

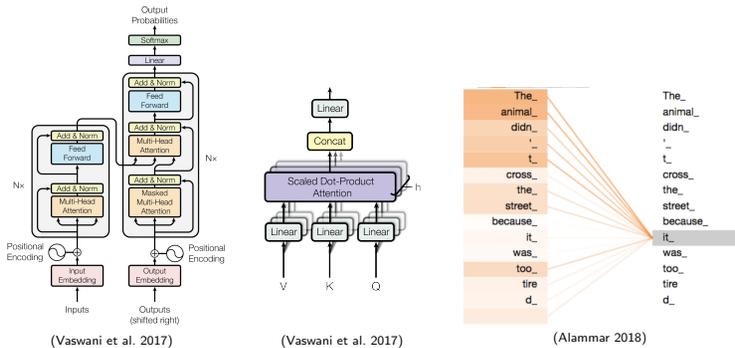
Introduction to Large Language Models (LLMs)

- ▶ Large Language Models (LLMs) such as GPT are key advancements in AI.
- ▶ LLMs generate human-like text and handle language-based tasks.
- ▶ Their foundation is the transformer architecture with self-attention.
- ▶ Self-attention helps manage long-range text dependencies efficiently.

Synergies with Generative AI

Yujin Tang and Risto Miikkulainen

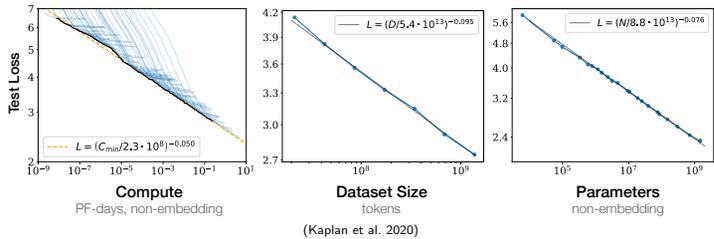
February 21, 2026



(a) Transformer model architecture. (b) Multi-head attention. (c) Scaled dot-product attention.

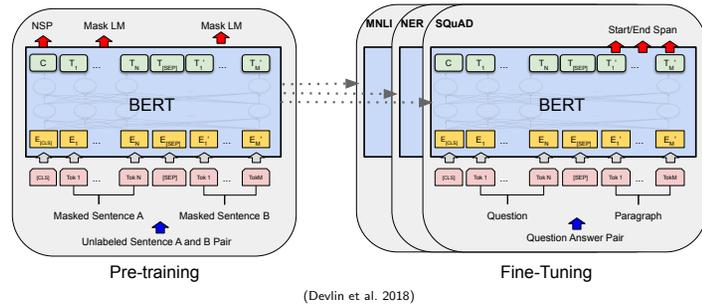
Scaling Laws and Applications of LLMs

- ▶ LLM performance improves logarithmically with size and data volume.
- ▶ Scaling laws predict the effectiveness of large models.
- ▶ Applications include chatbots, autonomous agents, and robotic interfaces.



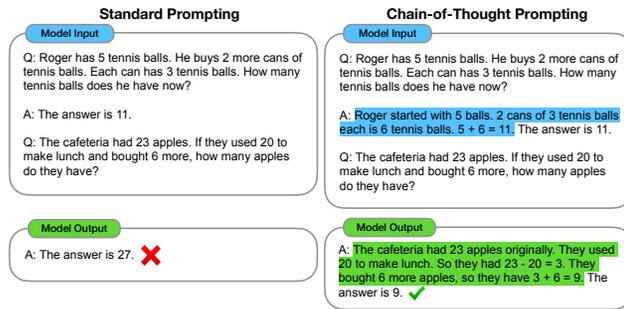
Training Large Language Models

- ▶ LLMs undergo pre-training on vast text corpora to predict the next token.
- ▶ Fine-tuning on task-specific data optimizes their performance.
- ▶ Models like BERT demonstrate the importance of meticulous tuning.



A New Field: Prompt Engineering

- ▶ LLMs need prompts to perform specific tasks efficiently.
- ▶ Discrete prompt-based adaptation eliminates direct parameter manipulation.
- ▶ Especially useful for black-box APIs like GPT-4 and Gemini.
- ▶ Effective prompt design, e.g. chain-of-thought, is crucial for LLM performance.

