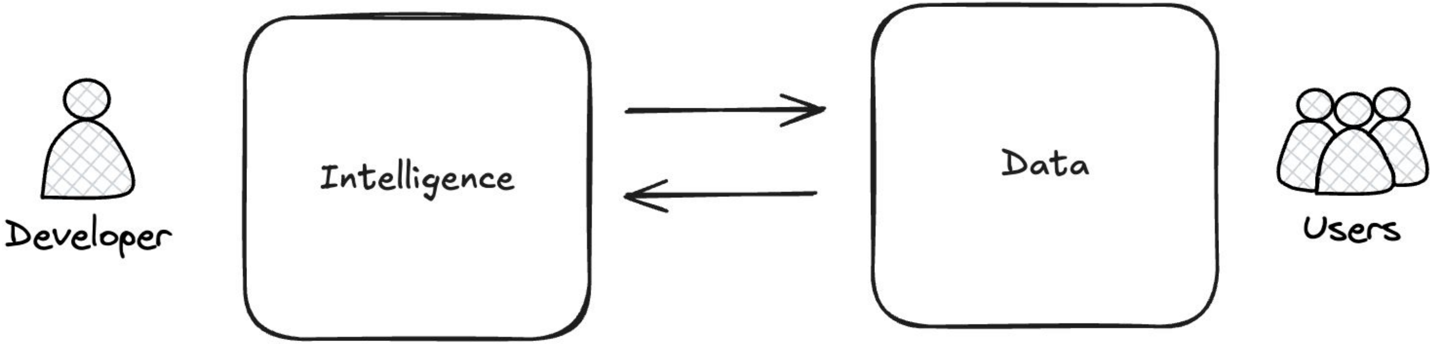
Workstream#2 Data & Algorithm Updates (Sep-09-2025)

LF InfiniEdge AI Workstream#2: data and algorithms

- Focus on AI applications edge computing & use cases (technological & business)
- Pushing the boundary of LLMs with data (on-prem) and algorithms
    - Retrieval accuracy 20% ⇒ 70+% (Complex RAG KDD 2024)
    - Coding capability 25% ⇒ 45% (Reflections) ⇒ 90% (Human Hints) (AI Competitive Coding Neurips 2024)
- Edge Data Agent – Data-on-prem, Code-on-the-Fly



Previous Key Learnings – SFT, RAG & Reasoning

Foundational model dominates the performance (trust scaling law)

SFT change styles and format but not adding new knowledges

RAG (2024 KDD Cup Complex Retrieval Challenges)

- Accuracy 20% ⇒ 70+% (Fine Tune / Data Preproc.)
- Battling with the model's memory

Reasoning (2024 NeuRIPs AI for Competitive Coding)

- Accuracy 25% ⇒ 45% (Reflections) ⇒ 90% (Human Hints)
- Agents x50 cost

EDA converts a user's *own* **data into** a *local,* on-demand **agent** service at the **edge**

Data-on-prem, Code-on-the-Fly

Leaderboards

SWE-agent-LM-32B, the open-weight SotA on Verified, trained on synthetic data generated by **SWE-smith**. More in the paper!

Bash Only | Verified | Lite | Full | Multimodal

Bash Only evaluates all LMs with a minimal agent on SWE-bench Verified (details)

Filters: Open Scaffold ▾ All Tags ▾

Model	% Resolved	Org	Date	Logs	Trajs	Site	Release
🆕✅ Claude 4 Opus (20250514)	67.60		2025-08-02	✓	✓		–
🆕✅ GPT-5 (2025-08-07) (medium reasoning)	65.00		2025-08-07	✓	✓		–
🆕✅ Claude 4 Sonnet (20250514)	64.93		2025-07-26	✓	✓		–
🆕✅ GPT-5 mini (2025-08-07) (medium reasoning)	59.80		2025-08-07	✓	✓		–
🆕✅ o3 (2025-04-16)	58.40		2025-07-26	✓	✓		–
🆕✅ Qwen3-Coder 480B/A35B Instruct	55.40		2025-08-02	✓	✓		–
🆕✅ Gemini 2.5 Pro (2025-05-06)	53.60		2025-07-26	✓	✓		–
🆕✅ Claude 3.7 Sonnet (20250219)	52.80		2025-07-20	✓	✓		–
🆕✅ o4-mini (2025-04-16)	45.00		2025-07-26	✓	✓		–
🆕✅ Kimi K2 Instruct	43.80		2025-08-07	✓	✓		–

