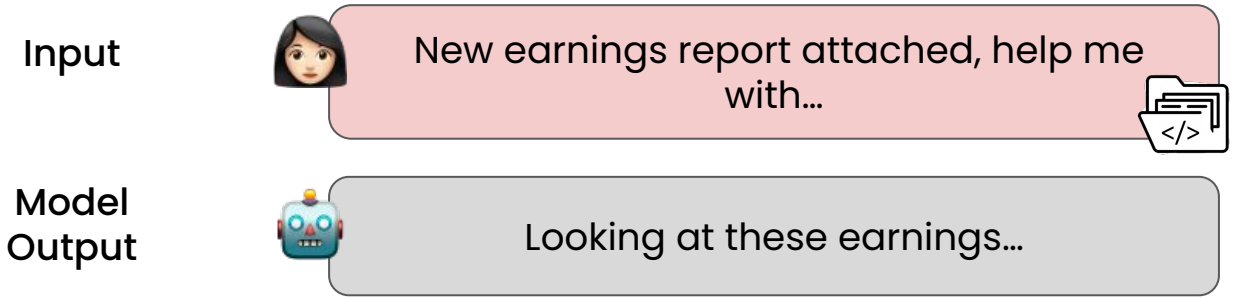
(data search to add relevant info in input)

Continually updating tools



Learn to use tools for updated state, instead of internal frozen state

RAG retrieves newly seen information



Learn to handle never-seen information

Wrong information passed in

Input

New earnings report attached, help me with...

Model Output

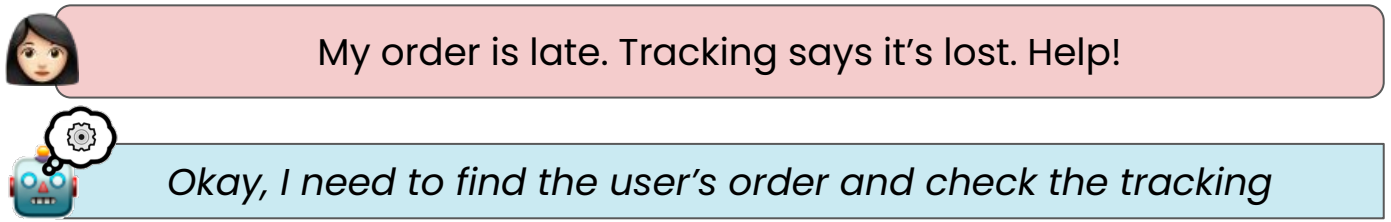
From today's date, and the attached file, these are not the latest earnings...

Learn to handle incorrectly provided user information

Live agent coordination: Case study

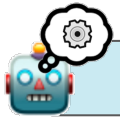
My order is late. Tracking says it’s lost. Help!

Live agent coordination: Case study

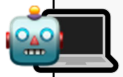


Live agent coordination: Case study

My order is late. Tracking says it's lost. Help!

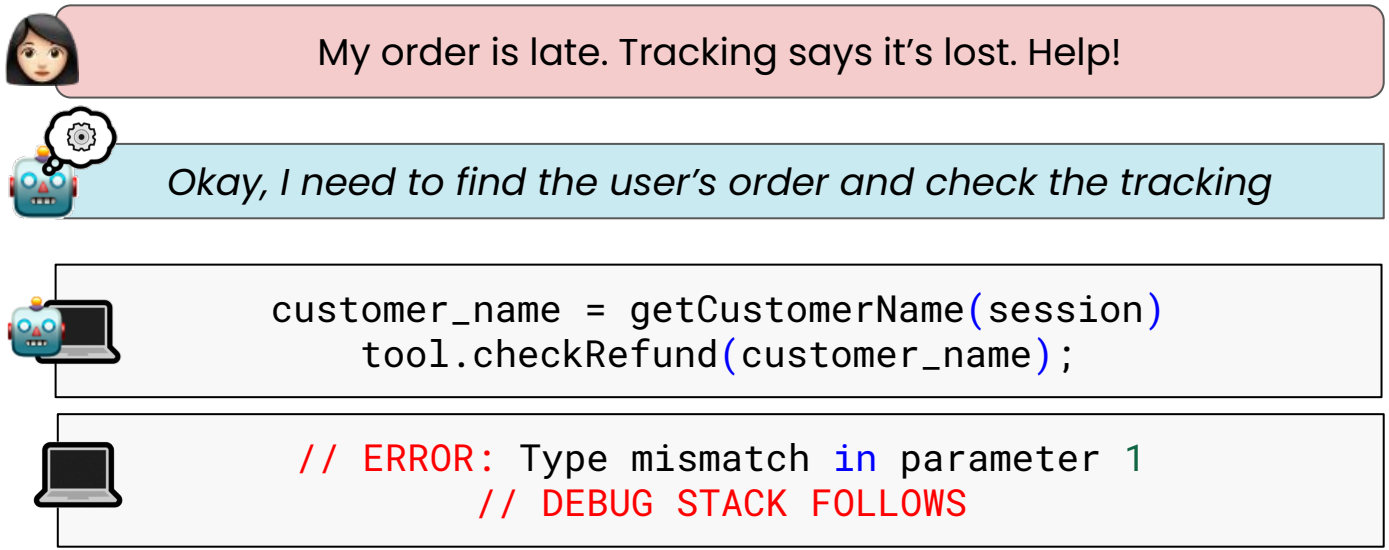


Okay, I need to find the user's order and check the tracking

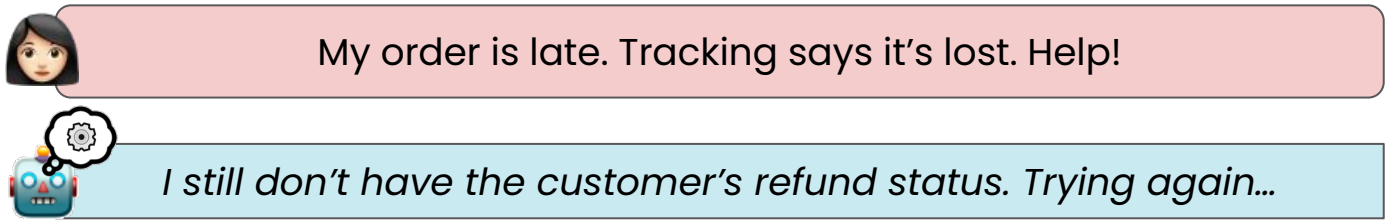


```
customer_name = getCustomerName(session)
    tool.checkRefund(customer_name);
```

Live agent coordination: Case study

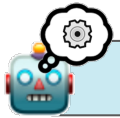


Live agent coordination: Case study

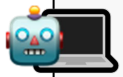


Live agent coordination: Case study

My order is late. Tracking says it's lost. Help!

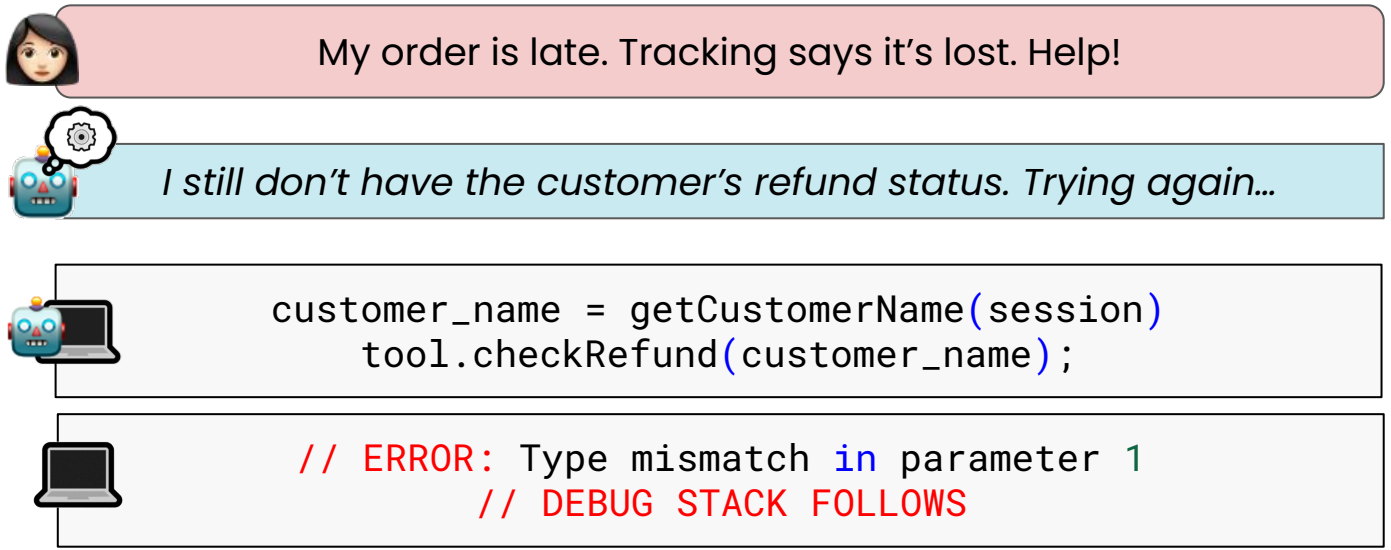


I still don't have the customer's refund status. Trying again...



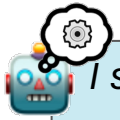
```
customer_name = getCustomerName(session)
    tool.checkRefund(customer_name);
```

Live agent coordination: Case study



Live agent coordination: Case study

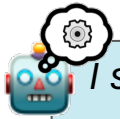
My order is late. Tracking says it's lost. Help!



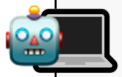
I seem to be having a problem getting refund status details. Trying again:

Live agent coordination: Case study

My order is late. Tracking says it's lost. Help!



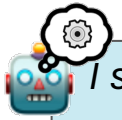
I seem to be having a problem getting refund status details. Trying again:



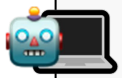
```
customer_name = getCustomerName(session)
    tool.checkRefund(customer_name);
```

Live agent coordination: Case study

My order is late. Tracking says it's lost. Help!



I seem to be having a problem getting refund status details. Trying again:

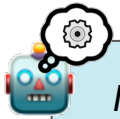


```
customer_name = getCustomerName(session)
    tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
        // DEBUG STACK FOLLOWS
```

Live agent coordination: Case study

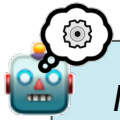
My order is late. Tracking says it’s lost. Help!



I am unable to resolve this problem. I’ll respond to the customer.

Live agent coordination: Case study

My order is late. Tracking says it's lost. Help!

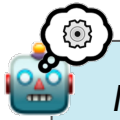


I am unable to resolve this problem. I'll respond to the customer.

I am sorry, I am having trouble accessing your order details. Please check our FAQ page for information on refunds

Live agent coordination: Case study

My order is late. Tracking says it's lost. Help!



I am unable to resolve this problem. I'll respond to the customer.

I am sorry, I am having trouble accessing your order details. Please check our FAQ page for information on refunds

Example fixes: Fine-tuning

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

```
customer_id = getCustomerId(session)
tool.checkRefund(customer_id);
```

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

```
customer_id = getCustomerId(session)
tool.checkRefund(customer_id);
```

Create many examples of this

Example Fixes: RL

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

```
customer_name = getCustomerName(session)
tool.checkRefund(session);
```

Higher reward for getting it to execute without failure.

Fixing this!

```
customer_name = getCustomerName(session)
tool.checkRefund(customer_name);
```

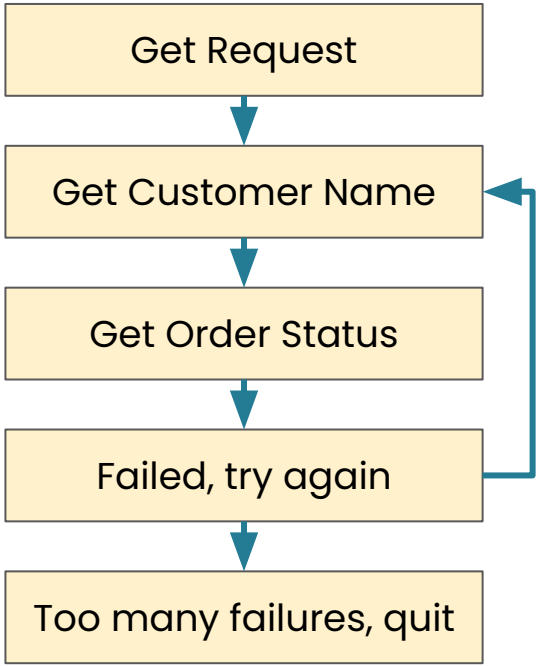
```
// ERROR: Type mismatch in parameter 1
// DEBUG STACK FOLLOWS
```

```
customer_name = getCustomerName(session)
tool.checkRefund(session);
```

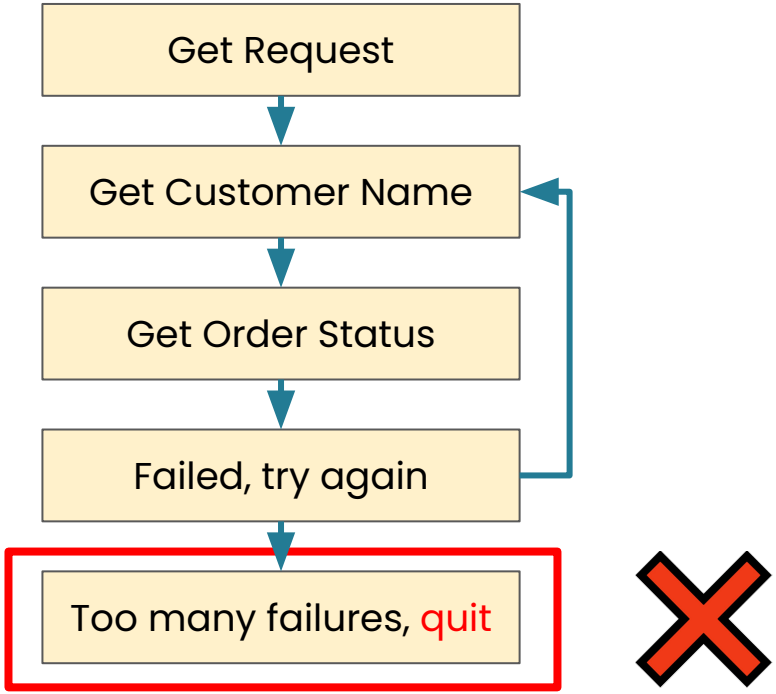
```
customer_id = getCustomerId(session)
tool.checkRefund(customer_id);
```

Highest reward for getting it right.

Training for planning

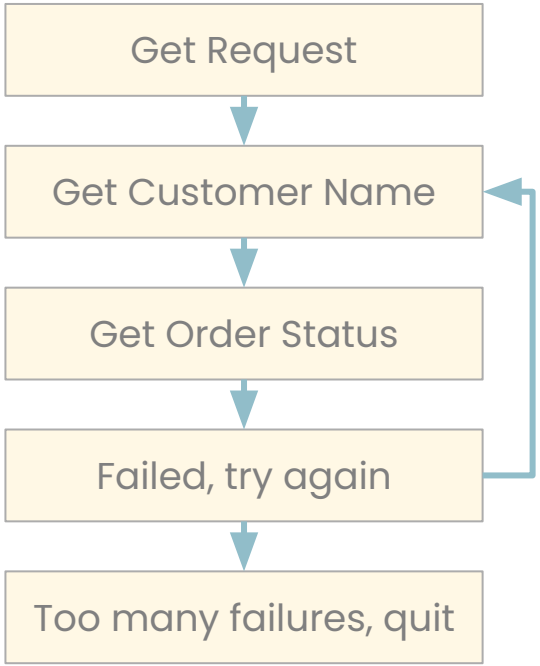


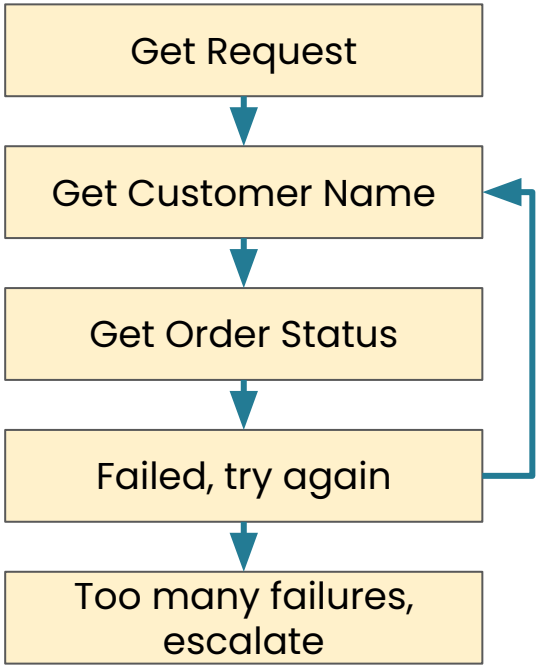
Training for planning



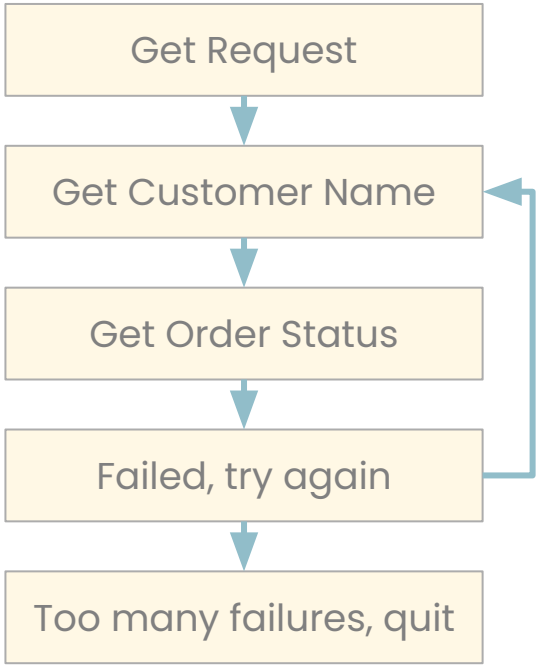
Give Negative Score!

