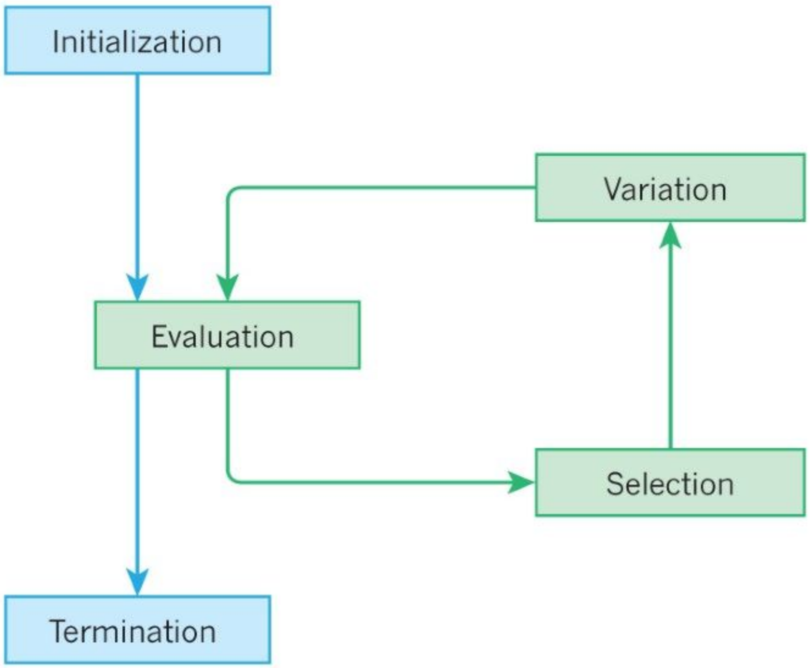
Evolutionary Algorithm

A gentle introduction

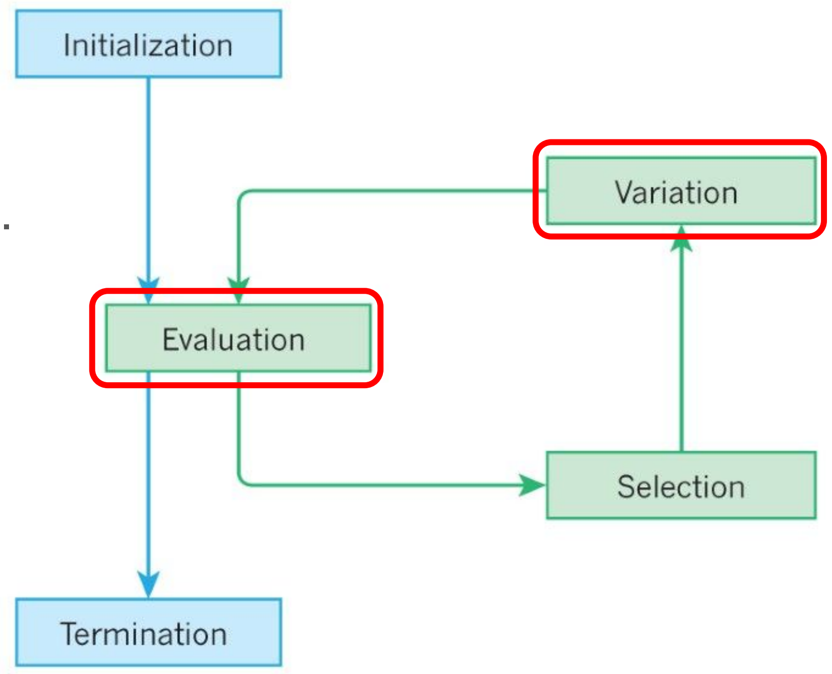
Evolutionary algorithm (EA)

- Based on the idea of biological evolution
- Maintain a population of structures that evolve according to evolution operator (e.g. mutation)
- Each individual structure is measured by fitness function (objective function)
- Selection focuses on high fitness structure -> natural selection



Evolutionary algorithm (EA)