(Previously) Single Agent.yaml

src > prompts > ! custom_agent.yaml

```yaml
system_prompt: |-
  You are an expert data analysis assistant who can perform any type of data exploration, processing, or transformation task using co
  You can leverage Python code within your responses to operate on data, explore patterns, produce visualizations, and answer complex

  You have access to a set of specialized tools (Python functions) which you can call by providing a code block in your response.
  The steps to solve any problem must follow these sequences:

  1. **Thought**
     Provide a concise explanation of your reasoning and the approach or tools you're about to use.

  2. **Code**
     Provide a Python code snippet (enclosed in triple backticks) ending with `<end_code>`.
     - Within these code snippets, you can call any relevant tool or write standard Python code.
     - The output of your code snippet is then displayed in the **"Observation"** field of the next step.

  3. **Observation**
     Show and summarize the results from your code snippet execution.

  You can do multiple cycles of Thought → Code → Observation if needed, until you have a definitive solution.

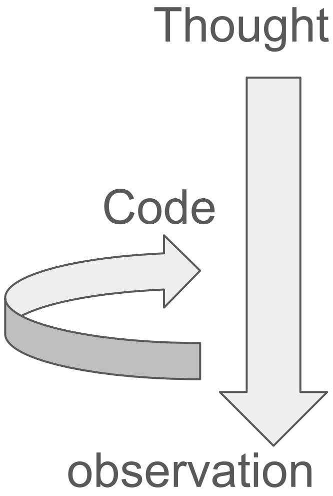
  **Final Answer**
  Once you have enough information to produce your conclusion or final data insights, provide your answer using the `final_answer` to

  ---

  **Example**
  **Task**: "Write SQL code to retrieve information from the SQLite database `'data/mydata.db'`."

  - **Thought**:
    "I will import the `SQLiteTool`, run the agent with the query, and then provide the result."

  - **Code**:
    ```py
    from smolagents import ToolCallingAgent, LiteLLMModel
    from tools.tool_sqlite import SQLiteTool
```



Initial plan is to follow the smolagent template to craft a single (all-in-one) system prompt for edge data agent and together with hardcoded functions and tools (e.g. data connectors, mcp server, …)
```

Git Update: adding EDA Eval Suite

Evaluation Results

Task	General LLM	Local Agent	Δ (Agent – LLM)
p1_finance_invoice_match	0.33	1.00	0.67
p1_hr_post_termination	0.00	1.00	1.00
p1_ops_spike	0.00	1.00	1.00
p2_emails_discount_thread	0.00	1.00	1.00
p2_audio_merge	0.17	1.00	0.83
p2_finance_fx	0.00	0.40	0.40
p3_ocr_invoice	0.00	1.00	1.00
p3_sql_recon	0.00	1.00	1.00
p4_eml_attachments	0.00	1.00	1.00
p4_xlsx_summary	0.00	0.00	0.00
Averages	**0.05**	**0.84**	**0.79**

Code Blame 22 lines (20 loc) · 899 Bytes

```python
# Implement `run(pack_dir: str, prompt: str) -> dict | str`
# - pack_dir: path to the located pack folder (so your agent can read files)
# - prompt: same text as given to general LLM
# Return JSON-compatible Python object, or a JSON string.
#
# TIP: You can route on task intent by reading files under `pack_dir`.
#  e.g., if 'answers.json' exists -> Pack1; 'answers_pack2.json' -> Pack2; etc.