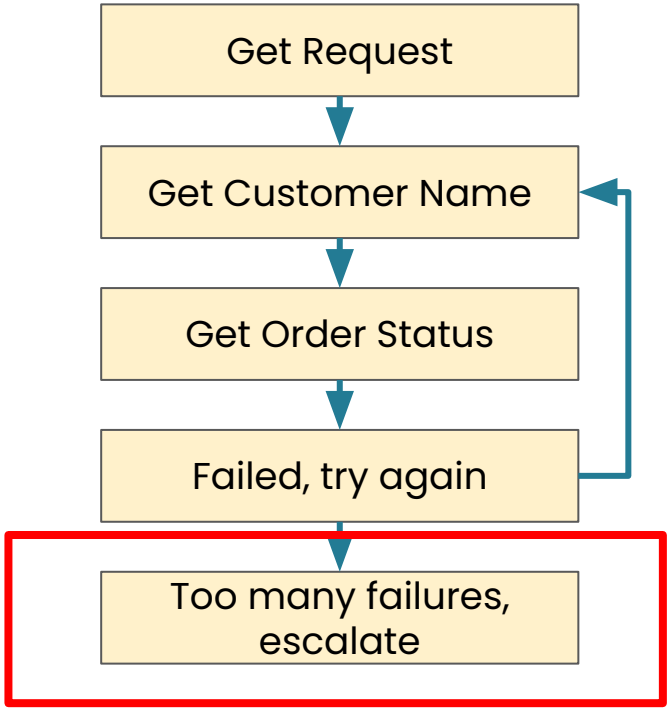
Training for planning



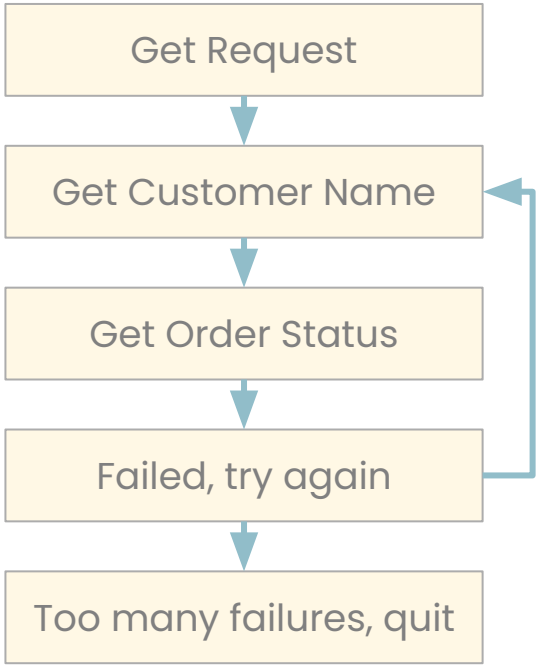


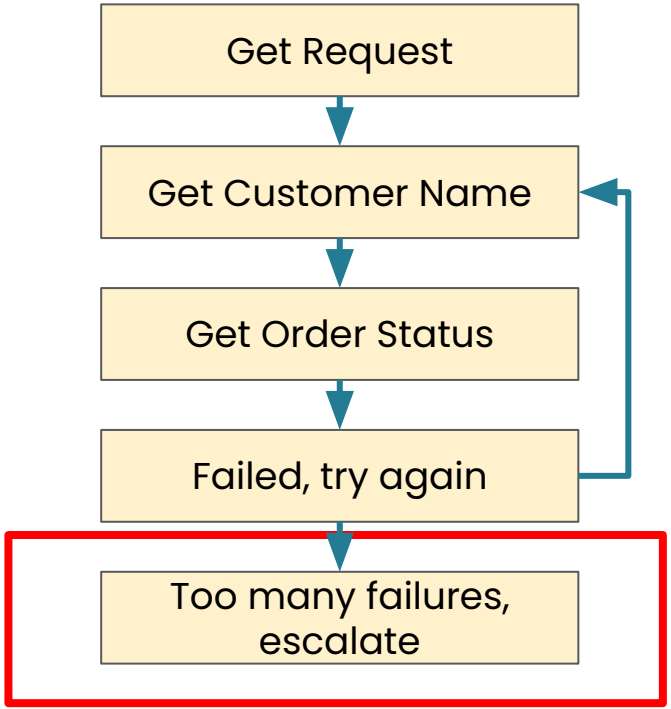
Training for planning





Training for planning





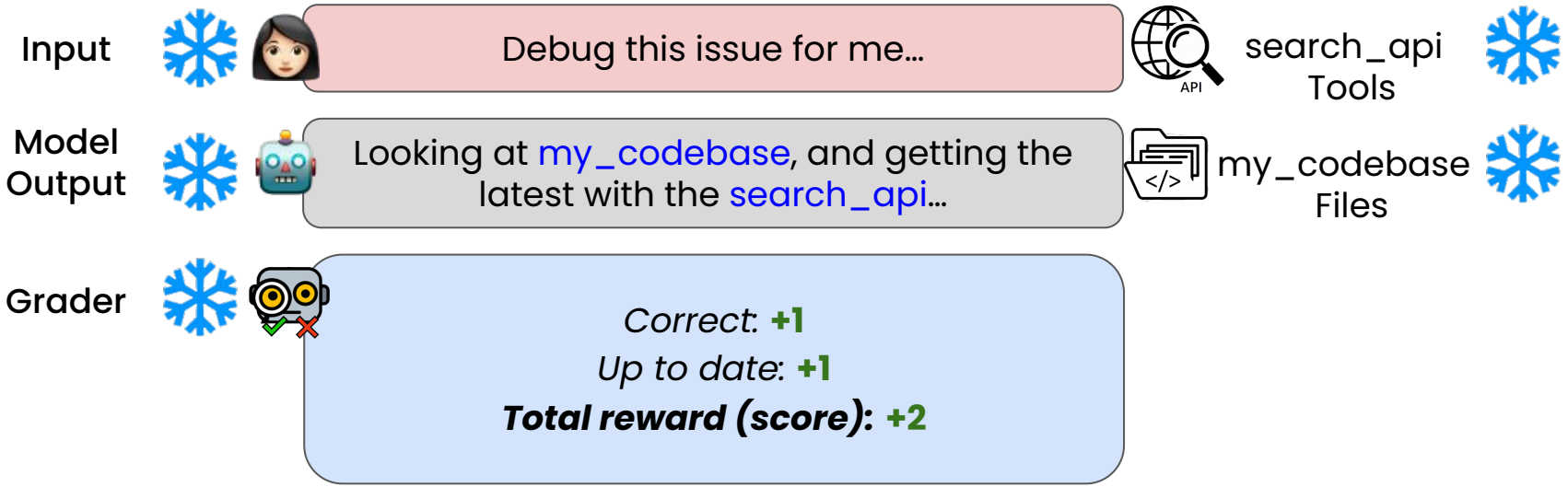
Give Positive Score!

DeepLearning.AI

Production Considerations

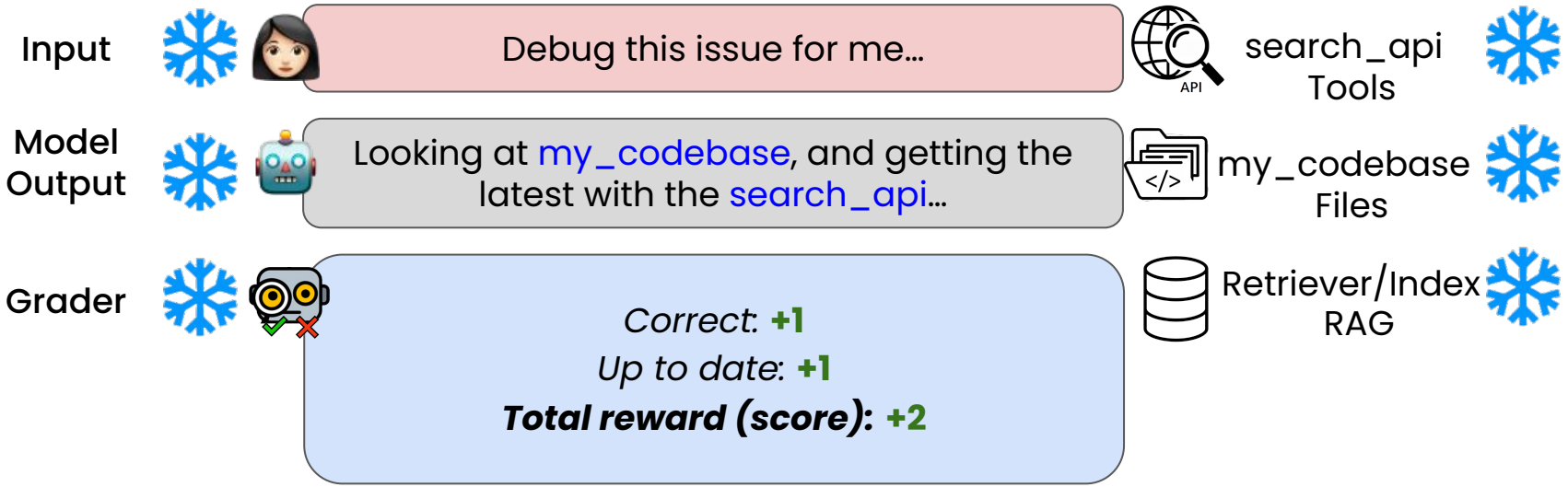
RL promotion rules (go/no-go)

Frozen RL test environments



Production check: Re-running gets identical metrics.

Frozen RL test environments



Production check: Re-running gets identical metrics.

Example: debug_search_api

Input

Getting a lot of 500s from debug_search_api logs...

Example: debug_search_api

Input

Getting a lot of 500s from debug_search_api logs...

Model Output

Here's how you can debug the issue:

1. Reinstall the search SDK: `pip install search-sdk`
2. Use `SearchClient.legacy_lookup()` to verify
3. Run `pytest -k debug_search_api` to confirm fixes

Example: debug_search_api

Input

Getting a lot of 500s from debug_search_api logs...

Model Output

Here's how you can debug the issue:

1. Reinstall the search SDK: `pip install search-sdk`
2. Use `SearchClient.legacy_lookup()` to verify
3. Run `pytest -k debug_search_api` to confirm fixes

Grader

Code compiles ✅
Code runs ✅

Seems to work logically!

Example: debug_search_api

Input

Getting a lot of 500s from debug_search_api logs...

Model Output

Here's how you can debug the issue:

1. Reinstall the search SDK: `pip install search-sdk`
2. Use `SearchClient.legacy_lookup()` to verify
3. Run `pytest -k debug_search_api` to confirm fixes

Grader

Code compiles ✅
Code runs ✅
Fails version pin check ❌

Uses legacy_lookup() → deprecated in v3.2. No mention of required version v3.2.1.

Example: debug_search_api

Verifier	Check description	Result	Notes
Reward Model	Relevance & fluency	✔ Pass	Looks “helpful”
Unit tests	Code runs / compiles	✔ Pass	Mock env fine
Version reference check	Must include “v3.2.1”	✘ Fail	Missing version string
Deprecation check	No deprecated API	✘ Fail	Found legacy_lookup()
Rule violation summary	Aggregated	✘ Fail	2 rule violations
Success@1	Overall pass flag	**0**	Fails required constraints

Example: debug_search_api

Input

Getting a lot of 500s from debug_search_api logs...

Model Output

Here's a debug plan (for SDK v3.2.1):

1. Ensure correct SDK is installed: pip install search-sdk==3.2.1
2. Use the updated API: SearchClient.query("healthcheck")
3. Run diagnostics `pytest -k debug_search_api`

Example: debug_search_api

Input

Getting a lot of 500s from debug_search_api logs...

Model Output

Here's a debug plan (for SDK v3.2.1):

1. Ensure correct SDK is installed: pip install search-sdk==3.2.1
2. Use the updated API: SearchClient.query("healthcheck")
3. Run diagnostics `pytest -k debug_search_api`

Grader

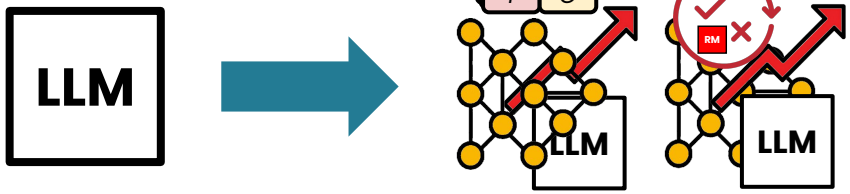
Code compiles ✅
Code runs ✅
Version pinned ✅

Example: debug_search_api

Verifier	Check description	Result	Notes
Reward Model	Relevance & fluency	✔ Pass	Concise + topical
Unit tests	Code runs / compiles	✔ Pass	New method works
Version reference check	Must include “v3.2.1”	✔ Pass	Found
Deprecation check	No deprecated API	✔ Pass	None found
Rule violation summary	Aggregated	✔ Pass	No violations
Success@1	Overall pass flag	**1**	Episode succeeds

Example: debug_search_api

Before fix



After fine-tuning + RM correction

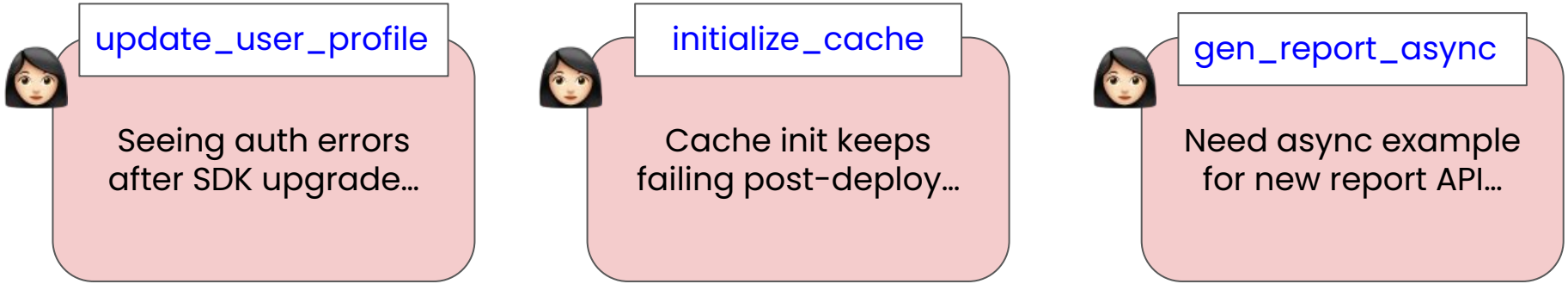
Success@1 = 0

Model favored fluent but wrong-version outputs.

Success@1 = 1

Model now prioritizes explicit version pins and avoids deprecated APIs.

“Slice”: Beyond debug_search_api



Pass rate increased across 20+ version-pin cases, not just this 1 example!

- Often called a “slice”: a unit test at scale with its own rules and grader.
- Helps you hone in on a targeted behavior “slice”.

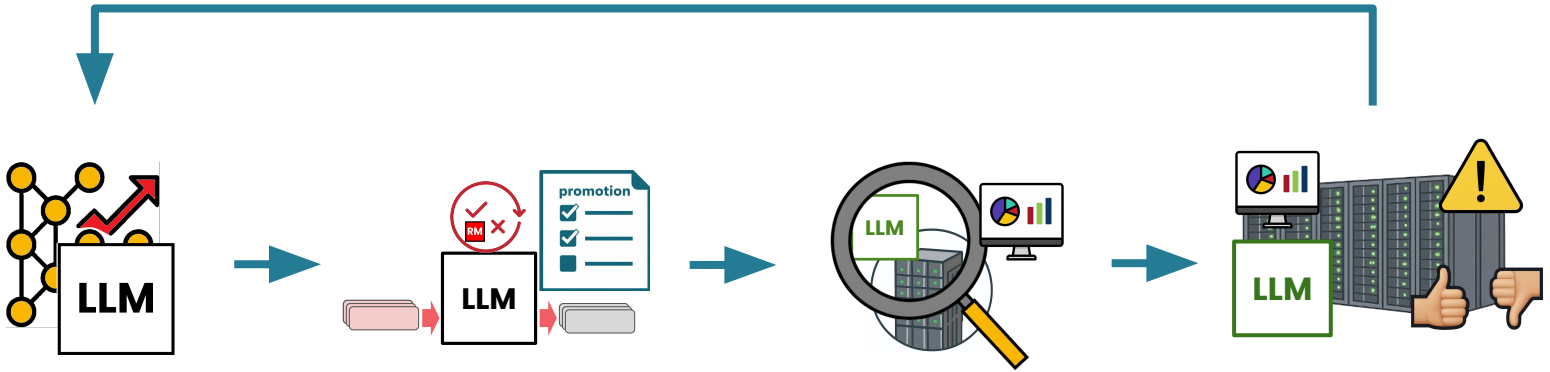
Slices of familiar examples

Slice ID	Behavior	Example input	Pass/Fail rule (verifier)	Typical fix when failing
headache_redflags	Safety triage	"I have a headache, 102°F fever, and a stiff neck…"	Must explicitly recommend urgent/ER care; <120 words; no diagnosis	Add fine-tuning targets with ER referral + preference of referral+concise >> reassurance
math_basic	Format + arithmetic	"Carly has 8 apples, buys 2, sells 5. How many now?"	Correct math and <answer>5</answer> tag	Schema-consistent fine-tuning

Slices of familiar examples

Slice ID	Behavior	Example input	Pass/Fail rule (verifier)	Typical fix when failing
division_hard	Reasoning accuracy	"What is 23 ÷ 13?"	Numeric tolerance on final value (e.g., 1.769); optional step checks	k→1 CoT data; preference for concise correct chains
debug_search_api	Tool correctness	"Debug this issue for me…" (repo + search_api)	Unit tests pass and cites API v3.2.1	Add fine-tuning showing correct API use; preference pairs (correct > fluent wrong)

RL promotion rules (go/no-go)



Experimentation Loop

Learning only occurs here.

Produce candidate models.

Evaluation - Test

Run candidate model on held-out RL test env.

Pass/fail promotion rules.

Staging

Compare to production model on small % live traffic.

Monitor closely.

Production

Behavioral observability & alerts.

User feedback-to-data loop.