- Based on the idea of biological evolution
- Maintain a population of structures that evolve according to evolution operator (e.g. mutation)
- Each individual structure is measured by fitness function (objective function)
- Selection focuses on high fitness structure -> natural selection



Comparison with Reinforcement Learning

	Reinforcement Learning	Evolutionary Algorithm
Goal	Optimize behavior to maximize certain target	
Data source	Sample based interaction/mutation	
# of policy	Typically a single policy	Maintain a population of policies
How to update	Policy gradient, value based, actor-critic…	Selection and mutation, gradient-free
Feedback signal	Stepwise feedback from value functions	Outcome-based feedback

SCIENCE

AlphaEvolve: A Gemini-powered coding agent for designing advanced algorithms

14 MAY 2025

By AlphaEvolve team

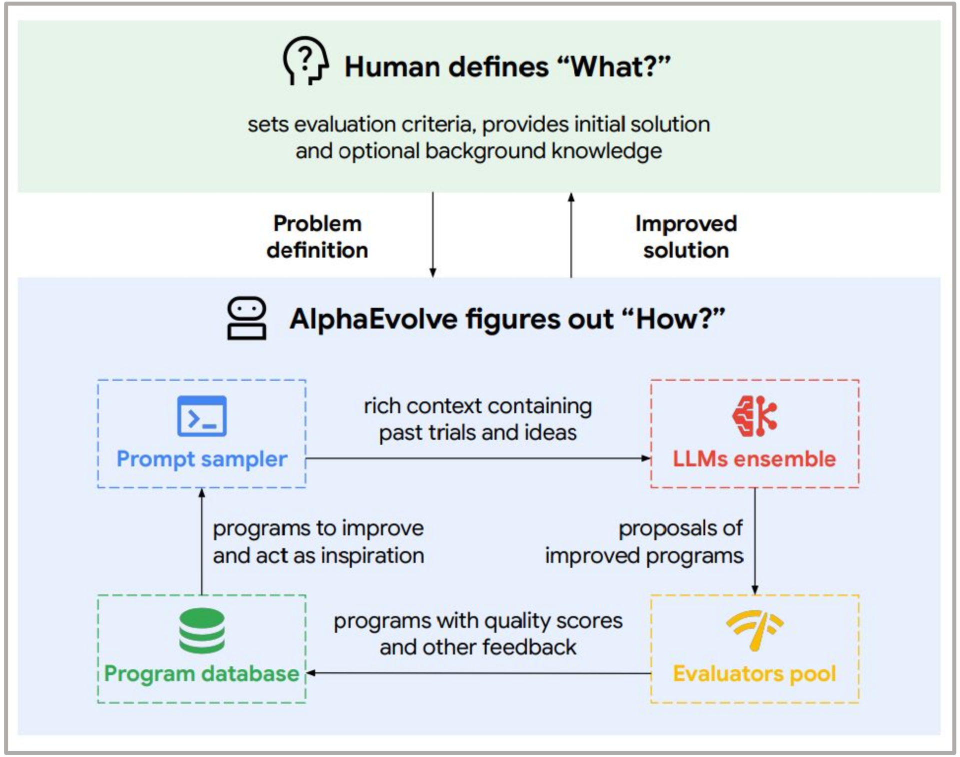
AlphaEvolve paper: https://arxiv.org/pdf/2506.13131

AlphaEvolve

- User: evaluation function and initial solution

- AlphaEvolve: look at past generations and by prompting, let LLM improve the program

- LLM: propose improved solutions



AlphaEvolve paper: https://arxiv.org/pdf/2506.13131

AlphaEvolve - additional details

- Employs two models: Gemini 2.0 Flash and Gemini 2.0 Pro
- Evaluation can be a combination between rule-based and LLM-based scores
- Only optimized for scheduling throughput but not speed in each step
- Will be given a compute budget for each run

AlphaEvolve paper: https://arxiv.org/pdf/2506.13131

AlphaEvolve - user provided file

```
# EVOLVE-BLOCK START
"""Image classification experiment in jaxline."""

import jax
...
# EVOLVE-BLOCK-END

...

# EVOLVE-BLOCK-START
class ConvNet(hk.Module):
  def __init__(self, num_classes): ...
  def __call__(self, inputs, is_training): ...


def sweep():
  return hyper.zipit([...])
# EVOLVE-BLOCK-END

...

def evaluate(eval_inputs) -> dict[str, float]:
  ...
  return metrics
```