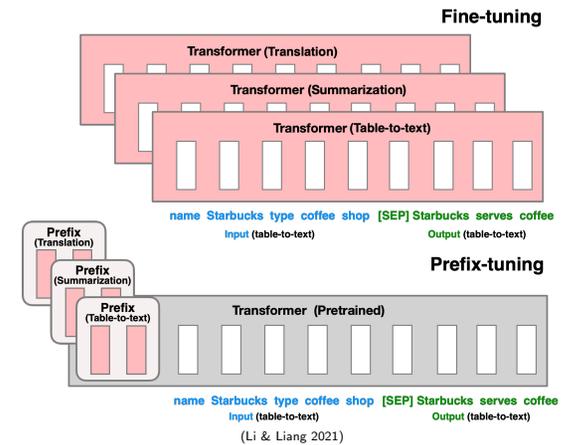


(Wei et al. 2023)



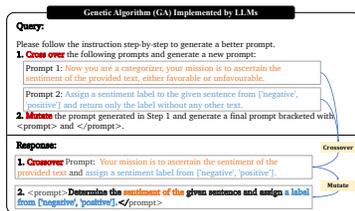
Challenges in Prompt Engineering

- ▶ Prefix (i.e. system prompt) tuning can be used to establish expertise.
- ▶ Prefix is fixed and often not readable.
- ▶ Visible online prompts can be optimized dynamically as well.
- ▶ Prompt engineering often requires extensive human effort and expertise.



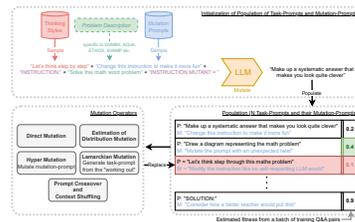
Evolutionary Algorithms for Prompt Optimization

- ▶ Prompts are treated like gene sequences in evolutionary algorithms.
- ▶ Evolution helps maintain prompt diversity, avoiding local optima.
- ▶ LLMs modify prompts iteratively, refining strategies.



(Guo et al. 2024)

(a) EvoPrompt



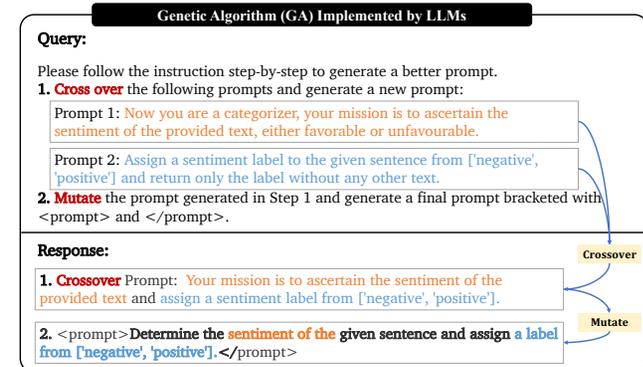
(Fernando et al. 2024)

(b) Promptbreeder



EvoPrompt Overview

- ▶ EvoPrompt uses Genetic Algorithm (GA) and Differential Evolution (DE) for prompt optimization.
- ▶ Suitable for black-box LLM APIs.
- ▶ GA and DE improve prompts through crossover and mutation.

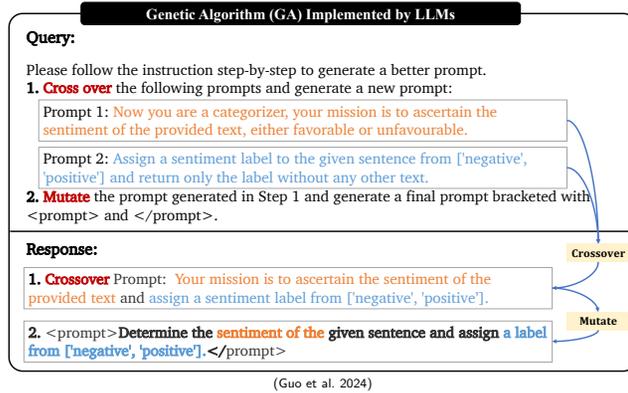


(Guo et al. 2024)



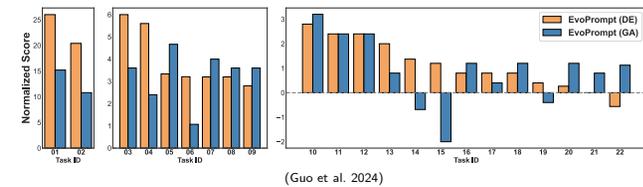
EvoPrompt: The Evolutionary Process

- ▶ Initial prompts are evaluated and combined to create new ones.
- ▶ Crossover and mutation operations enhance the prompts.
- ▶ The LLM itself generates new candidate prompts based on performance feedback.