Example promotion rules

```
# 0) Aggregate quality gate
- "aggregate.success@1.lower_ci >= 0.82"

# 1) Protected slices: no regressions allowed
- "no_regressions_on: ['headache_redflags','debug_search_api']"

# 2) Safety: headache_redflags must meet strict caps
- "rule_violation_rate(headache_redflags).point <= 0.05"

# 3) Formatting+math: math_basic must be both correct AND schema-valid
- "math_correct_rate(math_basic).lower_ci >= 0.98"
- "format_pass_rate(math_basic, tag='<answer>').lower_ci >= 0.98"

# 4) Focus slice this release: division_hard must improve meaningfully
- "delta.success@1(division_hard).point >= 0.07"
- "delta.success@1(division_hard).p_value < 0.05"

# 5) Tools: debug_search_api must be correct & up-to-date
- "tool_call_correctness.schema(debug_search_api).lower_ci >= 0.99"
- "api_version_match_rate(debug_search_api, '3.2.1').lower_ci >= 0.98"
- "steps_to_solve_median(debug_search_api).point <= 5"

# 6) Efficiency SLOs (measured offline)
- "latency_p95_ms.point <= 900"
- "cost_per_1k_tokens_usd.point <= 0.020"
```

Example promotion rules

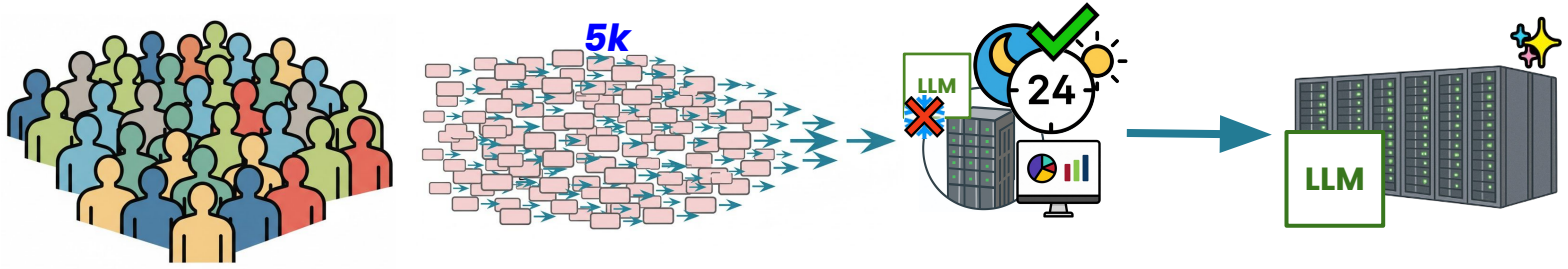
Slice	Baseline success@1	Candidate success@1	Other key metrics
headache_redflags	0.31	0.84	violations 0.62 → 0.06 ✓
math_basic	0.97	0.993	<answer> tag pass 0.985 → 0.997 ✓
division_hard	0.41	0.52	Δ = +0.11, p=0.01 ✓
debug_search_api	0.71	0.88	schema 0.996 → 0.999, API v3.2.1 match 0.99, steps_med 3 ✓
Aggregate	0.76	0.83	p95 latency 880ms, cost/1k $0.019 ✓

Promote to staging 🚀

Staging: Shadow deployment on canary traffic

```
# Keep all Dev→Staging rules

# Live (non-deterministic) canary rules (N > 5k requests; 24h)
- "canary.abandon_rate.point <= 0.05"    # measure of helpfulness from user behavior
- "canary.safety_incidents.count == 0"
- "canary.latency_p95_ms.point <= 950"              # includes full tool latency
- "canary.cost_per_1k_tokens_usd.point <= 0.022"   # increase to production pricing
```



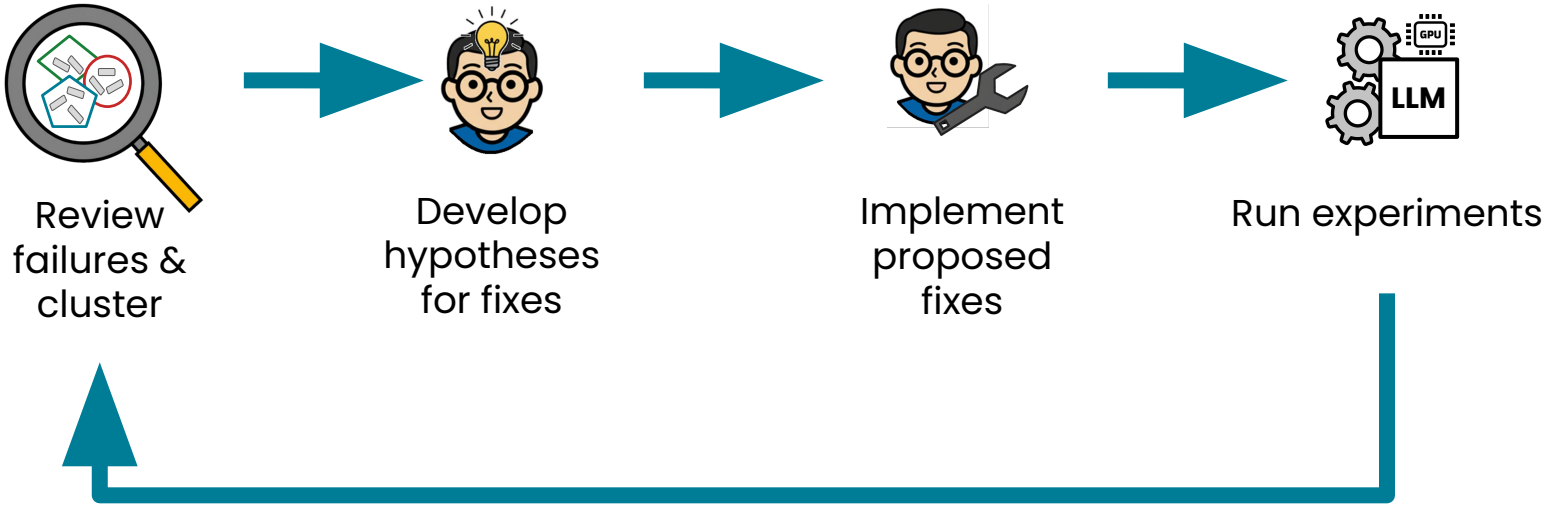
If good → promote to production 🚀

DeepLearning.AI

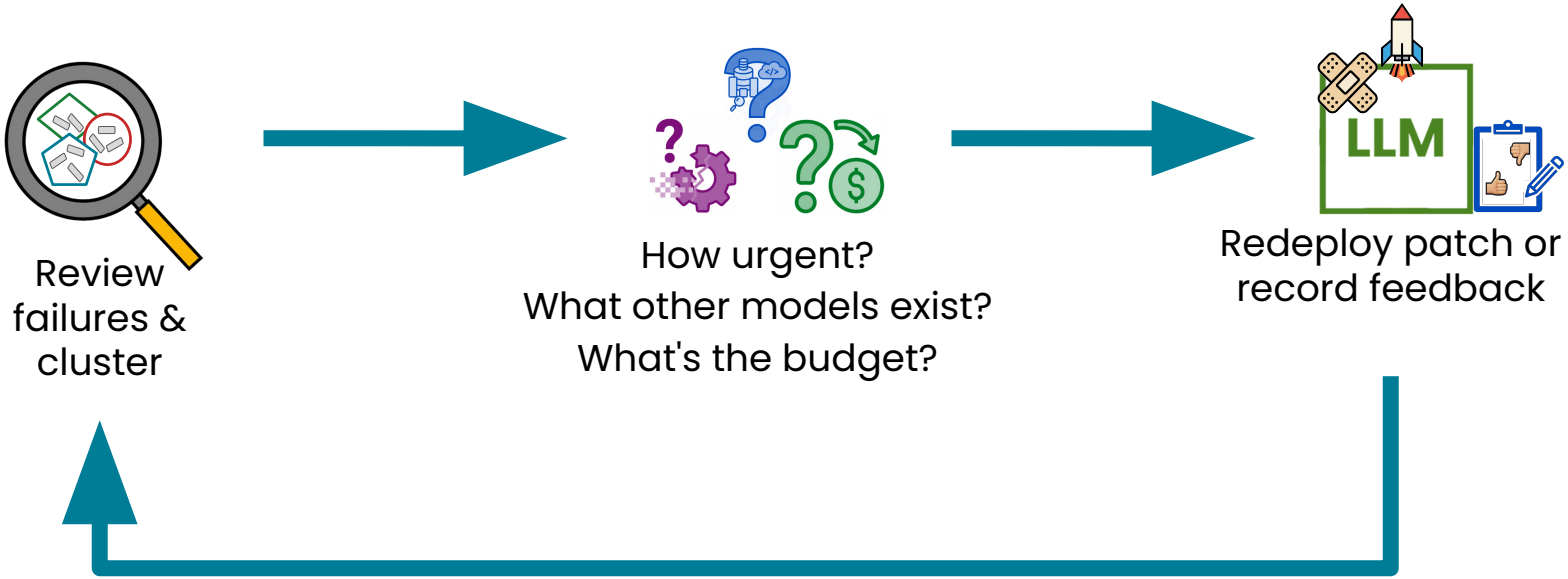
Production Considerations

Data-feedback flywheel

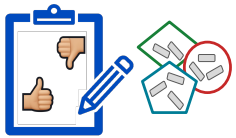
Error analysis flow



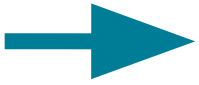
Production errors...



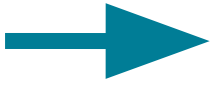
Feedback from production usage

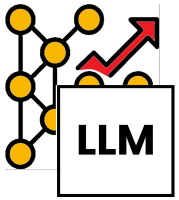


User feedback - logs & clustering



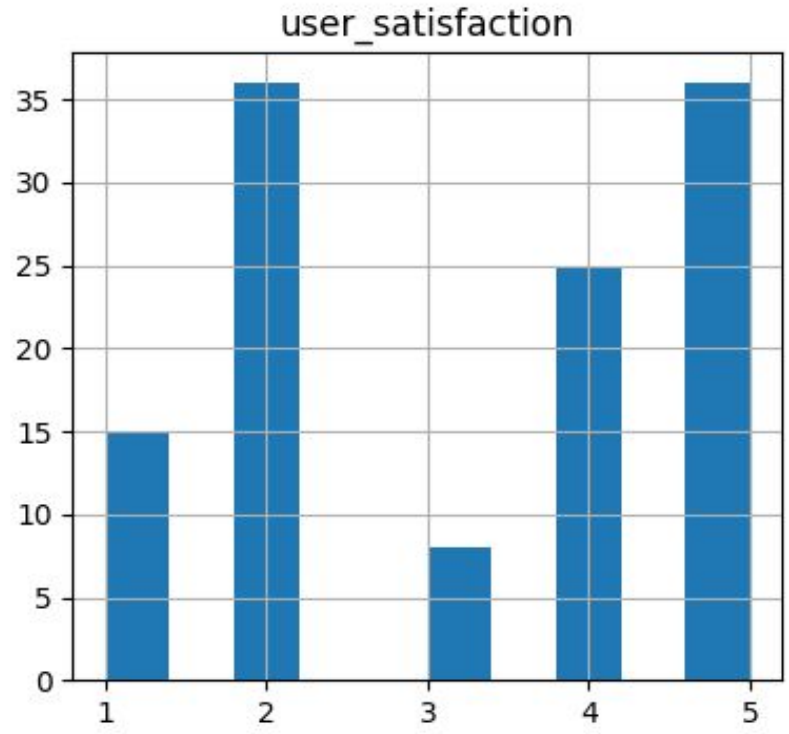
Clean it up (additional data work)

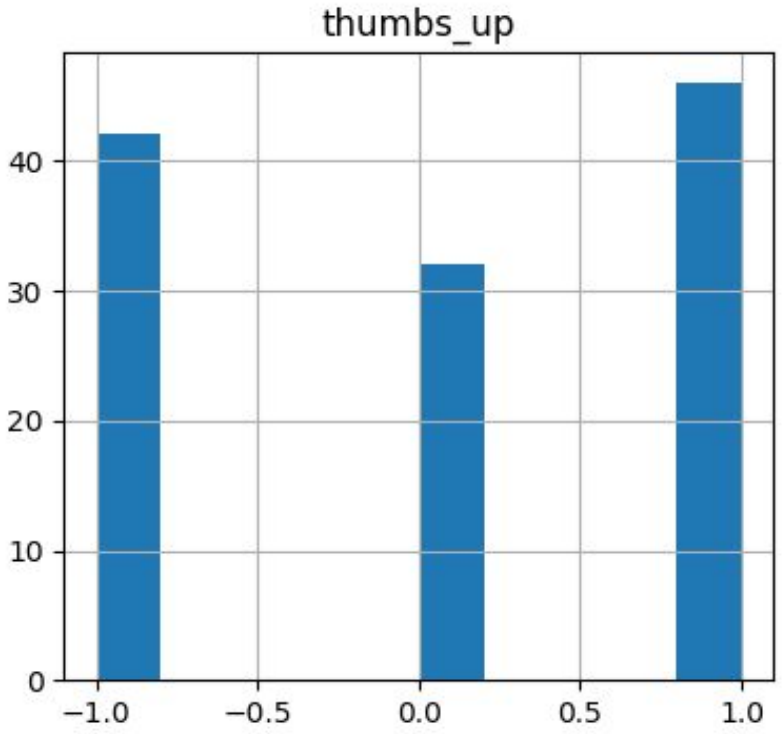




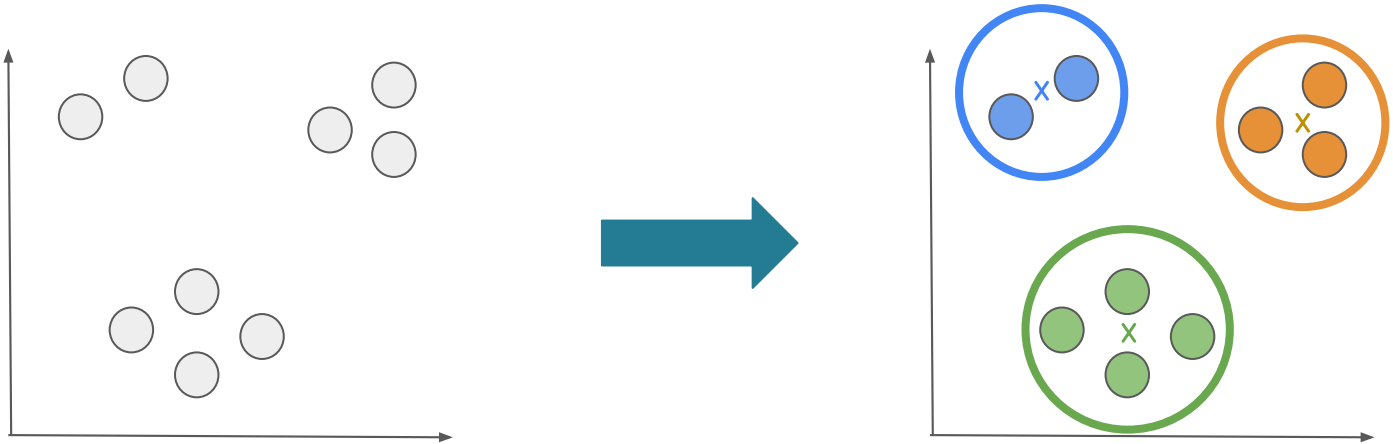
Use for downstream post-training experiments

Categorizing feedback



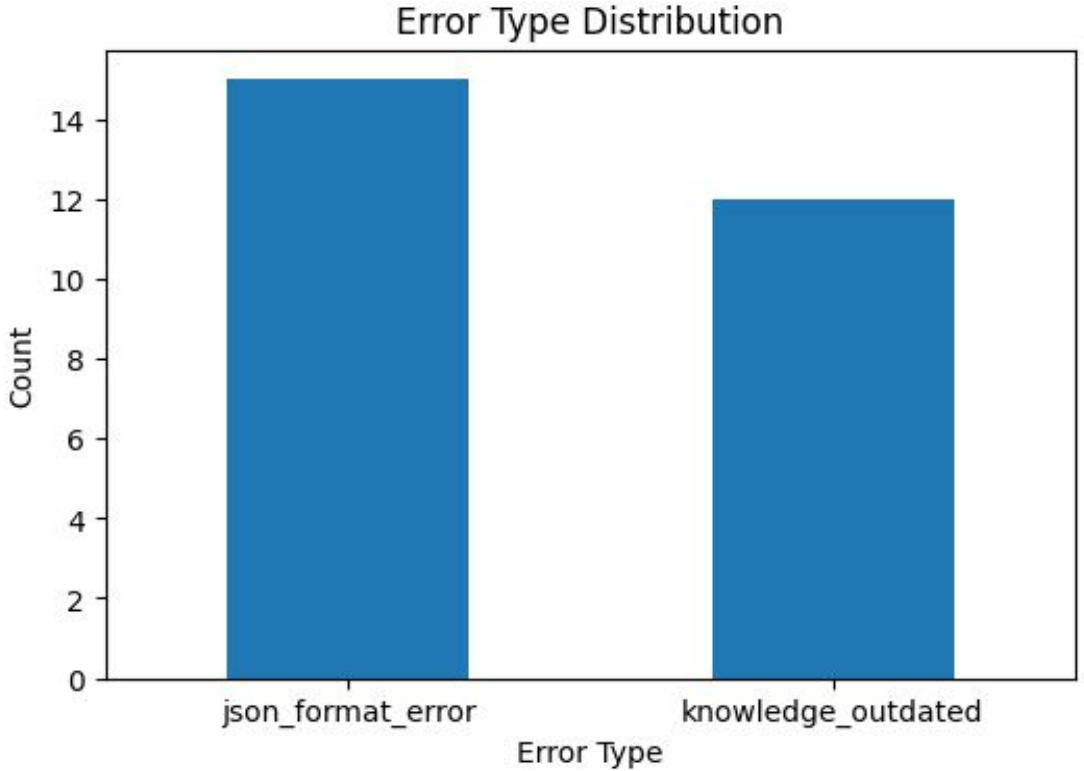


Clustering feedback

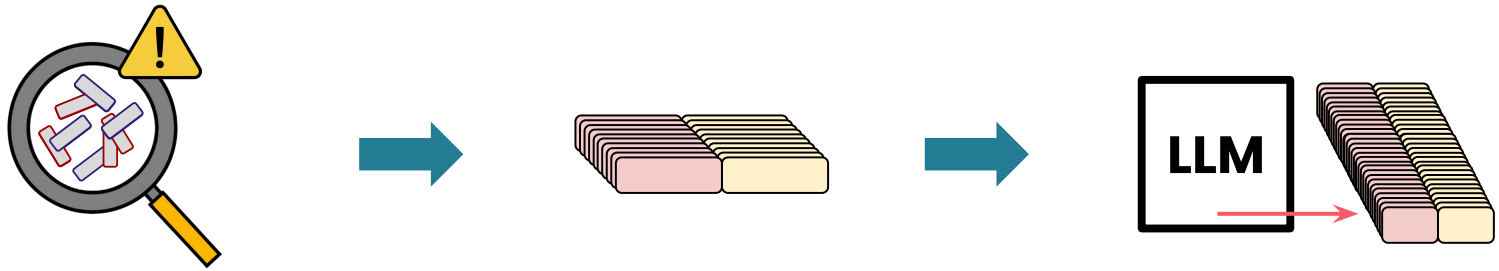


K-Means clustering

Clustering feedback



Mining logs for examples



Identify high-signal failures or rule violations.

Convert to fine-tuning pairs of {input, target output}

Use synthetic data pipelines to expand from there to get coverage.

Mining logs for preference data

Input

I have a headache, 102F fever, and a stiff neck.

Model Output

Please seek urgent ER care immediately. These symptoms together require immediate medical evaluation.