AlphaEvolve - prompt

```
Act as an expert software developer. Your task is to iteratively
improve the provided codebase. [...]

- Prior programs

Previously we found that the following programs performed well
on the task at hand:

top_1_acc: 0.796; neg_eval_log_loss: 0.230; average_score: 0.513

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classess=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._conv2 = hk.Conv2D(num_channels * 2, kernel_shape=3)
    self._conv3 = hk.Conv2D(num_channels * 4, kernel_shape=3)
    self._logits_module = hk.Linear(num_output_classes)
[...]
```

```
- Current program

Here is the current program we are trying to improve (you will
need to propose a modification to it below).

top_1_acc: 0.862; neg_eval_log_loss: 0.387; average_score: 0.624

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classes=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
    self._logits_module = hk.Linear(num_output_classes)
[...]

SEARCH/REPLACE block rules:
[...]

Make sure that the changes you propose are consistent with each
other. For example, if you refer to a new config variable
somewhere, you should also propose a change to add that
variable.

Example:
[...]

Task
Suggest a new idea to improve the code that is inspired by your
expert knowledge of optimization and machine learning.

Describe each change with a SEARCH/REPLACE block.
```

AlphaEvolve - example output

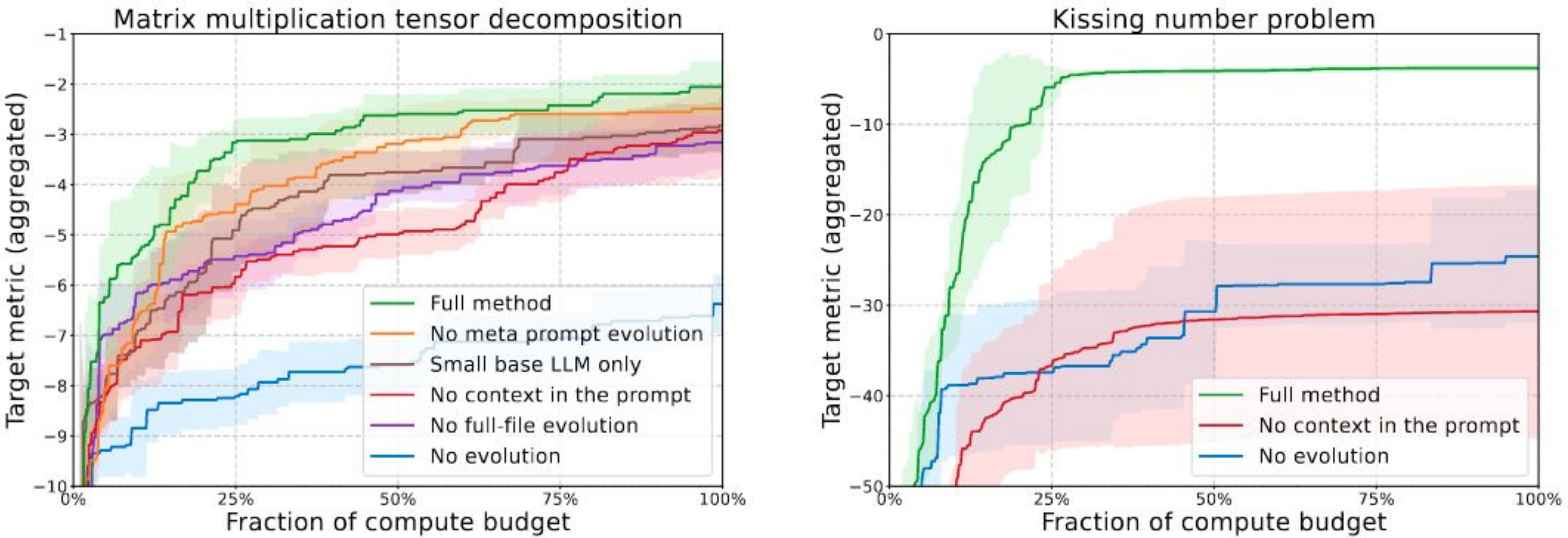
```
The current model uses a simple ResNet architecture with only
three ResNet blocks. We can improve its performance by
increasing the model capacity and adding regularization. This
will allow the model to learn more complex features and
generalize better to unseen data. We also add weight decay to
the optimizer to further regularize the model and prevent
overfitting. AdamW is generally a better choice than Adam,
especially with weight decay.


<<<<<<< SEARCH
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
=======
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels, stride=1)
    self._block3 = ResNetBlock(num_channels * 2, stride=2)
    self._block4 = ResNetBlock(num_channels * 2, stride=1)
    self._block5 = ResNetBlock(num_channels * 4, stride=2)
    self._block6 = ResNetBlock(num_channels * 4, stride=1)
>>>>>>> REPLACE
<<<<<<< SEARCH
  def optimizer(self, learning_rate):
    return optax.adam(learning_rate)
=======
  def optimizer(self, learning_rate):
    return optax.adamw(learning_rate, weight_decay=1e-4)
>>>>>>> REPLACE
```