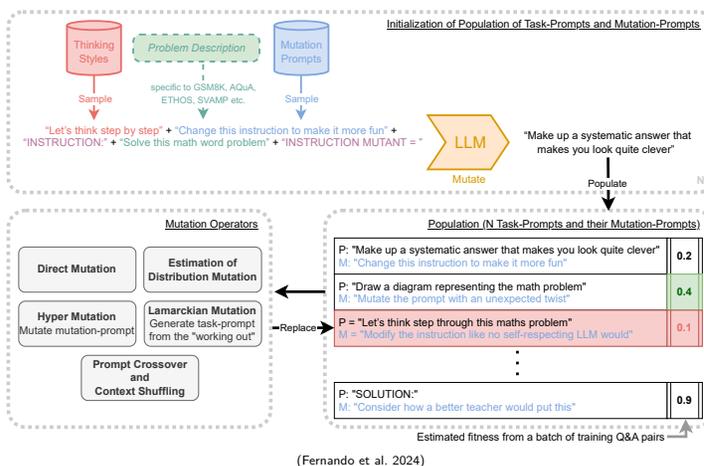
EvoPrompt Results on Benchmark Tasks

- ▶ EvoPrompt significantly improves performance on the challenging Big Bench Hard (BBH) tasks.
- ▶ DE variant led to a 25% improvement in some tasks.
- ▶ GA variant also showed strong results with improvements up to 15%.



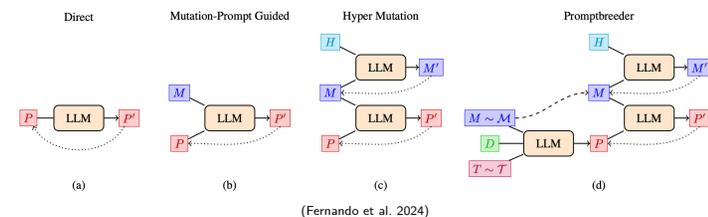
Promptbreeder: Optimizing Prompt Mutations as Well

- ▶ Promptbreeder automates prompt exploration and refinement.
- ▶ It uses evolutionary algorithms to optimize both task and mutation prompts.
- ▶ Self-referential system evolves mutation prompts as well.



Promptbreeder: Key Features

- ▶ Initial population of prompts comes from problem descriptions.
- ▶ LLMs are used to mutate task prompts with guidance from mutation prompts.
- ▶ Evolution refines both task prompts and mutation prompts for continual improvement.
- ▶ More complex mutations combine multiple prompts to explore new prompt spaces.



Performance of Promptbreeder

- ▶ Tested across arithmetic reasoning, commonsense reasoning, instruction induction, etc.
- ▶ Consistently outperformed Plan-and-Solve techniques.
- ▶ Robust in both zero-shot and few-shot scenarios.