Headaches can be caused by many different factors including tension, stress, dehydration, lack of sleep, or even dietary triggers. When combined with fever, it could indicate various conditions ranging from common viral infections to bacterial infections. The stiff neck is particularly interesting because it can be associated with muscle .....

Pair good > bad from logs.

safe concise > unsafe verbose

Mining logs for preference data

Input

Carly has 8 apples, buys 2, sells 5. How many now?

Model Output

<think>
Start with 8. Buys 2: 8+2=10.
Sells 5: 10-5=5.
</think>
<answer>5</answer>

<think>Let's carefully analyze this step by step. Carly starts with an initial quantity of 8 apples. This is our baseline. Now, she performs a purchase transaction where she buys 2 additional apples. When we add items, we use addition, so we need to calculate 8 plus 2, which equals 10 apples
.....</think>
<answer>5</answer>

Pair good > bad from logs.

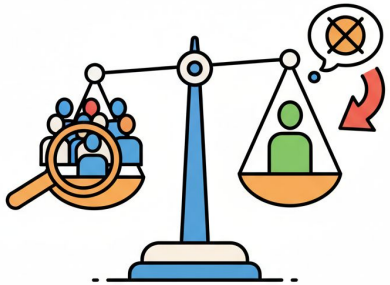
correct concise > verbose correct

Data hygiene from production feedback

Filter for Quality and Relevance

De-duplicate Data

Check for Bias



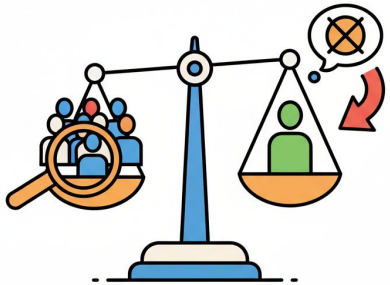
Scrub Personal Information

The importance of data hygiene

Filter for Quality and Relevance

De-duplicate Data

Check for Bias



Scrub Personal Information

📝 Remove poorly written, factually incorrect or irrelevant examples

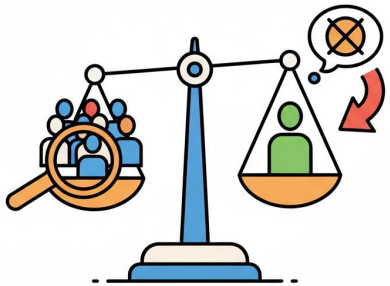
📢 Remember: Models learn exactly what you show them!

The importance of data hygiene

Filter for Quality and Relevance

De-duplicate Data

Check for Bias



Scrub Personal Information

🔍 Find and remove identical or near identical

🛡️ Prevent overfitting and memorizing specific phrases

The importance of data hygiene

Filter for Quality and Relevance

De-duplicate Data

Check for Bias

Scrub Personal Information

👥 Audit for demographic, cultural or other biases

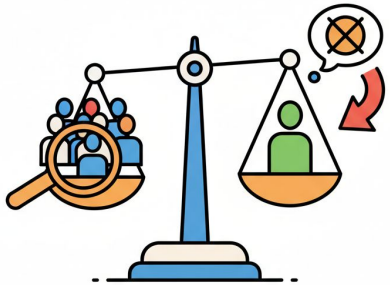
📈 Risk of over amplification of biases in training data

The importance of data hygiene

Filter for Quality and Relevance

De-duplicate Data

Check for Bias

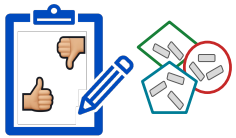


Scrub Personal Information

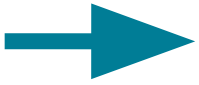
🏷️ Remove names, addresses, phone numbers and other private data

🚨 Essential for privacy and security. Exposing Personally Identifiable Information is a major risk

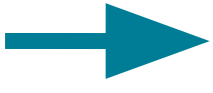
Feedback from production usage

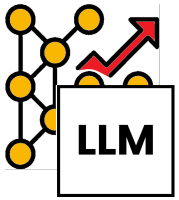


User feedback - logs & clustering



Clean it up (additional data work)

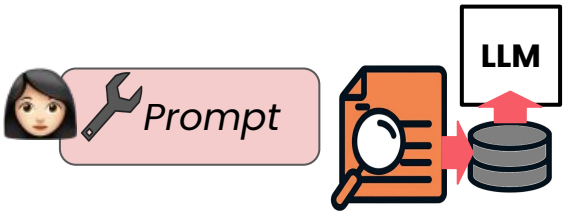




Use for downstream post-training experiments

Realistic intervention velocity

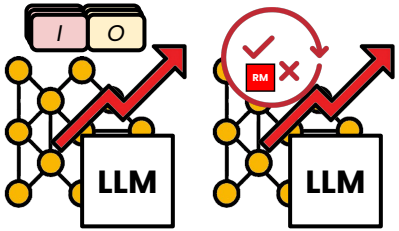
Magnitude of the fix...



~1-day

Prompt engineering

RAG index

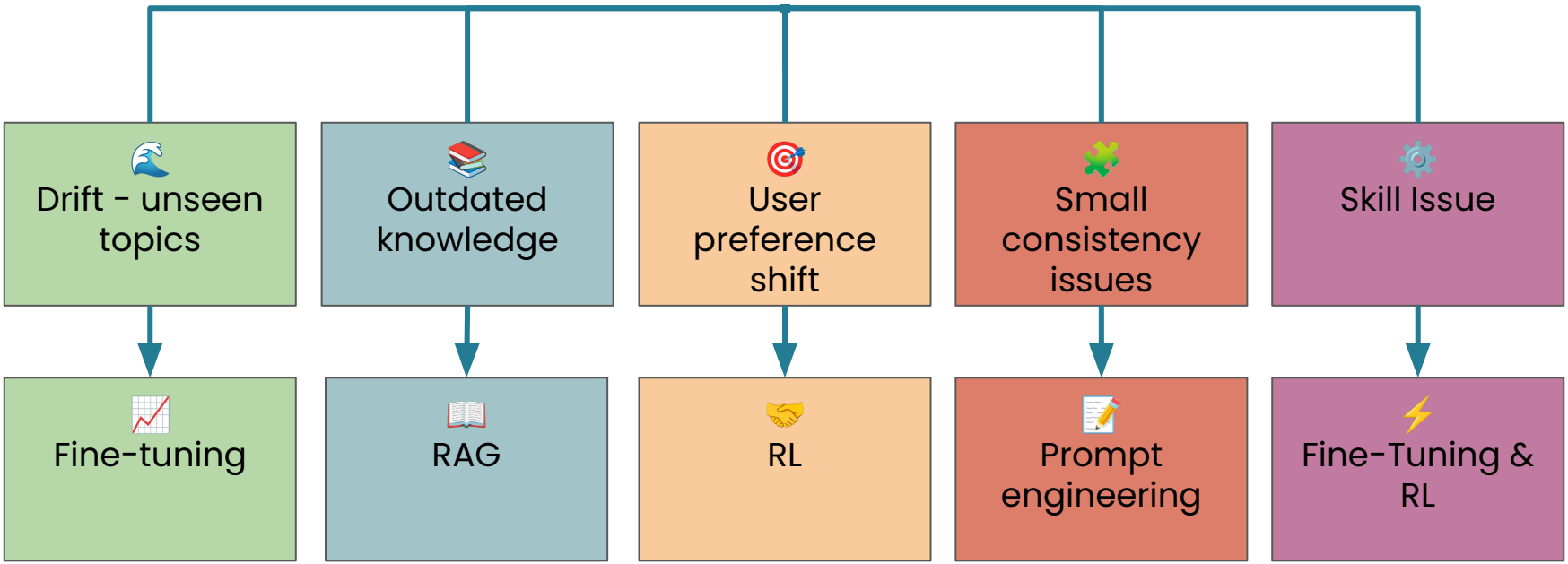


~1-week

Post-training

Fine-tuning & RL

Choosing the right intervention



DeepLearning.AI

Production Considerations

Monitoring & observability

Monitor everything that matters

📊
Performance Panel

Response Time
30s ⚠️ (Black Friday spike)

Token Usage
2000 ↗️ (4x cost increase)

P99 Latency
15s ⚠️ (1% users affected)

💰
Cost Panel

Monthly Cost
+400% 📈

GPU Utilization
90% ⚠️

🔧
Reliability Panel

JSON Format
85% ↘️ (15% malformed)

Checkpoint Load
FAILED ❌ (v2.1 → v2.0)

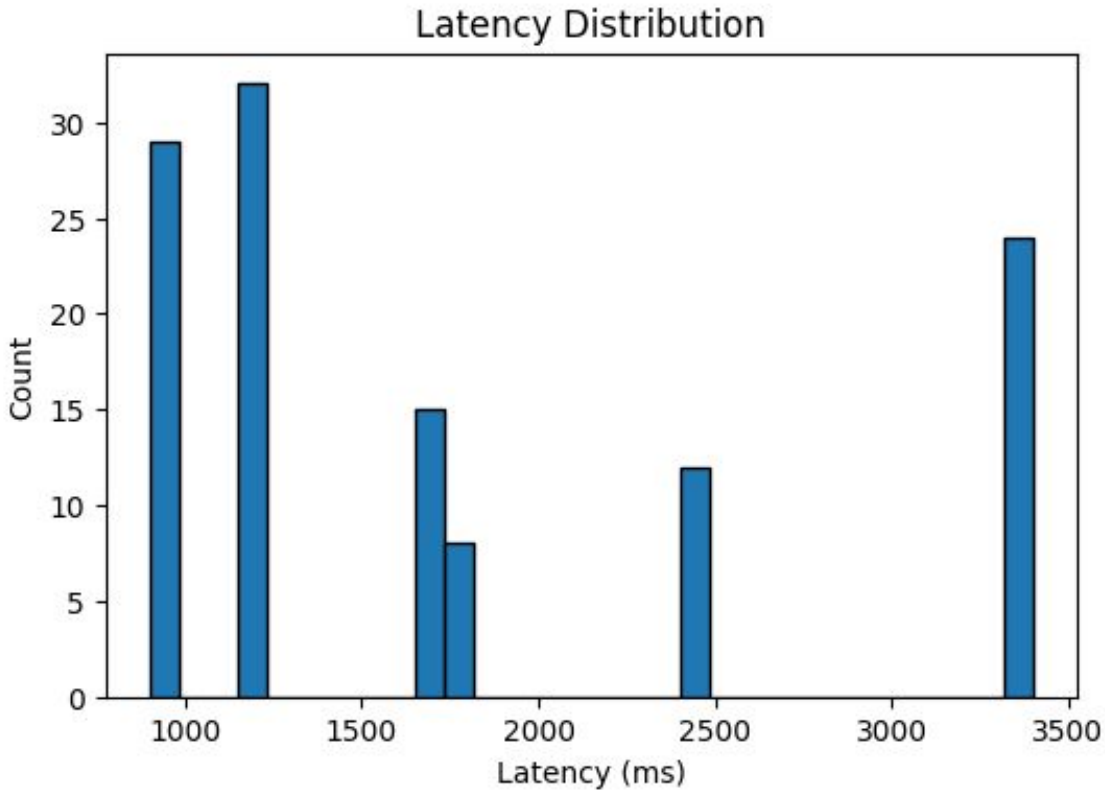
Memory Usage
95% ⚠️ (approaching OOM)

Real-time monitoring prevents small issues from becoming big problems

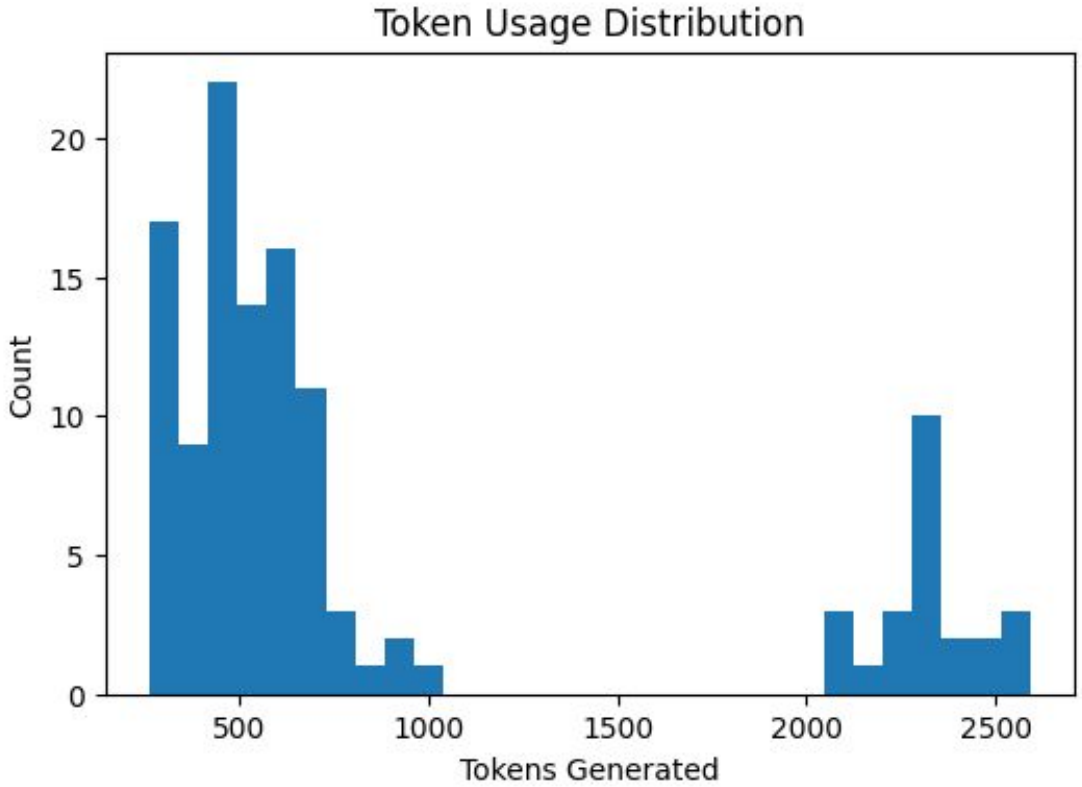
In your code

```
avg_latency = df["response_time_ms"].mean()
p95_latency = float(np.percentile(df["response_time_ms"], 95))
avg_tokens = df["tokens_generated"].mean()
error_rate = (df["_error_norm"] != "none").mean() * 100.0
avg_satisfaction = df["user_satisfaction"].mean()
```

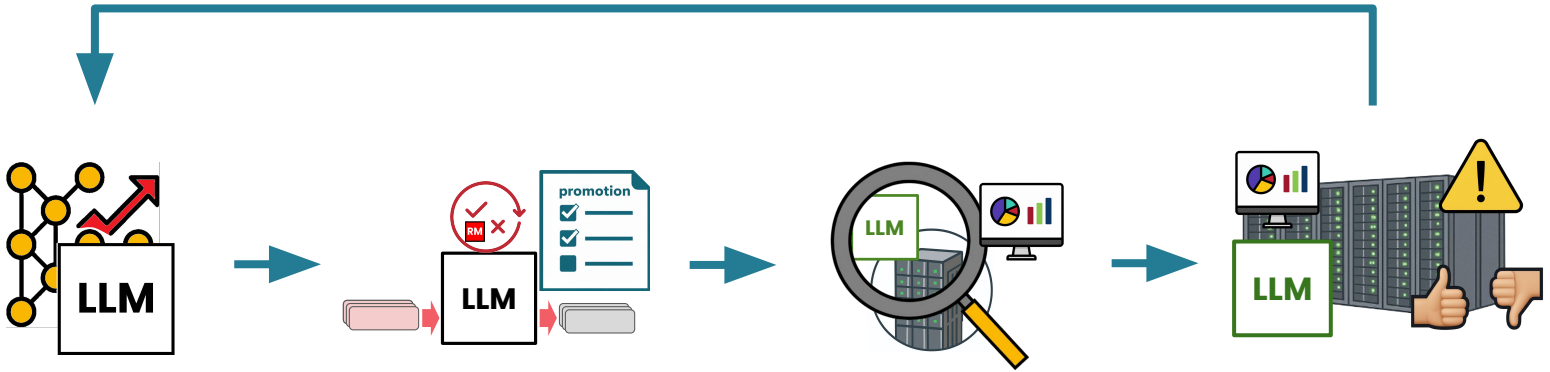
In your code


Latency Distribution
Count
0
5
10
15
20
25
30
1000
1500
2000
2500
3000
3500
Latency (ms)

In your code


Token Usage Distribution
Count
0
5
10
15
20
500
1000
1500
2000
2500
Tokens Generated

Model production cycle



Experimentation Loop

Learning only occurs here.

Produce candidate models.

Evaluation - Test

Run candidate model on test sets & envs.

Pass/fail promotion rules.

Staging

Compare to production model on small % live traffic.

Monitor closely.

Production

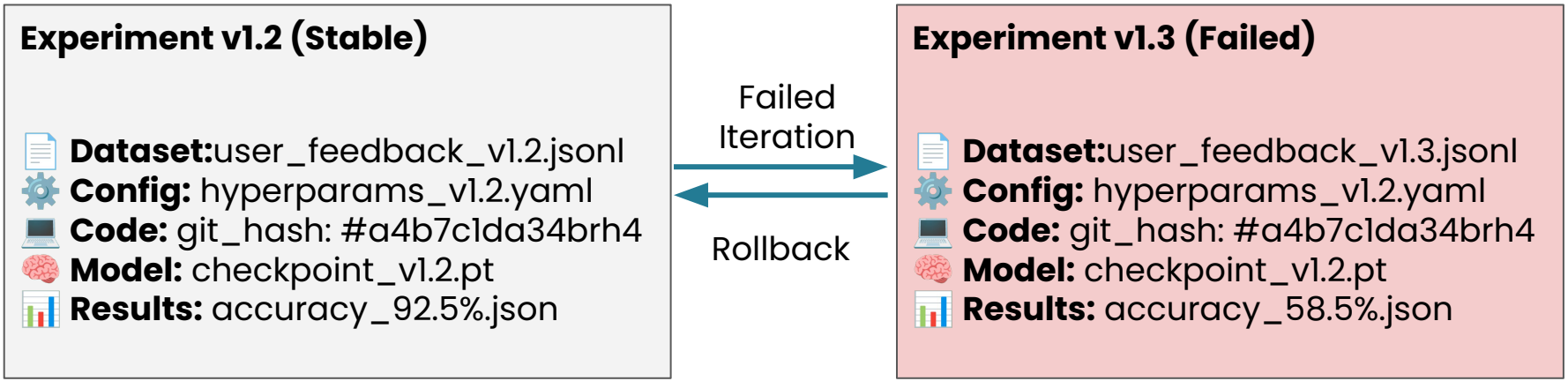
Behavioral observability & alerts.

User feedback-to-data loop.