AlphaEvolve - applications

AlphaEvolve has shown successful discoveries in

- Matrix multiplication
- Some mathematical problems
- Improving data center scheduling
- …

AlphaEvolve - Ablation study

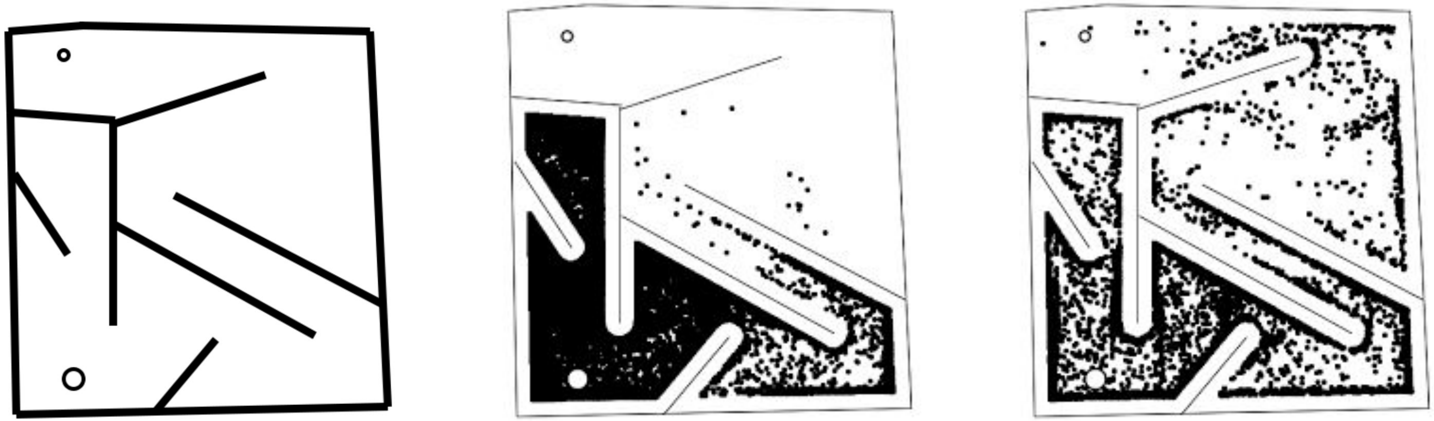


Novelty Search

A Paradox

If you try too hard to solve a hard problem, you'll fail

If you ignore the objective, you're more likely to success!