import os, json

def run(pack_dir: str, prompt: str):
    # TODO: Wire your local tools. Below is a tiny heuristic demo:
    # This demo simply returns {}. Replace with real logic.
    # You can detect files under pack_dir and parse:
    # - CSVs in finance/, hr/, ops/
    # - .ics in calendar/
    # - PDFs in pdfs/ or pdf_scans/ (via OCR)
    # - PBM images in scans/ (Pack 3) -> OCR
    # - .eml and .mbox
    # - SQLite in product/ or sql/
    # - TAR/ZIP nested archives in archives/
    return {}
```

Next

More complex eval data (real-time data, larger files, huggingface repo)

New suite of data to challenge the memory conflicts of LLMs

- When local data doesn’t match with the LLM’s memory

More complex retrieval tasks (reasoning, and multi-step queries)

LLM researcher's toolkit

Tools:

- Training – Pytorch Lightning + Megatron (for LLM)
- Inference – vLLM, SGLang
- RL – veRL (Megatron + vLLM + Ray), SkyRL

Challenges/Opportunities:

- Inference merging into the training stage for RL
- System ⇒ more complexity, asynchronous, distributed …
- Workload ⇒ multi-modal, long-sequences, sparse, streaming, dynamic

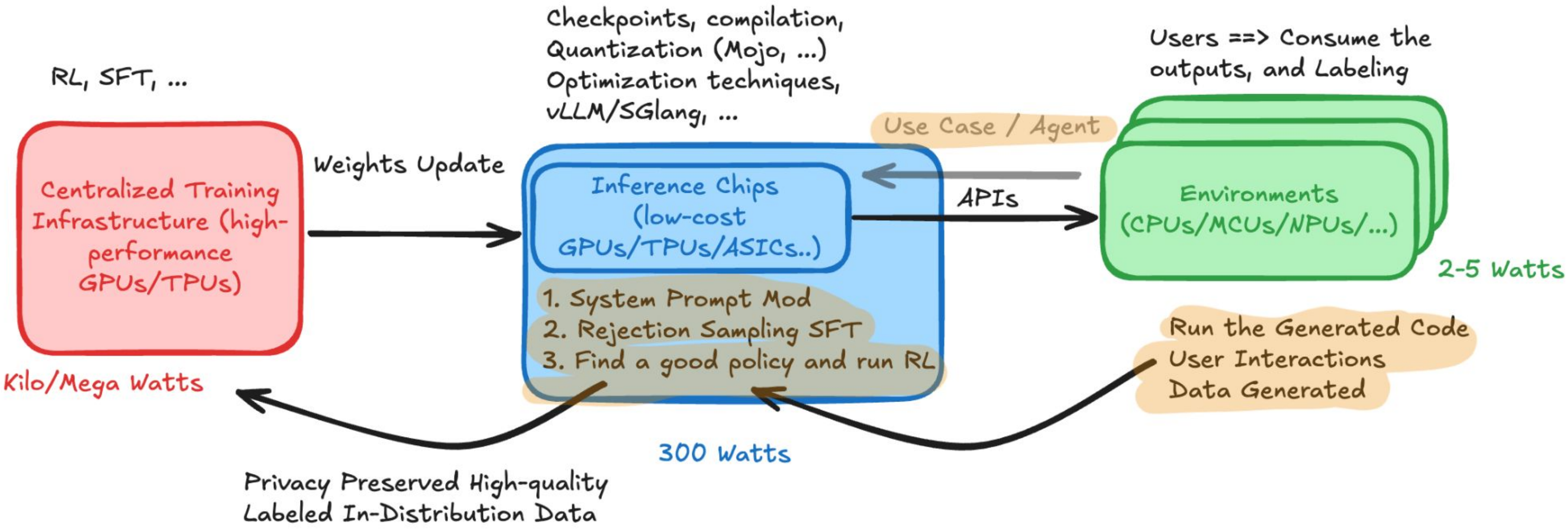
Goals of LLM Inference infrastructure

Speed – Achieve low latency (TTFT, TPOT)

Efficiency – Achieve higher throughput (tokens/sec) and lower cost ($ / token)

Easy of Use – Support more models, Huggingface integration, API server …

Introducing feedback loop



- Build a pipeline using the off-the-shelf frameworks
- Test out a few use cases and see the performance *gain*

Release 2.2 Roadmap – Inference feedbacks & User Interactions

We want to keep **data** on the edge as much as possible

EDA needs to understand the environment (w/ or w/o the help from the user)

When user starts to query the data, the agent retrieve the **information**

Ways of retrieval can be templated (but not-rule-based / hardcoded)

To-do-list:

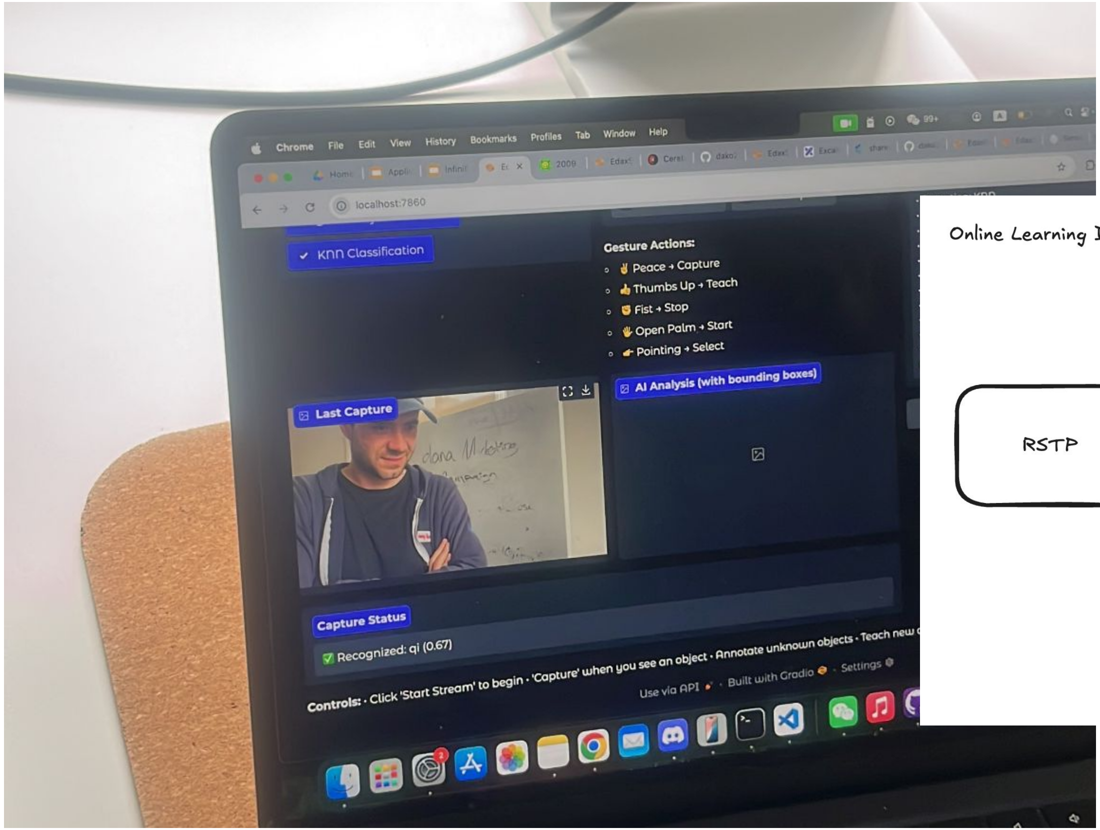
Upper level application –? distill the **knowledge**