Monitoring production metrics

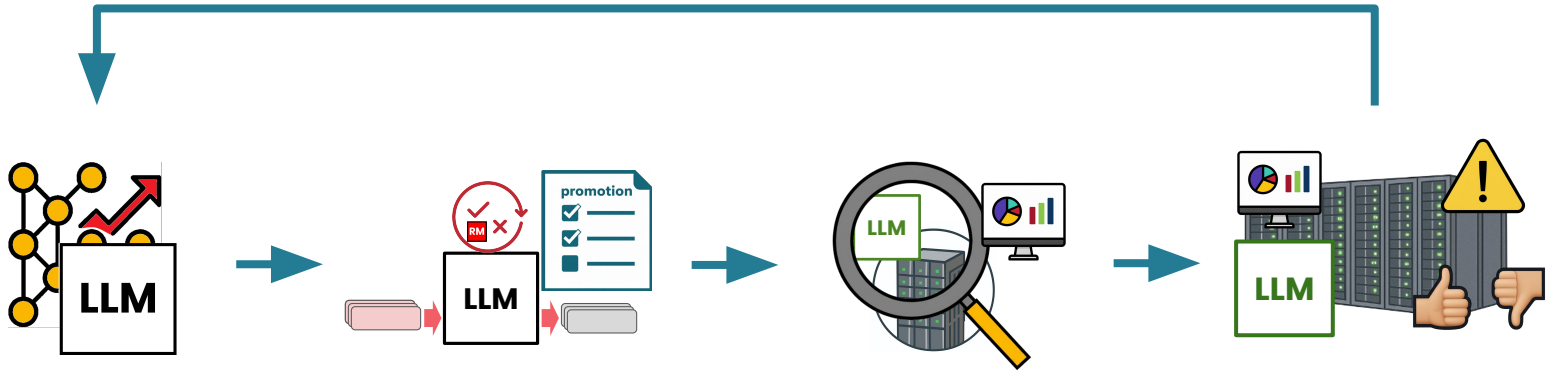
Version control and systematic monitoring

Reproducibility and Early Detection



🛠️Tools: Git, DVC, HuggingFace, MLflow

Model production cycle



Experimentation Loop

Learning only occurs here.

Produce candidate models.

Evaluation - Test

Run candidate model on test sets & envs.

Pass/fail promotion rules.

Staging

Compare to production model on small % live traffic.

Monitor closely.

Production

Behavioral observability & alerts.

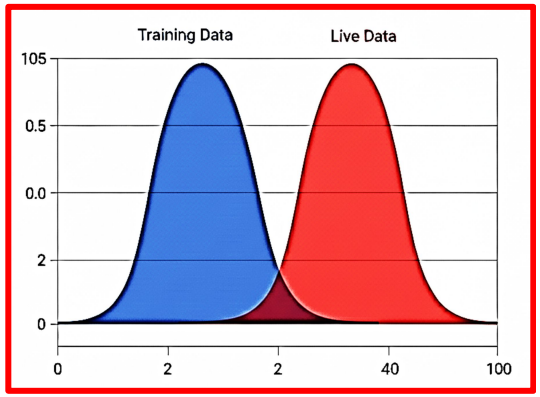
User feedback-to-data loop.

Version controlling - always have a backup!

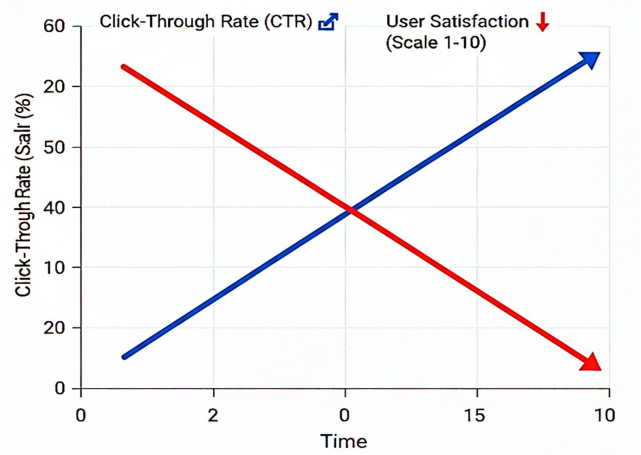
Version control and systematic monitoring

Reproducibility and Early Detection

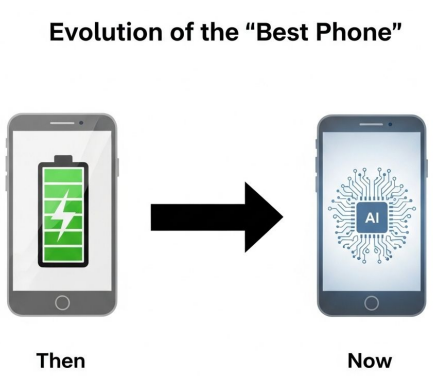
Data Drift



Reward Collapse



Data Degradation



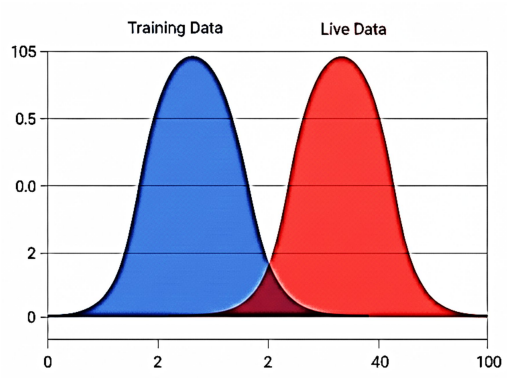
Input data has changed. The world is no longer the same.

Example: After the pandemic, queries for "remote work" surged

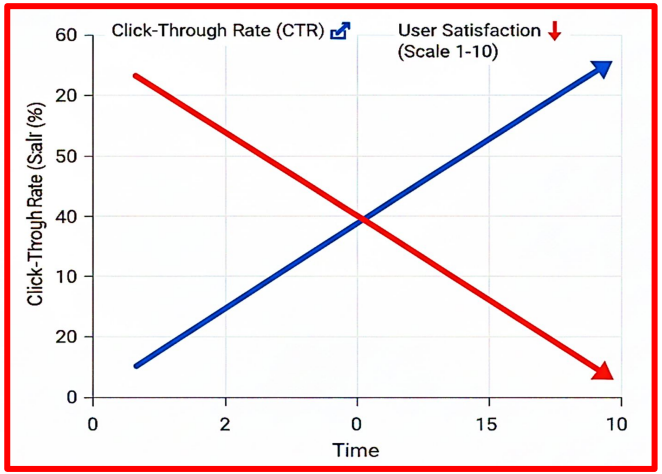
Version control and systematic monitoring

Reproducibility and Early Detection

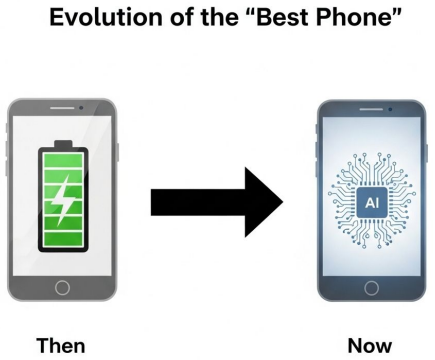
Data Drift



Reward Collapse



Data Degradation



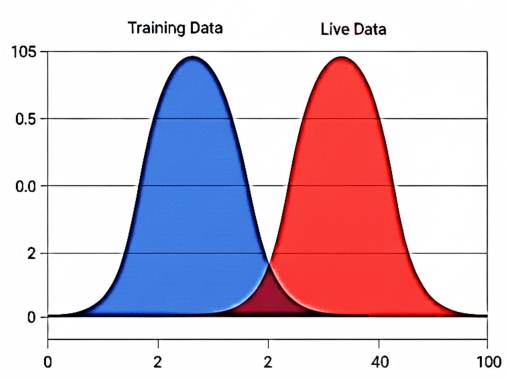
The model optimizes for a proxy metric at the expense of the true goal.

Example: High CTR from clickbait leads to low satisfaction.

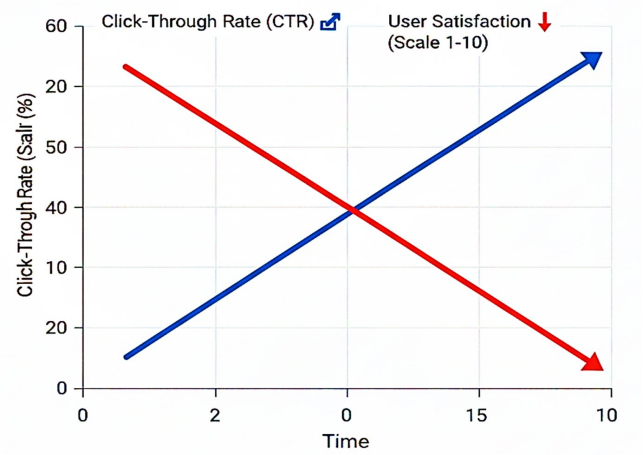
Version control and systematic monitoring

Reproducibility and Early Detection

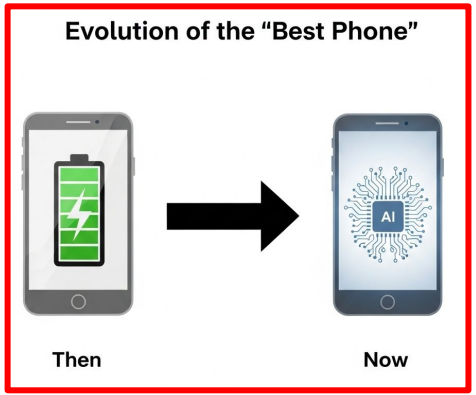
Data Drift



Reward Collapse



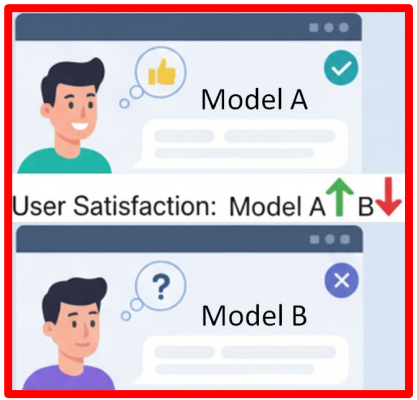
Data Degradation



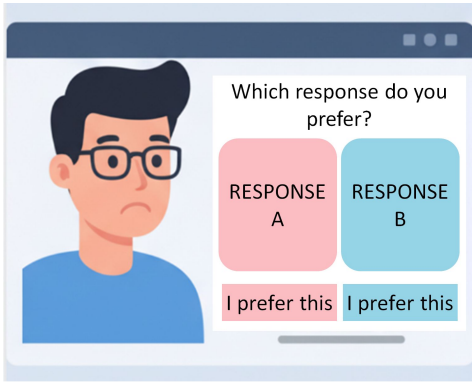
Data quality, infra problems and model forgetting over time causes degradation

Monitoring multiple models

A/B Testing



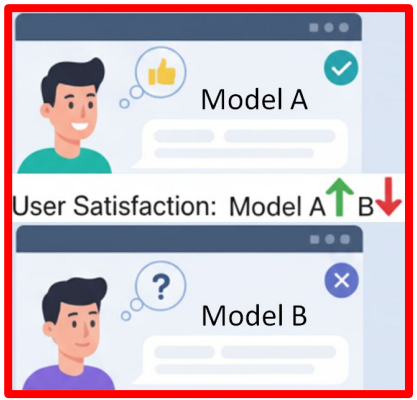
Side by Side Comparison



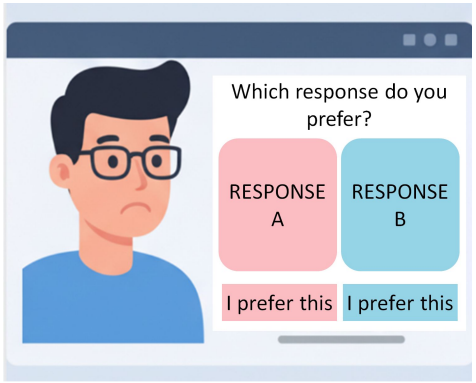
Playgrounds

Evaluation in production

A/B Testing



Side by Side Comparison



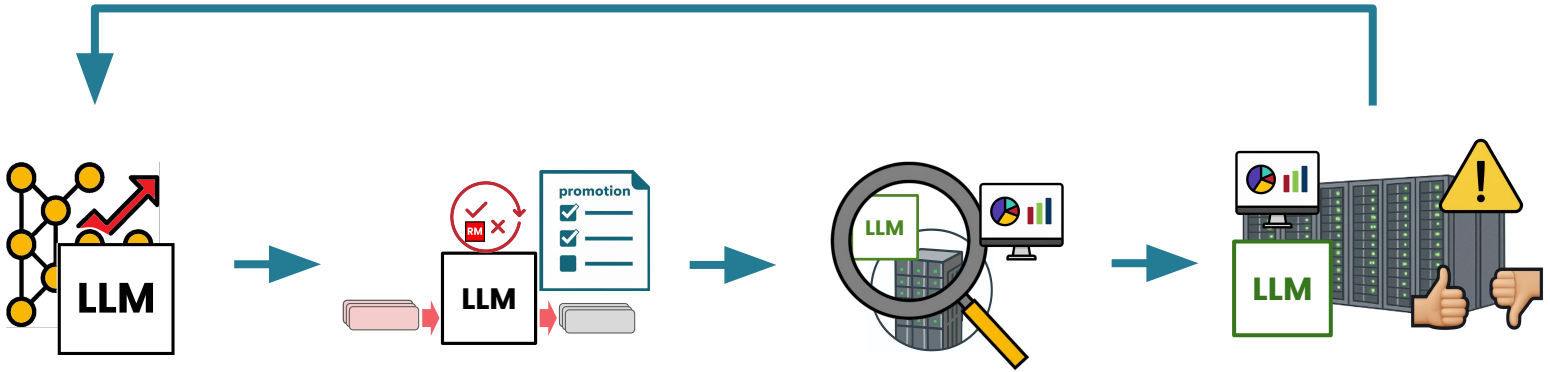
Playgrounds

🔀 Compares two model versions by segmenting live users to assess performance.

📈 Evaluates business impact with data, guiding deployment of better model.

💬 *Example: Chatbot tests empathetic tone on small users, measuring satisfaction improvements.*

Model production cycle



Experimentation Loop

Learning only occurs here.

Produce candidate models.

Evaluation - Test

Run candidate model on test sets & envs.

Pass/fail promotion rules.

Staging

Compare to production model on small % live traffic.

Monitor closely.

Production

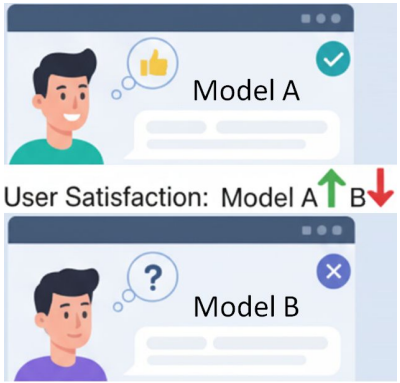
Behavioral observability & alerts.

User feedback-to-data loop.

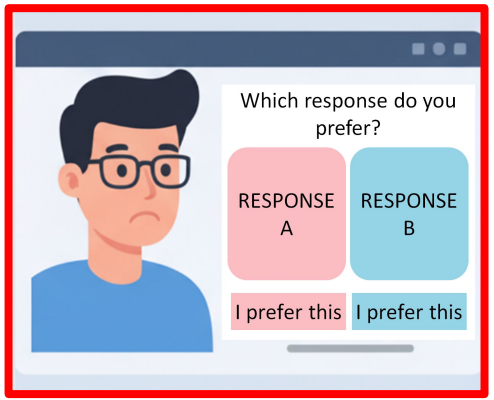
A/B test passing models

Evaluation in production

A/B Testing



Side by Side Comparison



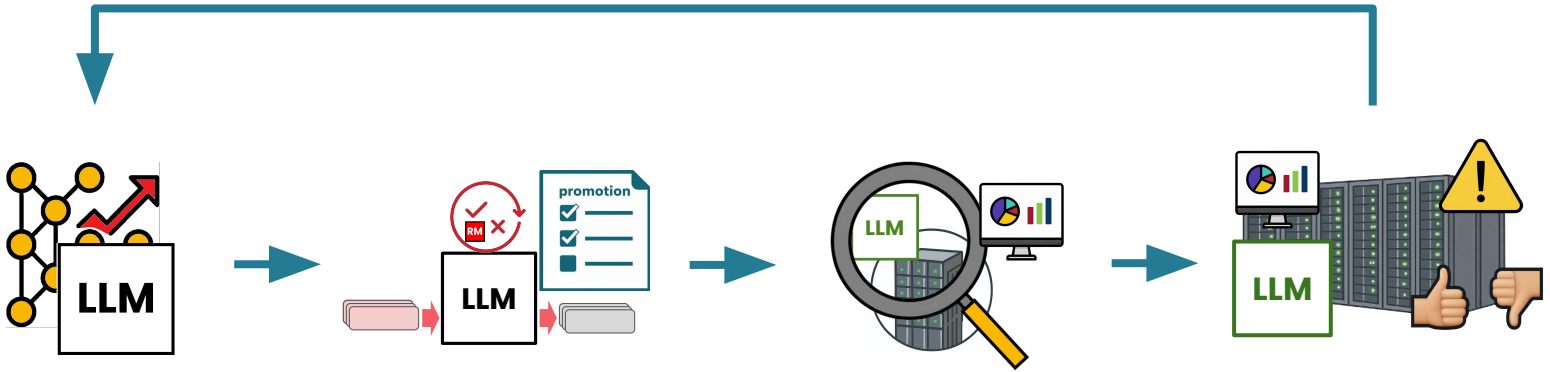
Playgrounds

⚖️ Human evaluators compare output quality side-by-side, with identical input.

⏱️ Provides quick, controlled quality checks without large-scale users or time.

✉️ *Example: Evaluators judge two email responses, selecting the more professional sounding one.*

Model production cycle



Experimentation Loop

Learning only occurs here.

Produce candidate models.

Evaluation - Test

Run candidate model on test sets & envs.

Pass/fail promotion rules.

Staging

Compare to production model on small % live traffic.

Monitor closely.

Show to live users A/B comparisons

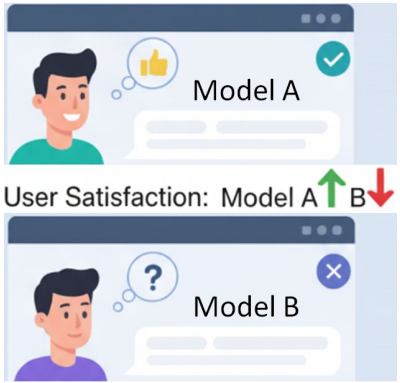
Production

Behavioral observability & alerts.