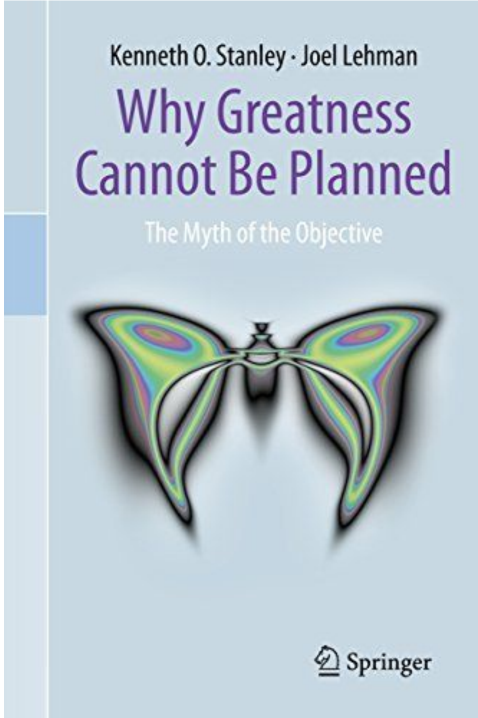
Maze setting　　Distance based　　Novelty based

Novelty Search: Lehman & Stanley

Slide reference: Prof.Jeff Clune's talk during ICML workshop

A book

Success often emerges from processes we can’t map out ahead of time



In open-ended problems:

- Objective functions may misdirect
- Breakthroughs emerge from searching for behavioral novelty
- Search for novelty leads to increasing complexity
- A reminder to allow room for exploration and discovery

AUTOMATED DESIGN OF AGENTIC SYSTEMS

Shengran Hu[1,2], **Cong Lu**[1,2], **Jeff Clune**[1,2,3]
[1]University of British Columbia, [2]Vector Institute, [3]Canada CIFAR AI Chair
{srhu,conglu}@cs.ubc.ca, jclune@gmail.com

Paper reference: https://arxiv.org/pdf/2408.08435

AutoML

- Hand designed solutions are eventually replaced by learned solutions as we put in more compute and data [The Bitter Lesson]
  - E.g. Computer vision: Handcrafted features -> CNNs
- AI-generating algorithms/AutoML
  - Meta-learning design components in AI systems
    - Neural Architecture Search / Hyper-parameter optimization

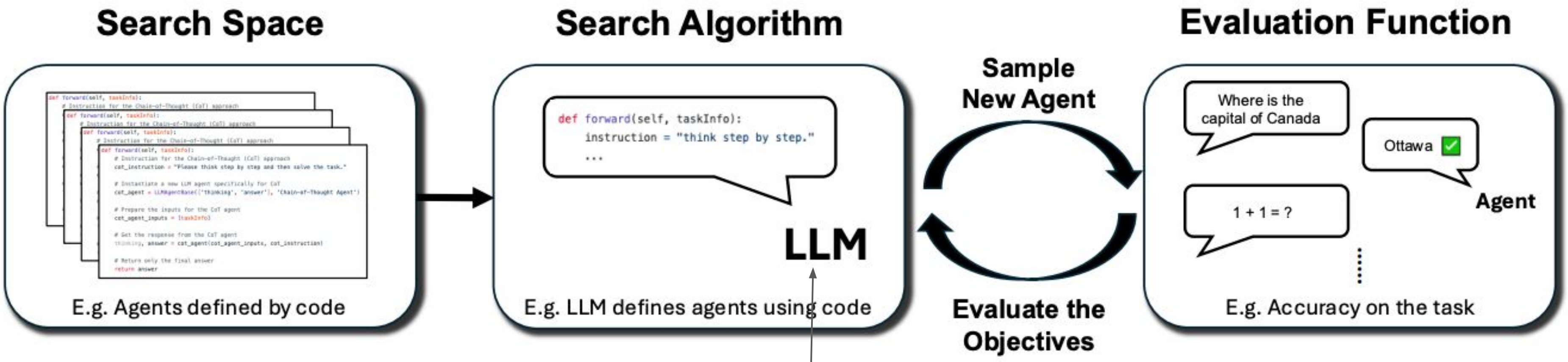
The bitter lesson: http://www.incompleteideas.net/IncIdeas/BitterLesson.html

Automated Design of Agentic System (ADAS)

- **Agentic system**: use Foundation Models (FMs) as modules in the workflow to solve tasks by planning, using tools, and carrying out multiple, iterative steps of processing
- **Automated Design of Agentic Systems (ADAS):** involves using a *search algorithm* to discover agentic systems across a *search space* that optimize an *evaluation function*.