	Method	LLM	MultiArith*	SingleEq*	AddSub*	SVAMP*	SQA	CSQA	AQuA-RAT	GSM8K
Zero-shot	CoT	text-davinci-003	(83.8)	(88.1)	(85.3)	(69.9)	(63.8)	(65.2)	(38.9)	(56.4)
	PoT	text-davinci-003	(92.2)	(91.7)	(85.1)	(70.8)	–	–	(43.9)	(57.0)
	PS	text-davinci-003	(87.2)	(89.2)	(88.1)	(72.0)	–	–	(42.5)	(58.2)
	PS+	text-davinci-003	(91.8)	(94.7)	(92.2)	(75.7)	(65.4)	(71.9)	(46.0)	(59.3)
	PS	PaLM 2-L	97.7	90.6	72.4	83.8	50.0	77.9	40.2	59.0
	PS+	PaLM 2-L	92.5	94.7	74.4	86.3	50.1	73.3	39.4	60.5
	APE	PaLM 2-L	95.8	82.2	72.2	73.0	38.4	67.3	45.7	77.9
	OPRO	PaLM 2-L	–	–	–	–	–	–	–	80.2
	PB (ours)	PaLM 2-L	99.7	96.4	87.8	90.2	71.8	85.4	62.2	83.9
Few-shot	Manual-CoT	text-davinci-003	(93.6)	(93.5)	(91.6)	(80.3)	(71.2)	(78.3)	(48.4)	(58.4)
	Auto-CoT	text-davinci-003	(95.5)	(92.1)	(90.8)	(78.1)	–	–	(41.7)	(57.1)
	PB	PaLM 2-L	100.0	98.9	87.1	93.7	80.2	85.9	64.6	83.5

(Fernando et al. 2024)

◀ ▶ ⏪ ⏩ 🔍 🔄

Promptbreeder Case Study: ETHOS Hate Speech Classification

- ▶ Promptbreeder evolved a strategy for hate speech classification.
- ▶ It improved accuracy from 80% to 89%.
- ▶ Example: Two sequentially applied, relatively long prompts.

Prompt 1: "Text contains hate speech if it includes any of the following:
 • Words or phrases that are derogatory, disrespectful, or dehumanizing toward a particular group of people.
 • Statements that generalize about a particular group of people in a negative way.
 • Statements that incite hatred or violence against a particular group of people.
 • Statements that express a desire for the extermination or removal of a particular group of people.
 7 : Text contains hate speech if it contains language that is hostile or discriminatory towards a particular group of people. This could include language that is racist, sexist, homophobic, or other forms of hate speech. Hate speech is harmful because it can create a hostile environment for marginalized groups and can lead to discrimination and violence."
Prompt 2: "You are given a piece of text from the internet. You are supposed to label the text as Hate Speech or Not based on the given criteria. Hate Speech is defined as speech that is intended to incite hatred or violence against a particular group of people based on their race, religion, sex, sexual orientation, or national origin. The given criteria are: 1. Identify the target group of the speech. This can be a specific group of people, such as a race, religion, or gender, or it can be a more general group, such as people with disabilities or sexual minorities. 2. Identify the harmful speech. This can be speech that is threatening, abusive, or derogatory. 3. Evaluate the context of the speech. This can include the speaker's intent, the audience, and the time and place of the speech. The advice was: Remember to always evaluate the context of the speech when making a determination as to whether it is hate speech or not. Speech that is intended to be humorous or satirical may not be considered hate speech, even if it contains harmful language."

(Fernando et al. 2024)

◀ ▶ ⏪ ⏩ 🔍 🔄

EvoPrompt vs. Promptbreeder

- ▶ EvoPrompt focuses on refining prompts with crossover and mutation.
- ▶ Promptbreeder takes a more complex approach with self-referential evolution.
- ▶ Both demonstrate the effectiveness of evolutionary algorithms in prompt optimization.

Aspect	EvoPrompt	PromptBreeder
Core Approach	Traditional genetic algorithm with simple text-based mutations and population-based evolution	Meta-learning system that evolves both prompts and mutation operators, using LLM for generating mutations
LLM Integration	Used primarily for fitness evaluation	Actively involved in evolution process, generates mutation operators
Complexity	Simpler implementation, straightforward evolution process	More complex architecture with meta-learning and self-improvement capabilities
Flexibility	Task-focused with predefined mutation operators	Can discover novel optimization strategies through meta-learning

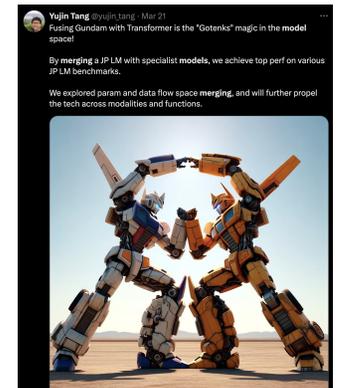
◀ ▶ ⏪ ⏩ 🔍 🔄

Introduction to Model Merging

- ▶ AI development might follow a path similar to human collective intelligence.
- ▶ AI systems will consist of many small, specialized models instead of a single, all-knowing system.
- ▶ Open-source AI models demonstrate a promising trend of merging specialized models.