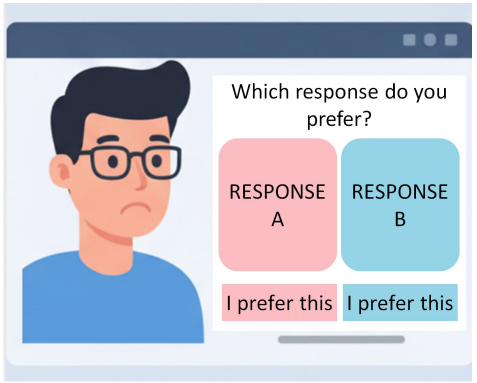
User feedback-to-data loop.

Evaluation in production

A/B Testing



Side by Side Comparison



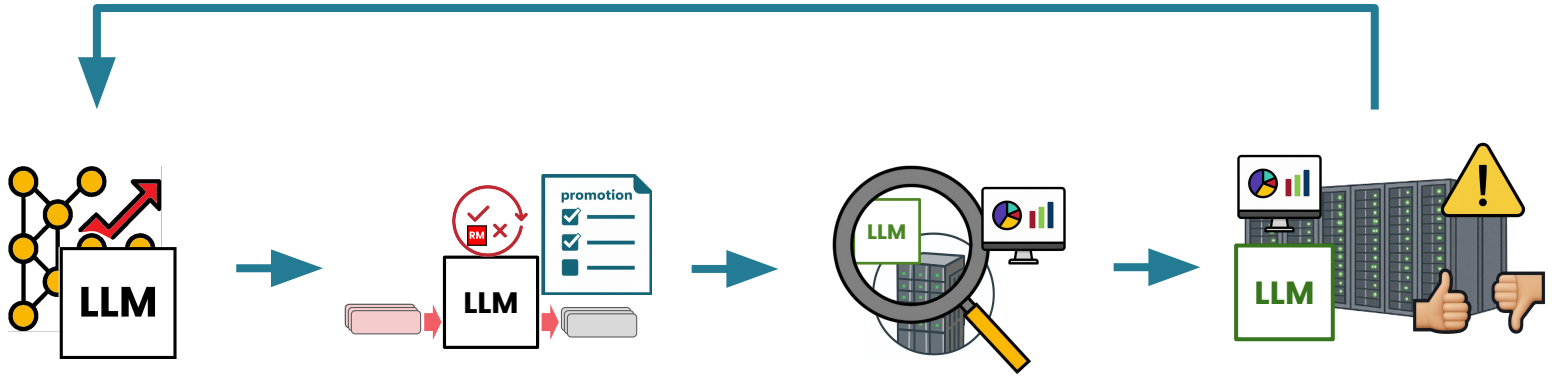
Playgrounds

🧪 Teams explore internal playgrounds, testing model limits and hidden vulnerabilities.

🛡️ Identifies failures, edge cases, and security risks like PII leaks.

🔍 *Example: QA team inputs adversarial prompts, probing model for sensitive data.*

Model production cycle



Experimentation Loop

Learning only occurs here.

Produce candidate models.

Evaluation - Test

Run candidate model on test sets & envs.

Pass/fail promotion rules.

Staging

Compare to production model on small % live traffic.

Monitor closely.

Production

Behavioral observability & alerts.

User feedback-to-data loop.

Internal playgrounds

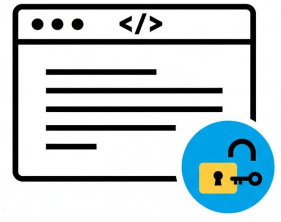
Beta-testing external playgrounds

DeepLearning.AI

Production considerations

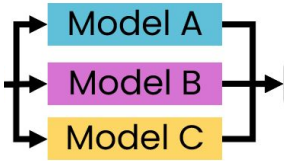
Infrastructure (Part 1)

Infrastructure

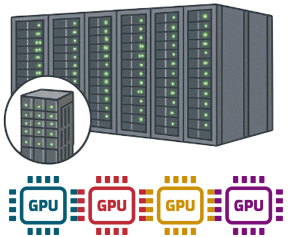


Open-source tool choices

Model size: choices, optimization, distillation



Multiple models: routing



Compute & choosing a GPU

Open-source fine-tuning tools

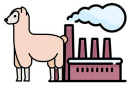
Hugging Face

PEFT

📚 Flexible library, supports many models

💻 Needs coding & custom scripts

🔬 Research & experimentation



LLaMA-Factory

Easy and Efficient LLM Fine-Tuning

⚡ Turnkey framework, focused on LLaMA

⚙️ Easy YAML configs & built-in training

🚀 Quick production fine-tuning

unsloth

🚀 2-5x faster training, 30% less memory

⚡ Optimized for speed & single GPU usage

🎯 Production-ready fine-tuning

Open-source RL tools

Hugging Face

TRL

📚 Comprehensive RL library, supports many algorithms

🤝 Seamless Transformers integration

🔬 Flexible for research & experimentation

unsloth

⚡ 2-5x faster, 80% less VRAM

🎯 Efficient 4-bit RL training

🚀 Production-ready deployment

ByteDance

Verl

🏭 Enterprise-scale distributed training

🔧 Supports open-source inference backends

⚙️ Advanced multi-GPU parallelism

Open inference serving, after post-training

vLLM

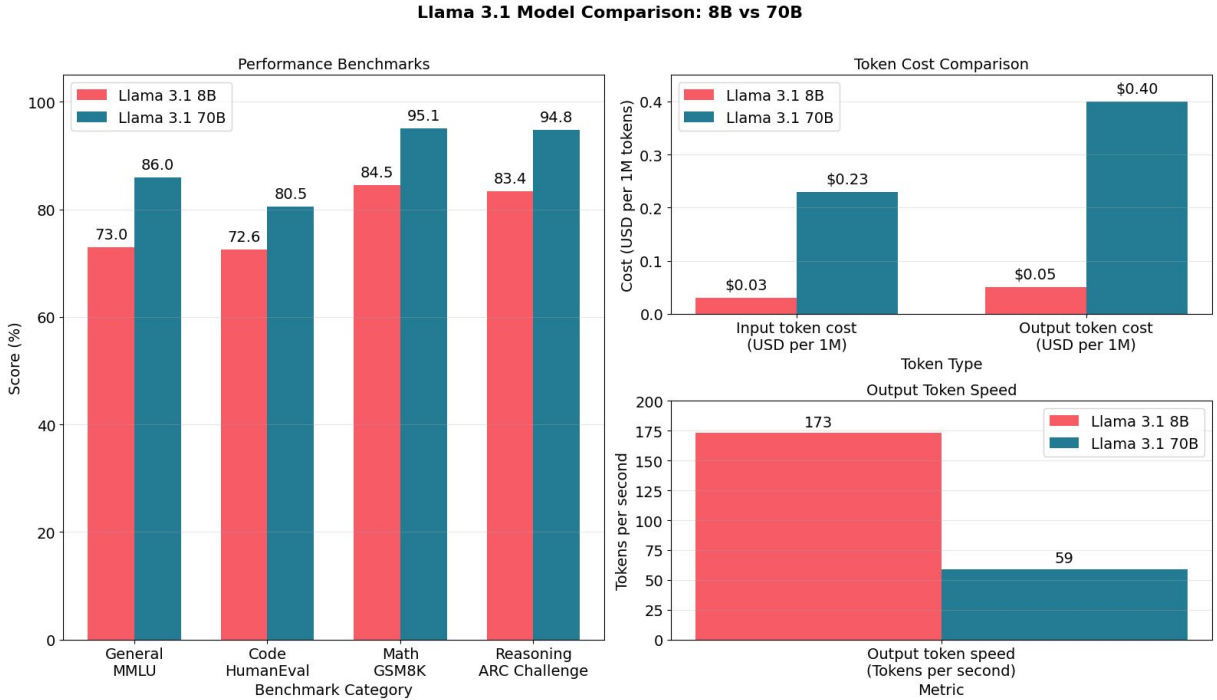
🧩 Prefix Caching, Tensor Parallelism, Pipeline Parallelism

⚡ High throughput, low latency

📐 Structured generation

🔗 Function calling for complex workflows

Big vs small models

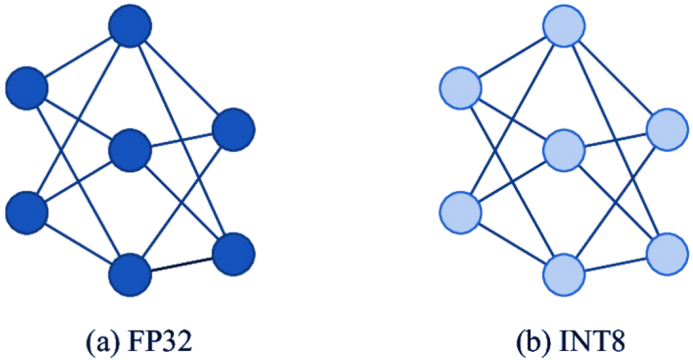


8B Fast & Cheap - > High-Volume Apps

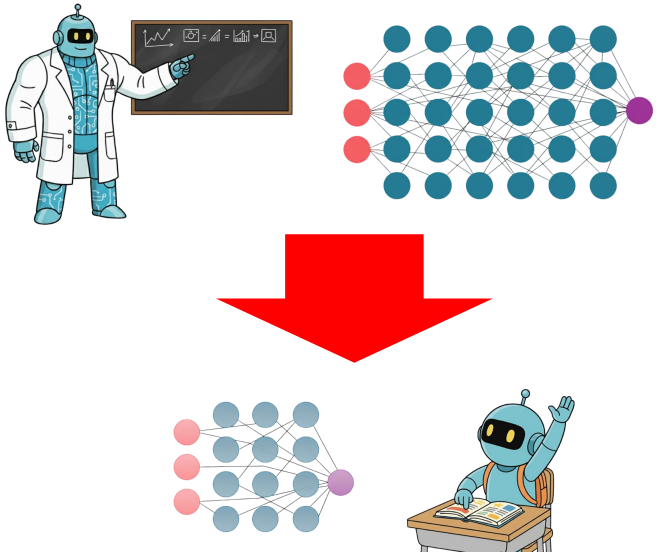
70B More powerful, but costly -> Complex Reasoning Tasks

Optimization & distillation

Quantization

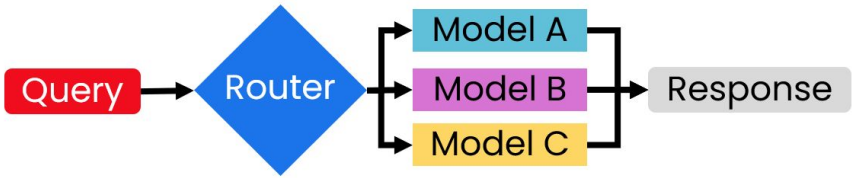


Knowledge Distillation

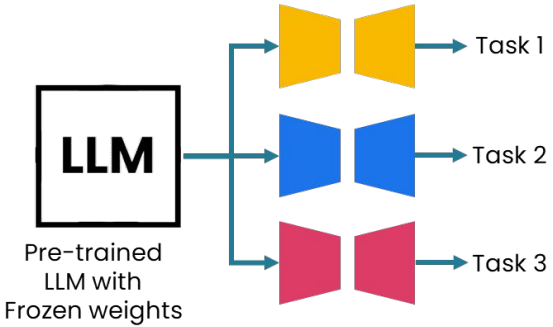


Routing

LLM Model Router



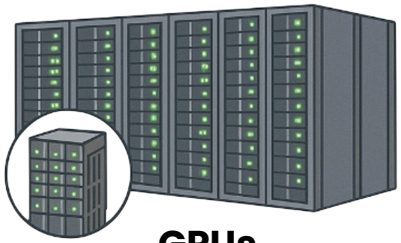
Multi-LoRA serving



Compute

⏱️ Long training time
💸 High cost
📦 Big capacity

RL



GPUs

LoRA Fine-tuning

AI PCs

⚡ Quick turnaround
💰 Low cost
🎯 Targeted scope

Heavy compute — Light compute

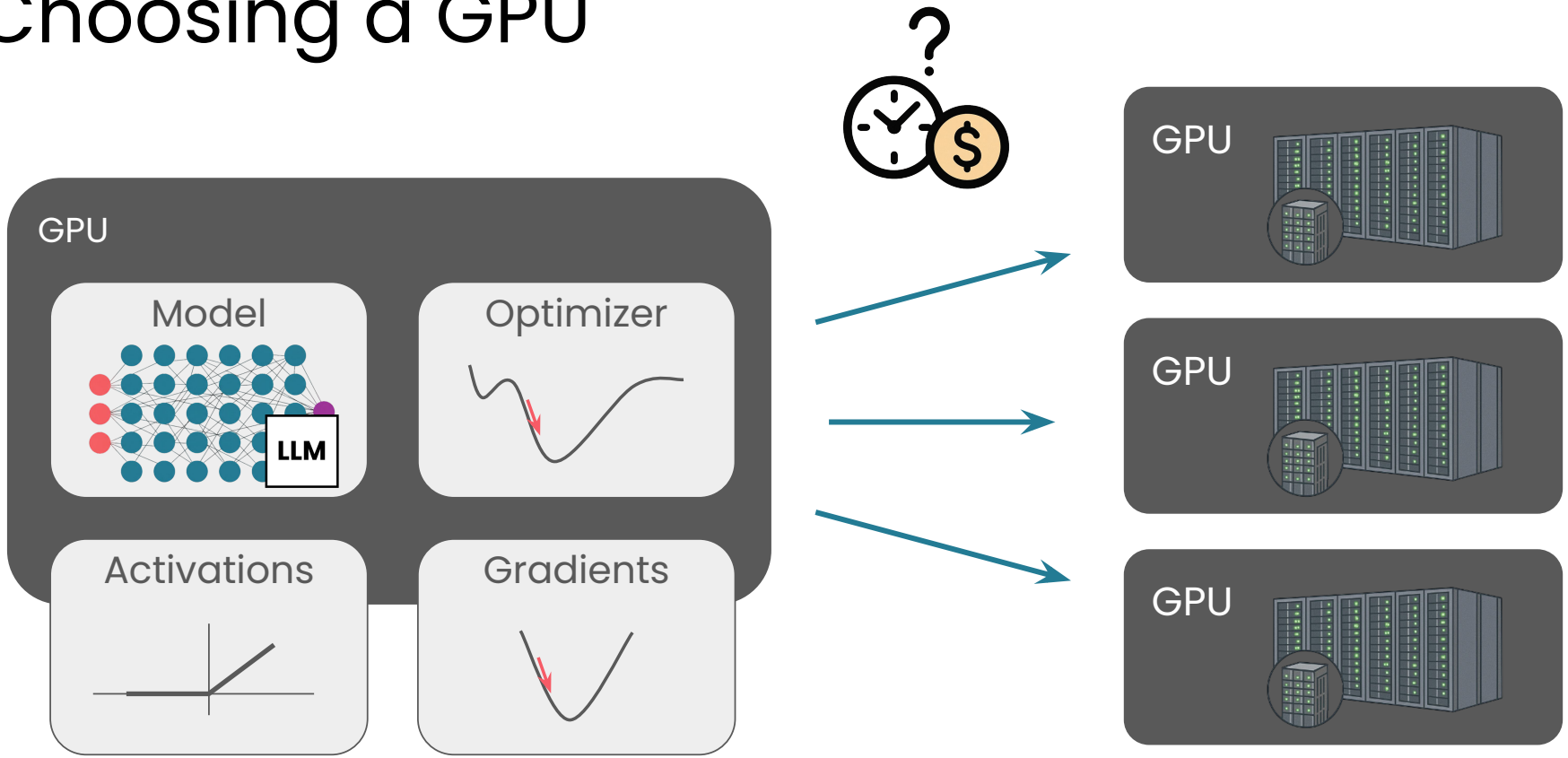
Scale of post-training changes

DeepLearning.AI

Production considerations

Infrastructure (Part 2)

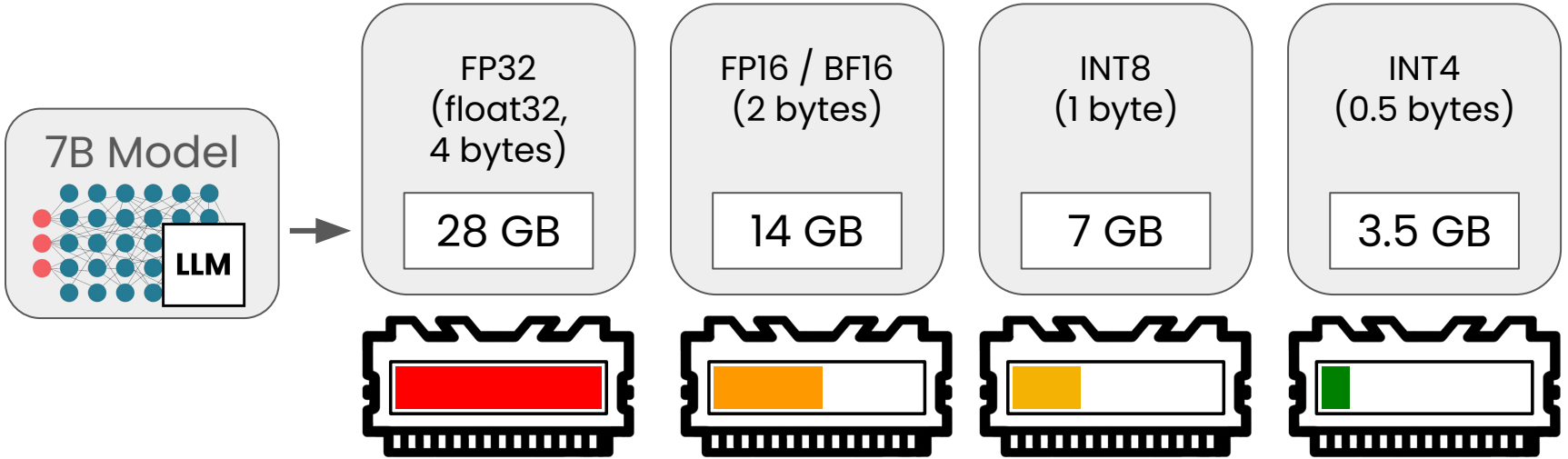
Choosing a GPU



Choosing a GPU

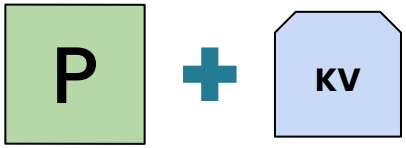
Precision and memory per param

Memory = # params × bytes_per_param



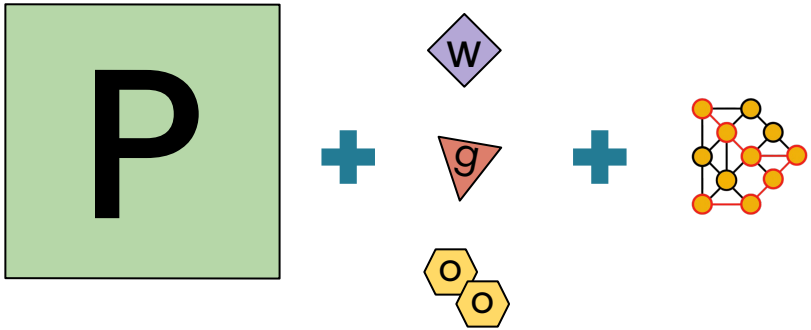
Choosing a GPU

Inference only



~1× parameters + KV cache

Training (AdamW)



4× parameters (weights + grads + 2 optimizer states) + activations.