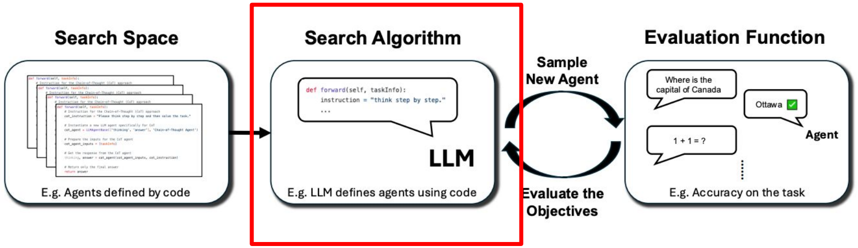
Automated Design of Agentic System (ADAS)

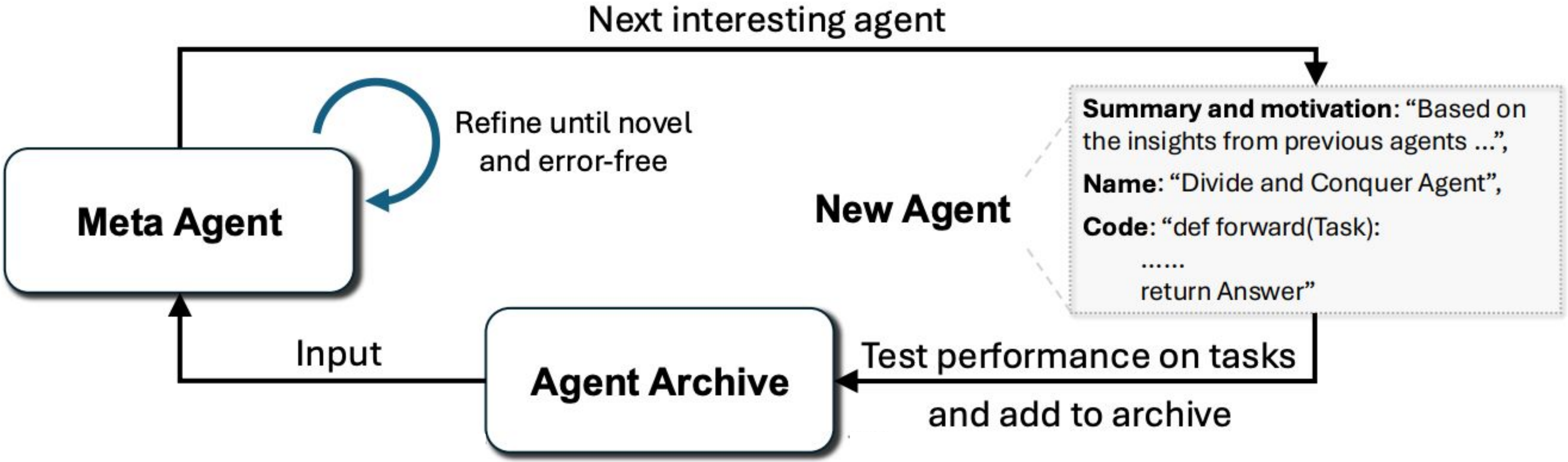
Three key components in ADAS



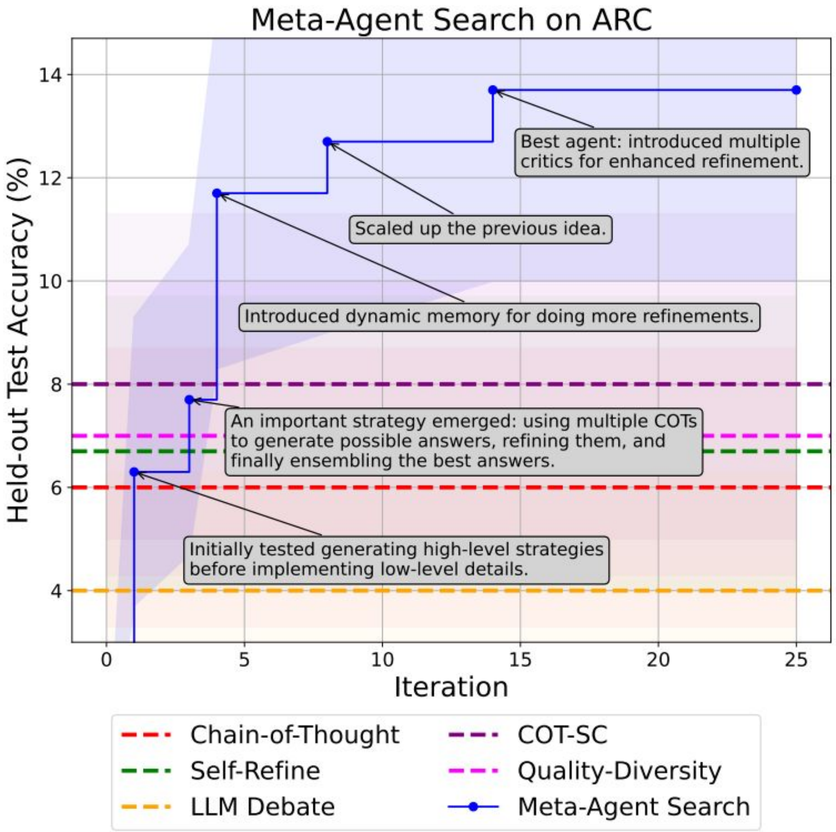
Define agent in code + use meta-agent to program new agents

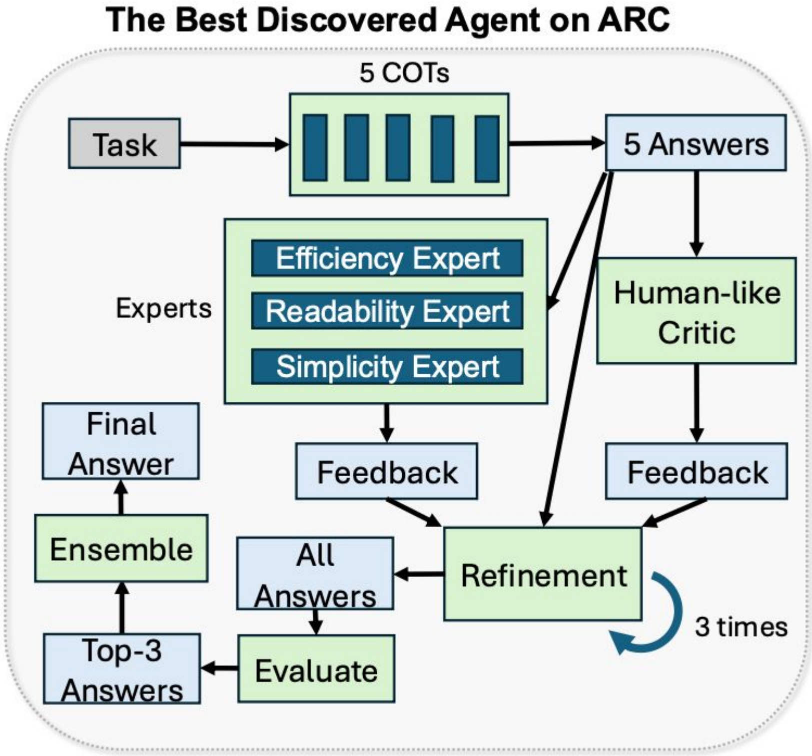
Meta agent search





Example on ARC challenge





Structured Feedback and Ensemble Agent

Summary

- Classical RL
  - Intuition behind RL: encourage good behavior, suppress bad behaviors
  - Key terminologies in RL: state, action, reward, value functions, different types of RL algorithms
- RL + LLM
  - RL shows its impact in LLM improvement: PPO, RLHF, DPO, GRPO…
- Evolutionary algorithm
  - Except from using RL as a search algorithm for optimal solution, we can also make use of evolutionary algorithm
- How to collect enough reward signals to scale up such feedback-dependent algorithms?