(Toei Animation 2025)

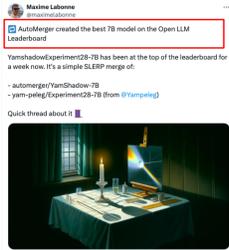


(Tang 2025)

◀ ▶ ⏪ ⏩ 🔍 🔄

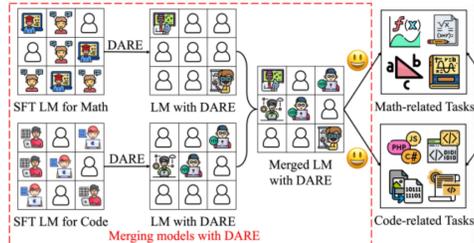
Model Merging and Democratization of AI

- ▶ Open models are extended and fine-tuned for different niches.
- ▶ Model merging allows combining strengths from different domains.
- ▶ Top-performing models are often merged versions of base models.



(Labonne 2025)

(a) Performance boost.

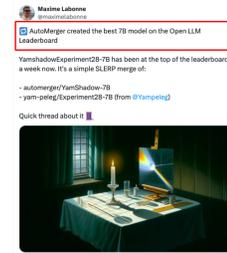


(Yu et al. 2023)

(b) Skill integration.

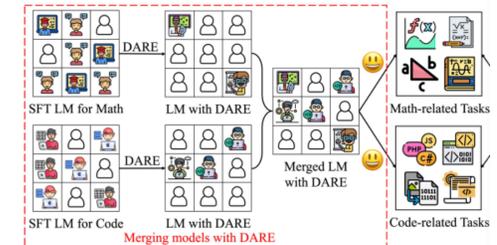
Challenges in Model Merging

- ▶ Merging models manually is often difficult and relies on intuition.
- ▶ Evolutionary algorithms provide systematic ways to discover optimal combinations.
- ▶ These algorithms explore novel combinations that human intuition might miss.



(Labonne 2025)

(a) Performance boost.

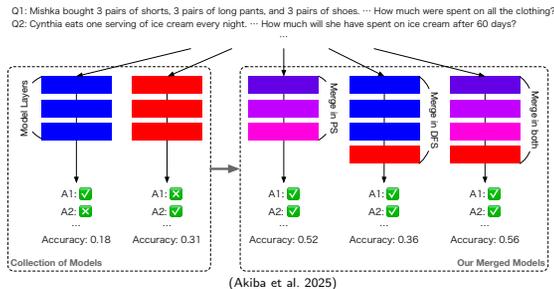


(Yu et al. 2023)

(b) Skill integration.

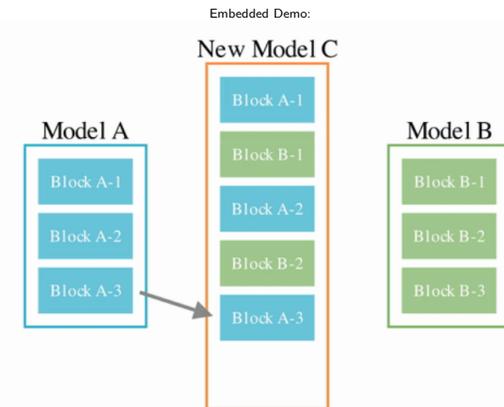
Evolutionary Model Merge Method

- ▶ Combines two approaches: Data Flow Space (Layer Merging) and Parameter Space (Weight Merging).
- ▶ Uses evolution to optimize how layers and weights from multiple models are merged.
- ▶ Capable of discovering architectural innovations beyond human intuition.



Merging in the Data Flow Space

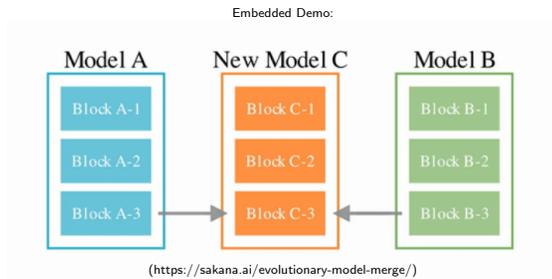
- ▶ Evolution explores combinations of layers from different models.
- ▶ Combinatorial space is too large for manual exploration.
- ▶ Evolutionary algorithms find the best combinations of layers for model merging.