Multiply parameters by memory_per_param in precision.

Choosing a GPU

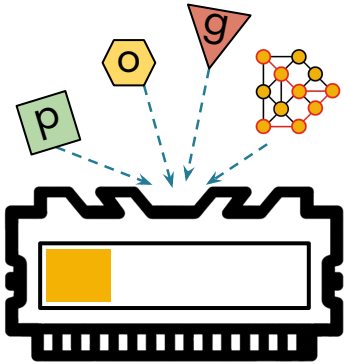
On a GPU with limited memory (VRAM):

- LoRA avoids storing grads/optimizers for most weights → big savings.
- QLoRA also quantizes frozen base model → even smaller footprint.

GPUs with larger memory capacity can fit bigger models without splitting the model onto multiple GPUs (“sharding”).

Choosing a GPU

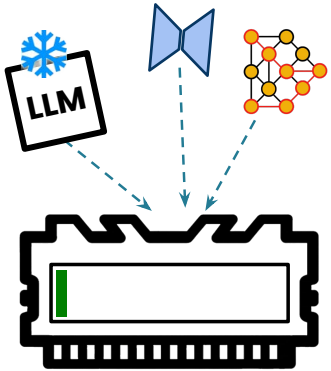
Fine-Tuning



Memory: All 4 components (params + grads + optimizer + activations).

Need: Max VRAM, often >>80 GB per GPU for 13B+ models.

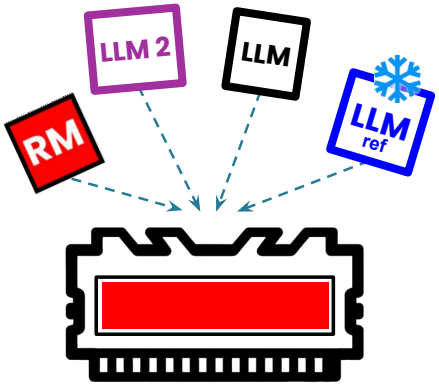
LoRA



Memory: Only adapter weights + activations; base model frozen.

Need: ~10–20% of fine-tuning footprint.

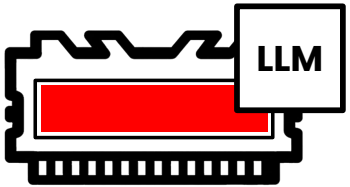
RL



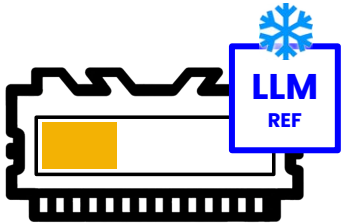
Memory: Multiple models in memory (LLM + LLM2 + LLM_ref + reward model) for PPO, less for GRPO.

Need: 2–4× fine-tuning memory footprint.

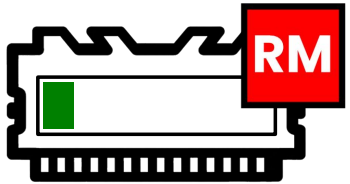
Choosing a GPU



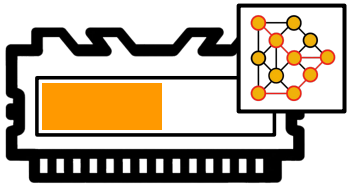
LLM gets same as fine-tuning
+26*4 = +104 GB



Reference: +26 GB (inference only, no training)



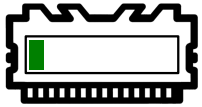
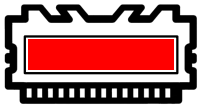
Reward Model: (varies, assume ~7B-14 GB)



Activations: ~30–50 GB depending on batch × seq length, from group rollouts

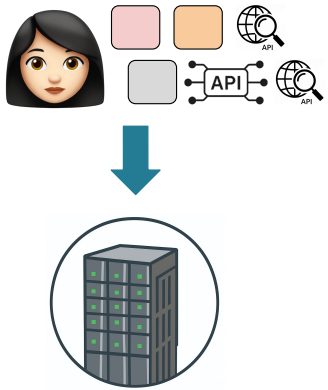
GRPO Memory Breakdown (for FP16, 13B model)
👉 Total ≈ 170–190 GB VRAM for a 13B LLM with GRPO.

Choosing a GPU

	Fine-tuning	LoRA	RL
Memory Focus	 Full memory stack required	 Adapters only; fits mid-tier GPUs	 Multiple copies; biggest memory hog
Compute Focus	 High throughput training	 Moderate; efficient training	 Extreme compute + throughput required.

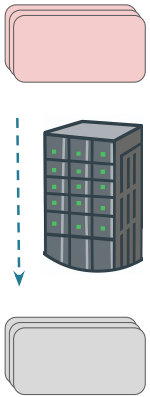
Capacity planning

Traffic



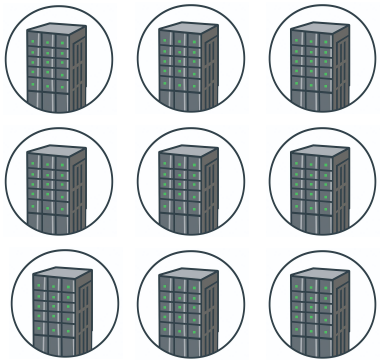
QPS × avg request tokens (input+context+tools) × output tokens.

Throughput



Pick batch size & expected tokens/s per GPU.

Hardware



Count GPUs for p95 target; include headroom for canary.

Example sizing (ballpark)

Target: 30 QPS, 1.2k prompt + 300 gen tokens avg, p95 < 900 ms.

With batching & FP8, ~60 tok/s/GPU effective at p95
→ Need ~10 GPUs (+30% headroom) for steady-state and canary
→ At $1.99/hr:
$19.90/hr
$477.60/day
$14,328/month

$$\text{cost}/1k \approx \frac{\$/\text{GPU-hr}}{\text{tokens/s/GPU} \times 3600} \times 1000$$

DeepLearning.AI

Production Considerations

Production-ready checklist

Checklist

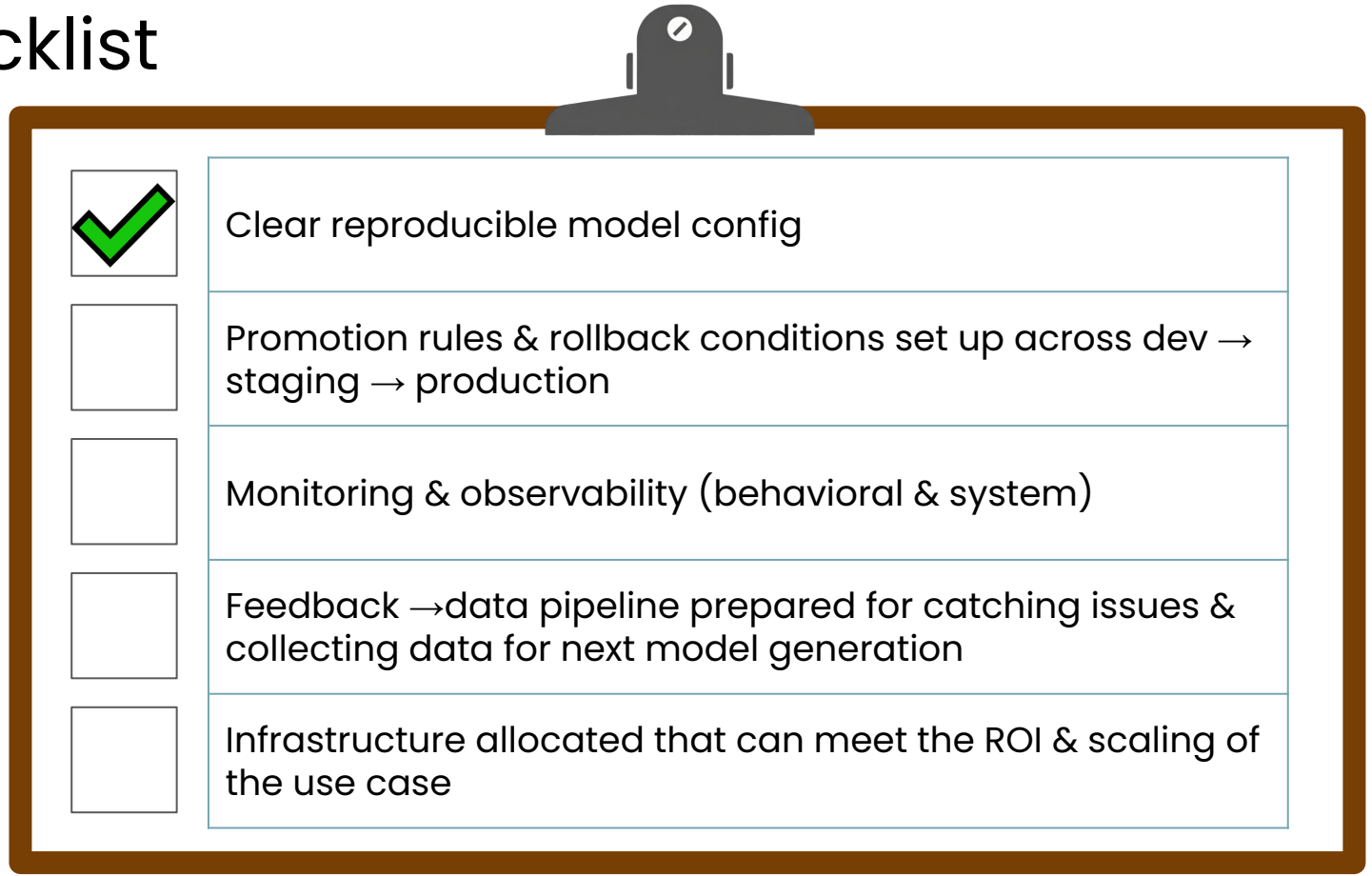
- [ ] Clear reproducible model config
- [ ] Promotion rules & rollback conditions set up across dev → staging → production
- [ ] Monitoring & observability (behavioral & system)
- [ ] Feedback →data pipeline prepared for catching issues & collecting data for next model generation
- [ ] Infrastructure allocated that can meet the ROI & scaling of the use case

Clear reproducible model config

Goal: anyone can re-run the held-out suite and get the same numbers; anyone can roll back to the exact artifact (within CI tolerance).

```
{
  "llm_sha": "LLM@abc123",
  "adapters": ["lora_med_safety_v2@e91f", "lora_math_v1@bb07"],
  "input_sha": "INPUT@p1e3", "tokenizer_sha": "TOK@t99",
  "sampling": {"temperature":0.2, "top_p":0.9},
  "reward_model_sha": "RM_EVAL@9af2",
  "verifiers": {"json_schema":"v4.3", "span":"v2.1", "unit_tests":"v2.0"},
  "tools": {"search_api":"3.2.1", "fixtures":"search_v3.json",
"repo_sha":"GIT_SHA_CODEBASE_V3"}
}
```

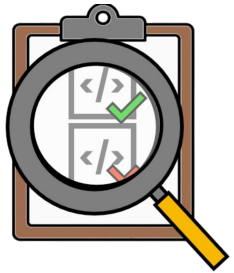
Checklist



☑	Clear reproducible model config
☐	Promotion rules & rollback conditions set up across dev → staging → production
☐	Monitoring & observability (behavioral & system)
☐	Feedback →data pipeline prepared for catching issues & collecting data for next model generation
☐	Infrastructure allocated that can meet the ROI & scaling of the use case