Update (fine-tune) the model for better services

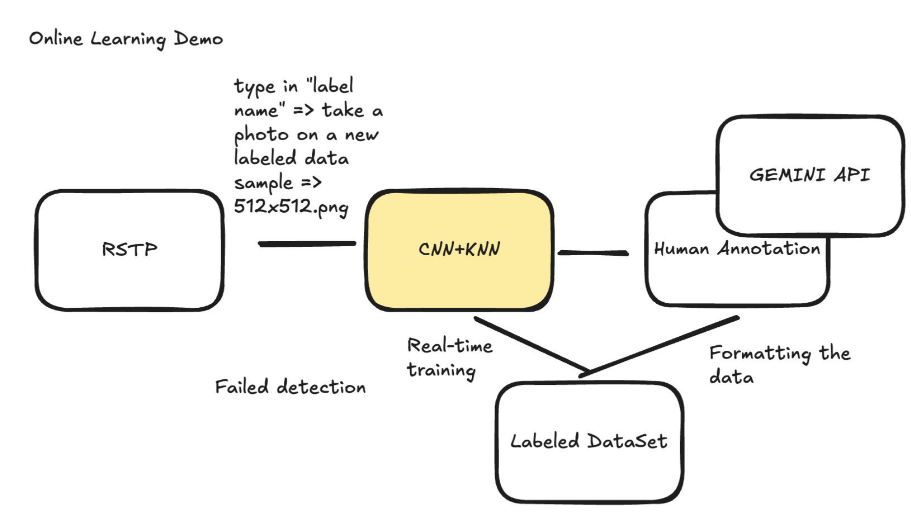
Three Layers

1. Data
2. Information ← EDA
3. Knowledge

Demo (contributor: @ebowwa)



EDA for online-learning



Sleeptime compute

During sleep, the human brain sorts through different memories, consolidating important ones while discarding those that don't matter [https://www.wired.com/story/sleeptime-compute-chatbots-memory/?utm_source=chatgpt.com] https://arxiv.org/pdf/2508.19828v2

Memory-R1

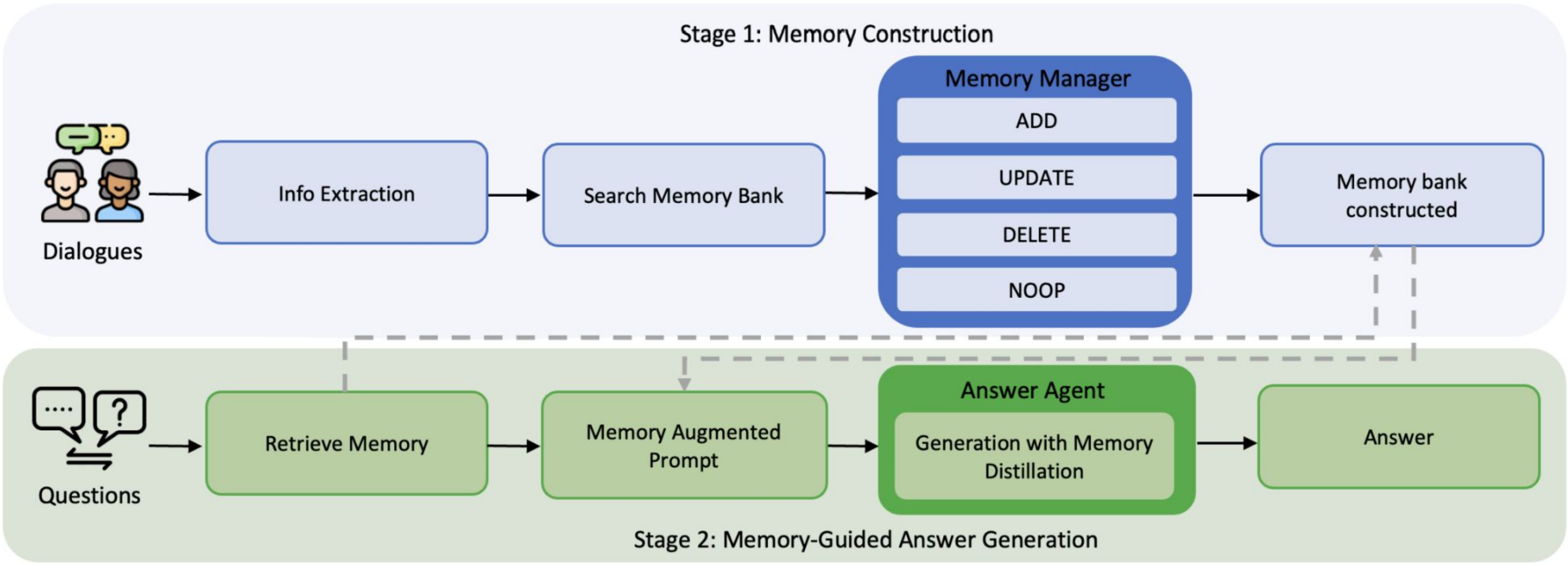


Figure 2: Overview of the Memory-R1 framework. Stage 1 (blue) constructs and updates the memory bank via the RL-fine-tuned Memory Manager, which chooses operations {ADD, UPDATE, DELETE, NOOP} for each new dialogue turn. Stage 2 (green) answers user questions via the Answer Agent, which applies a Memory Distillation policy to reason over retrieved memories.

Idea

If you achieve the end result with llm and record all the chats along the way, that means there is a path to there