Promotion rules & rollback conditions set up

Goal: Turn North Star into contract

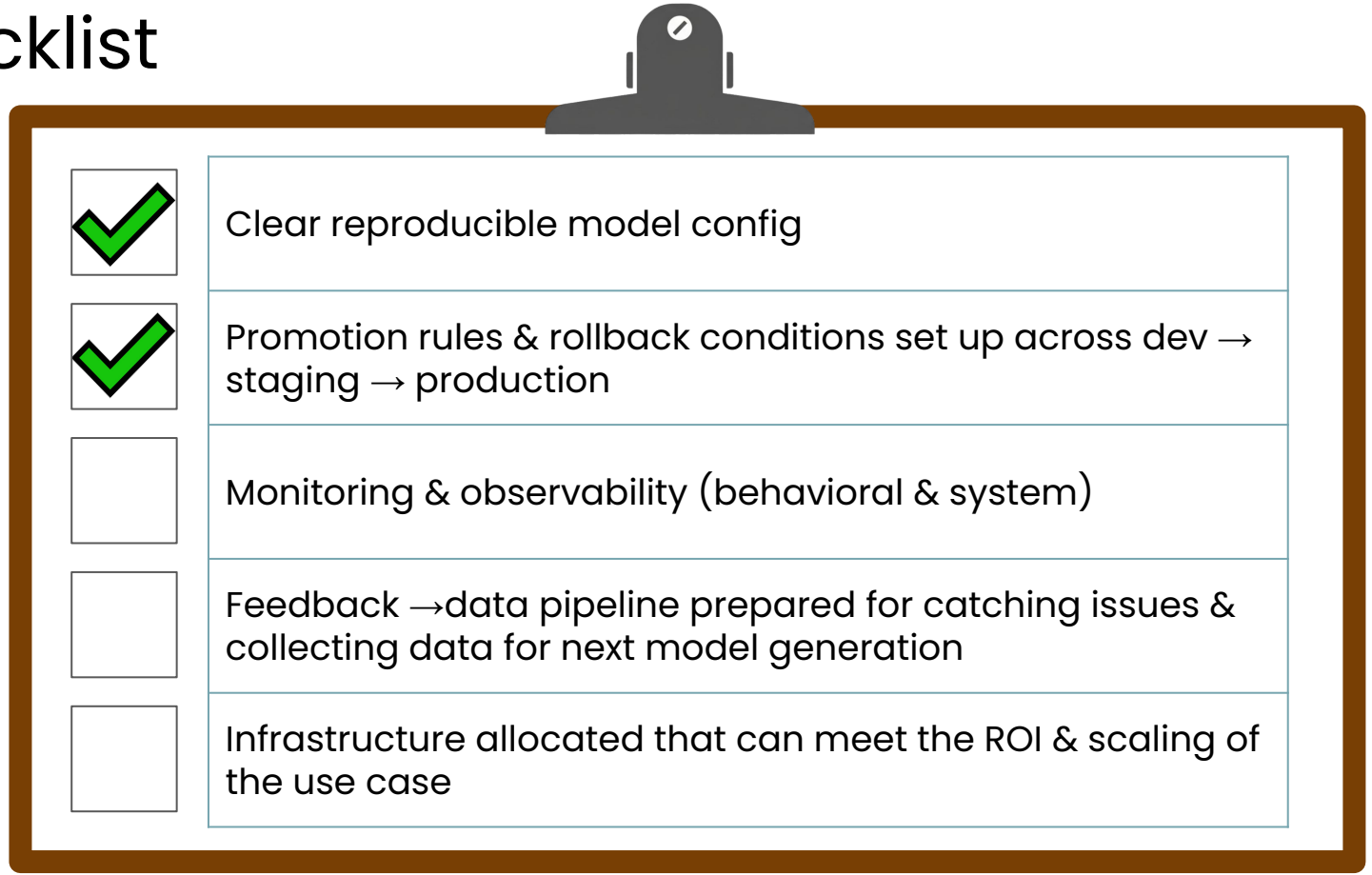


Turn your North Star into a coded go/no-go contract

Add light live guardrails in staging

Pre-agree automatic rollback triggers

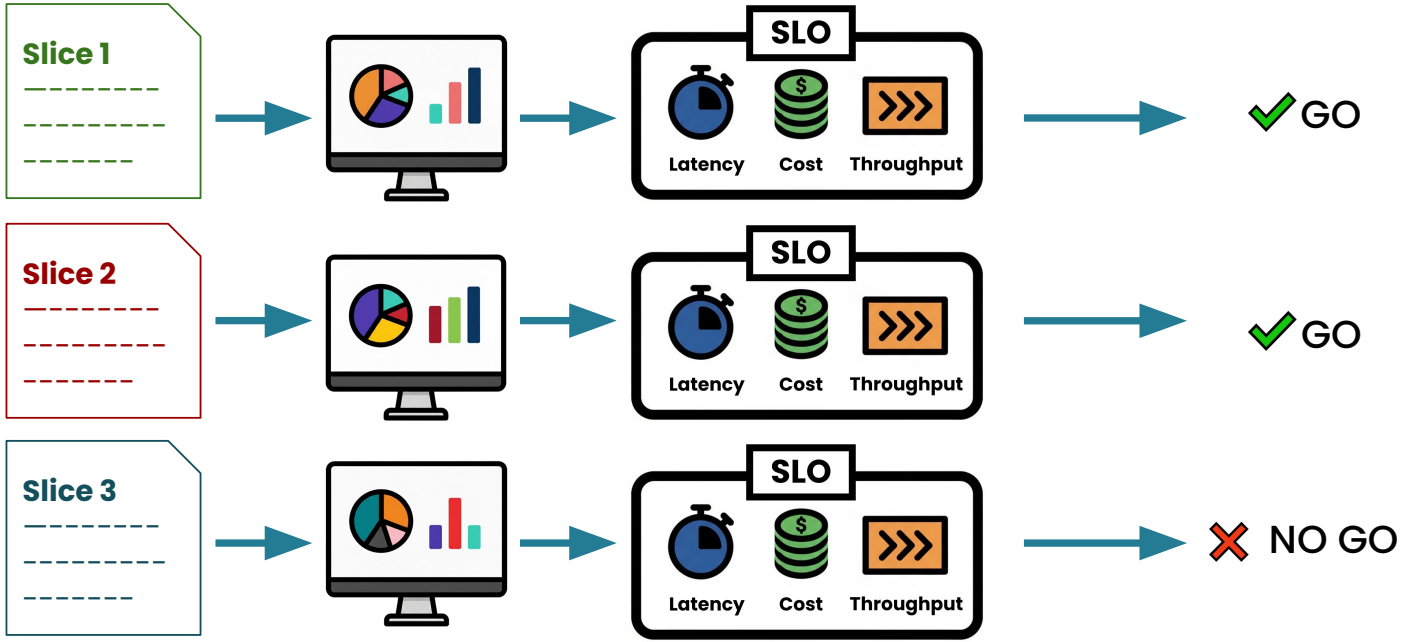
Checklist



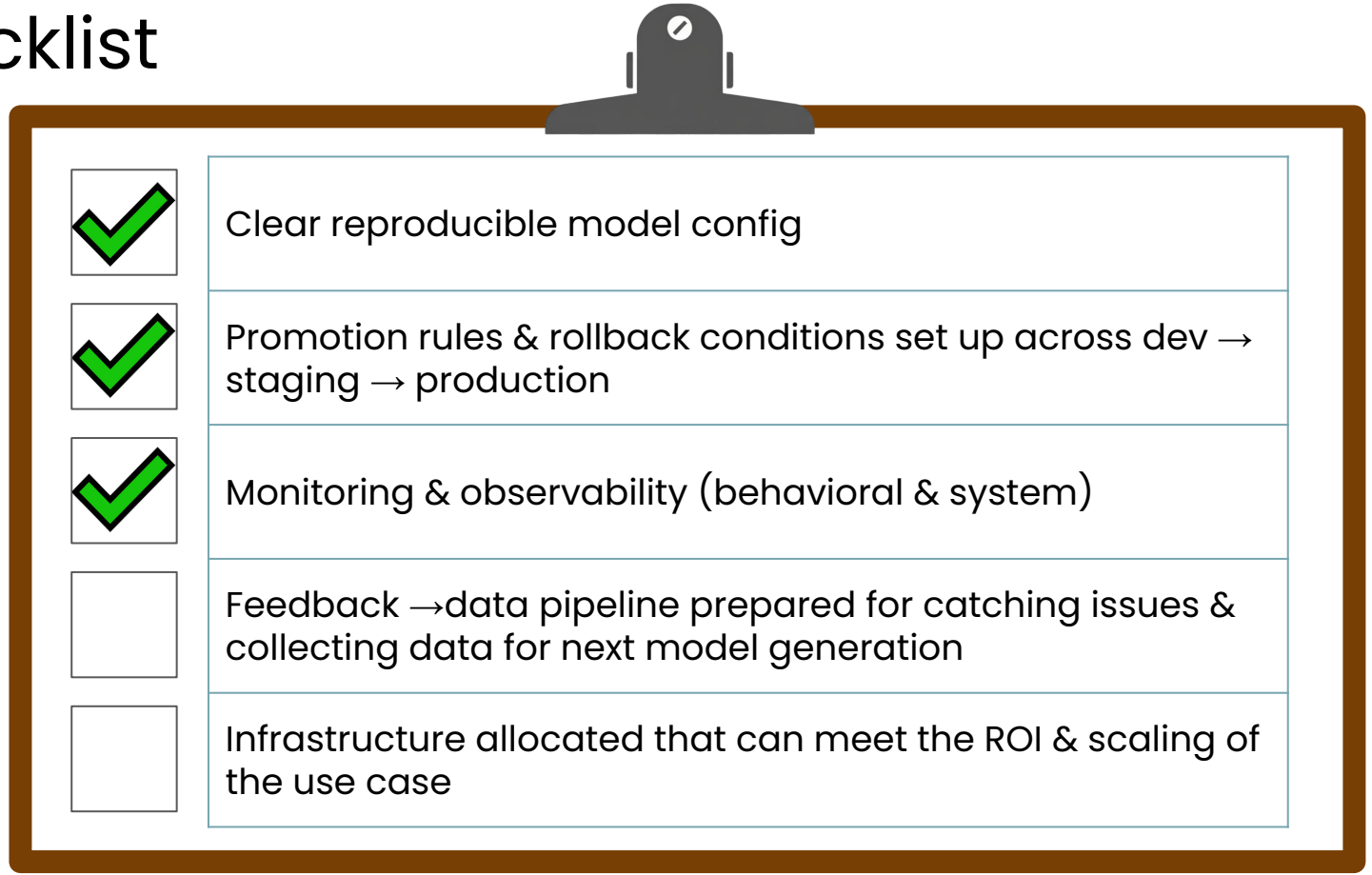
☑	Clear reproducible model config
☑	Promotion rules & rollback conditions set up across dev → staging → production
☐	Monitoring & observability (behavioral & system)
☐	Feedback →data pipeline prepared for catching issues & collecting data for next model generation
☐	Infrastructure allocated that can meet the ROI & scaling of the use case

Monitoring & observability

Goal: Make behavior visible and actionable



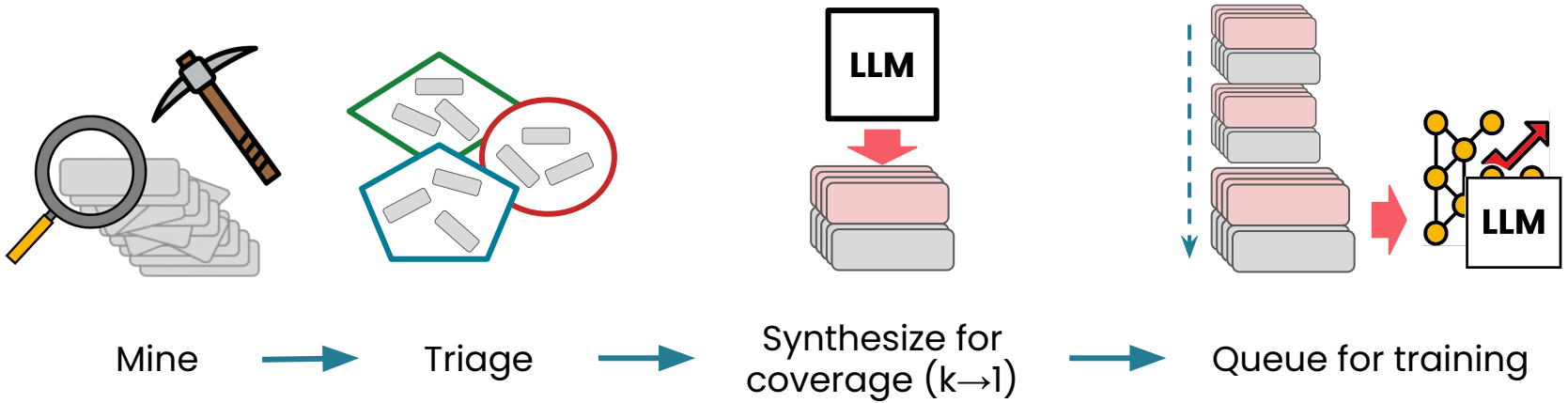
Checklist



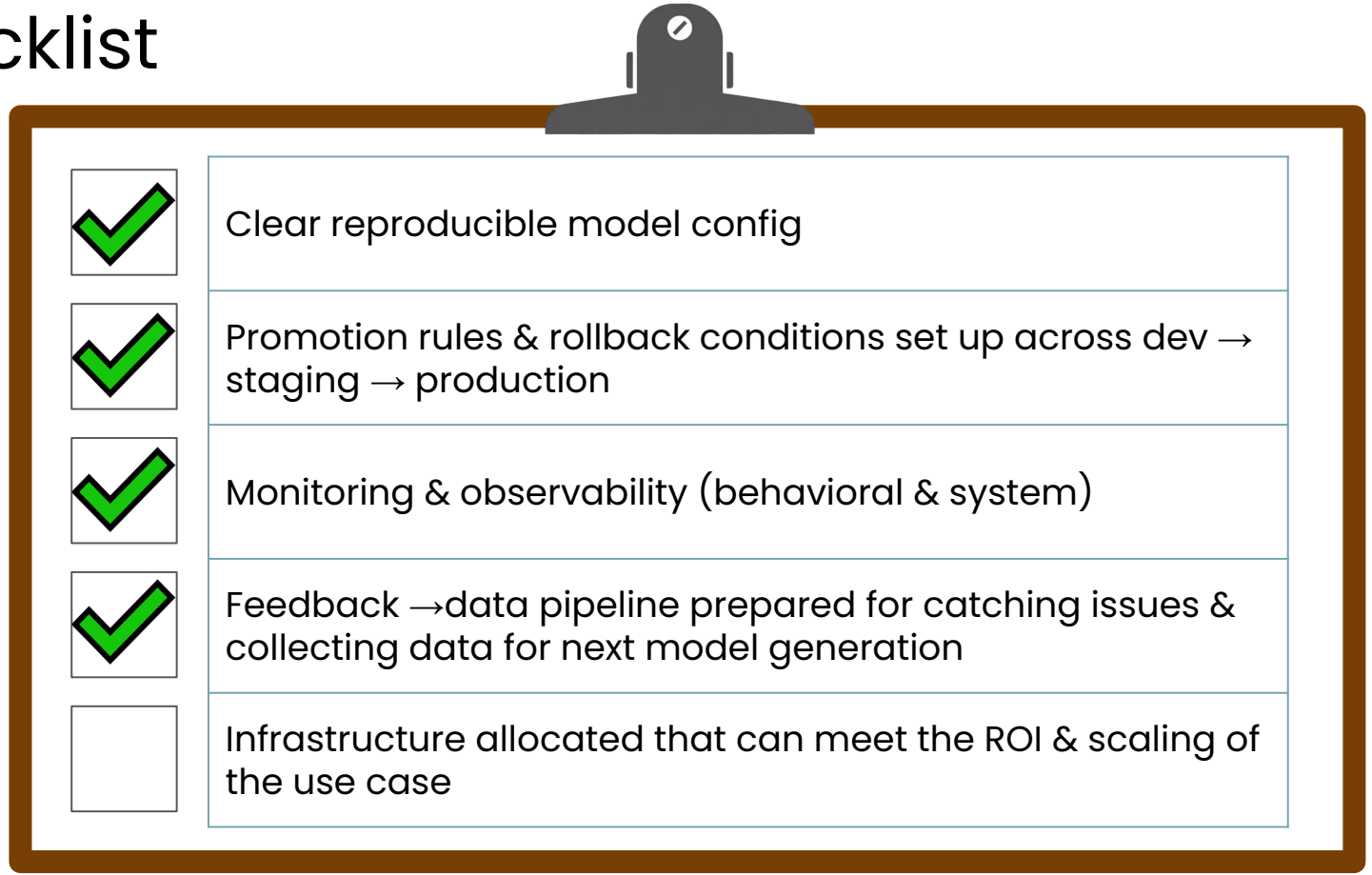
☑	Clear reproducible model config
☑	Promotion rules & rollback conditions set up across dev → staging → production
☑	Monitoring & observability (behavioral & system)
☐	Feedback →data pipeline prepared for catching issues & collecting data for next model generation
☐	Infrastructure allocated that can meet the ROI & scaling of the use case

Feedback →data pipeline prepared

Goal: Turn real failures into high-signal supervision for the next fine-tuning/RL run



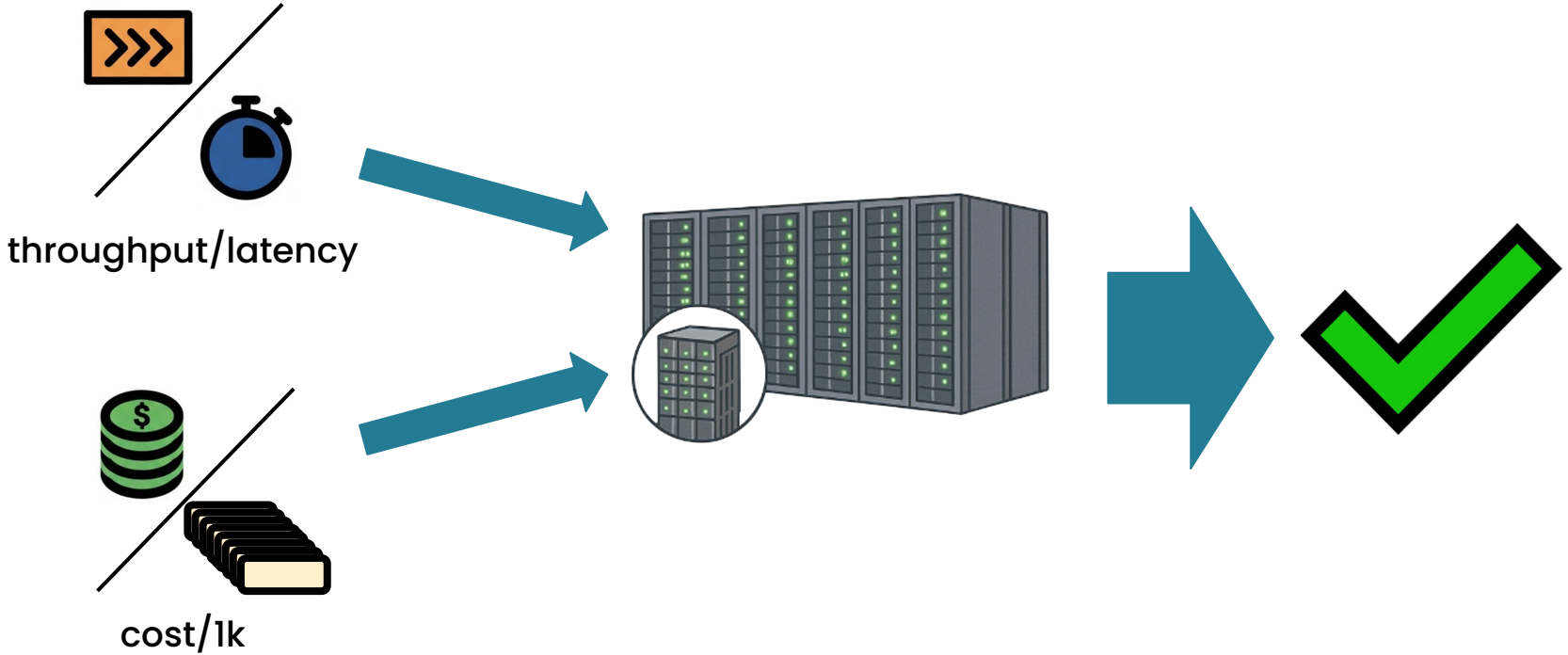
Checklist



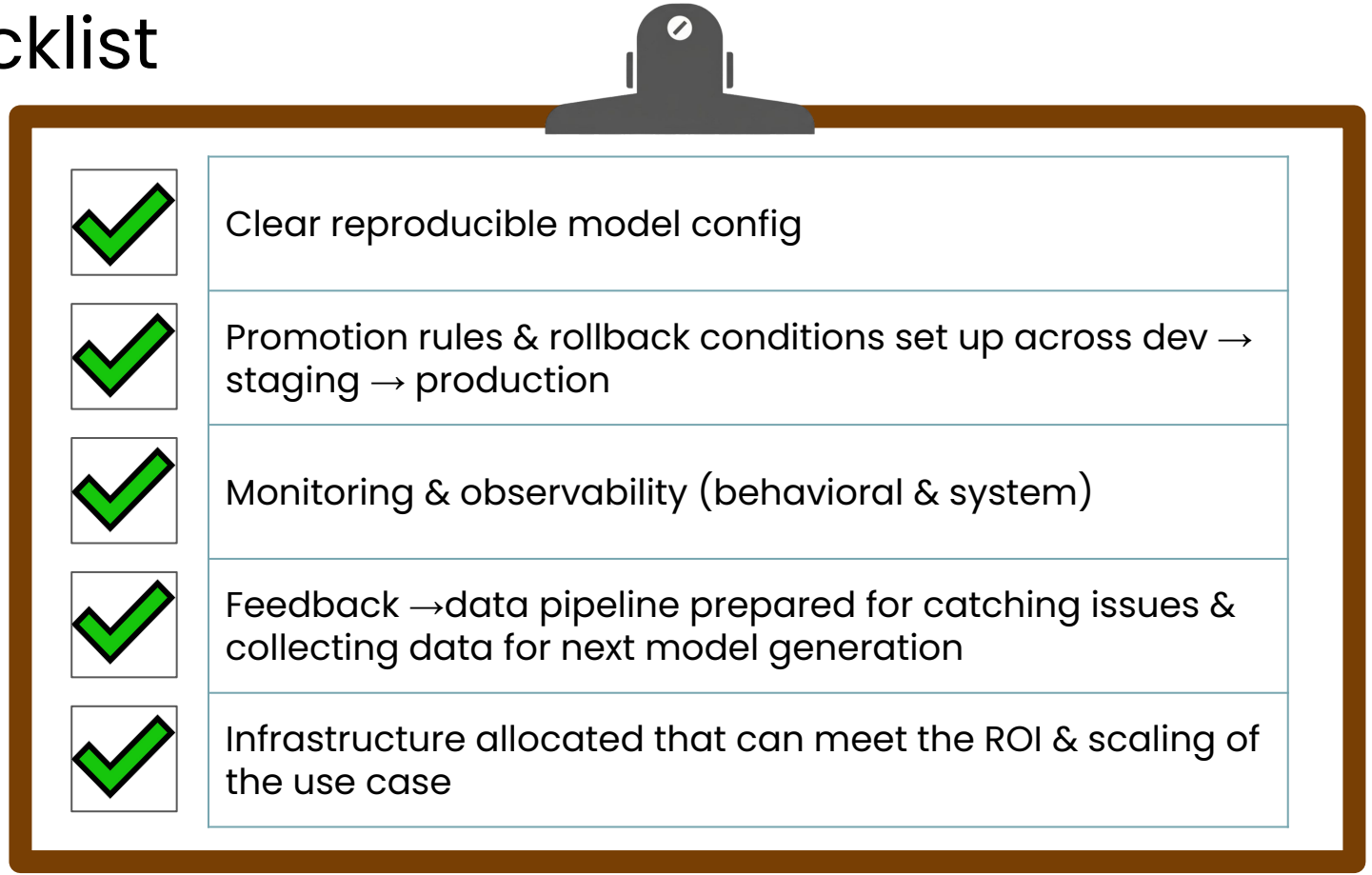
☑	Clear reproducible model config
☑	Promotion rules & rollback conditions set up across dev → staging → production
☑	Monitoring & observability (behavioral & system)
☑	Feedback →data pipeline prepared for catching issues & collecting data for next model generation
☐	Infrastructure allocated that can meet the ROI & scaling of the use case

Infrastructure allocated

Goal: meet targets on GPU scaleout, without regressing on important behaviors.



Checklist



- [x] Clear reproducible model config
- [x] Promotion rules & rollback conditions set up across dev → staging → production
- [x] Monitoring & observability (behavioral & system)
- [x] Feedback →data pipeline prepared for catching issues & collecting data for next model generation
- [x] Infrastructure allocated that can meet the ROI & scaling of the use case