(<https://sakana.ai/evolutionary-model-merge/>)

Merging in the Parameter Space

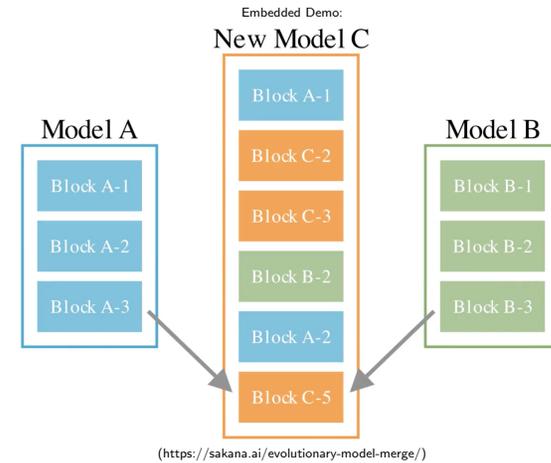
- ▶ Evolutionary algorithms also optimize weight merging between models.
- ▶ Different layers can use different mixing ratios for optimal performance.
- ▶ Evolution explores infinite ways of mixing weights for merged models.



Navigation icons: back, forward, search, etc.

Merging in Both Spaces

- ▶ Evolution can also combine both methods.
- ▶ Best of both worlds.



Navigation icons: back, forward, search, etc.

Case Study: Japanese Math LLM

- ▶ Combined three models: Shisa-Gamma (Japanese), WizardMath (Math), and Abel (Math).
- ▶ Evolution created a Japanese Math LLM over 100-150 generations.
- ▶ This model outperformed the source models significantly.

Id.	Model	Type	Size	MGSM-JA (acc ↑)
1	Shisa Gamma 7B v1	JA general	7B	9.6
2	WizardMath 7B v1.1	EN math	7B	18.4
3	Abel 7B 002	EN math	7B	30.0
4	Akiba et al. (2024) (PS)	1 + 2 + 3	7B	52.0
5	Akiba et al. (2024) (DFS)	3 + 1	10B	36.4
6	Akiba et al. (2024) (PS+DFS)	4 + 1	10B	55.2
7	Llama 2 70B	EN general	70B	18.0
8	Japanese StableLM 70B	JA general	70B	17.2
9	Swallow 70B	JA general	70B	13.6
10	GPT-3.5	commercial	-	50.4
11	GPT-4	commercial	-	78.8

(Akiba et al. 2025)

Navigation icons: back, forward, search, etc.

Case Study: Japanese Vision-Language Model

- ▶ Combined a Vision-Language Model (LLaVa-1.6-Mistral-7B) with a Japanese LLM (Shisa Gamma 7B v1).
- ▶ Evolution produced a capable Japanese VLM that outperformed existing models.
- ▶ Also learned aspects of Japanese culture, e.g. blue traffic light.



(<https://sakana.ai/evolutionary-model-merge/>)

Navigation icons: back, forward, search, etc.

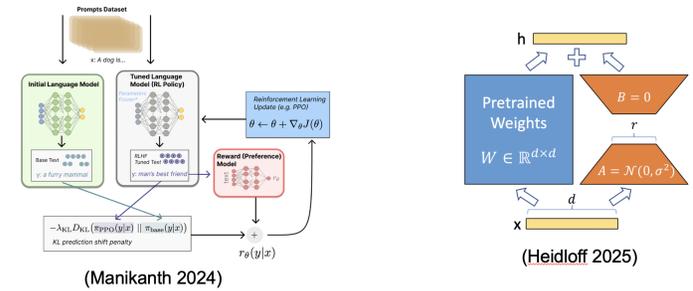
Case Study: Merging Diffusion Models

- ▶ Evolutionary merging applies to diffusion models (DM) as well.
- ▶ E.g. forming a 4-step DM that supports Japanese native prompts.
- ▶ Demonstrates the power of evolutionary algorithms in combining diverse models.



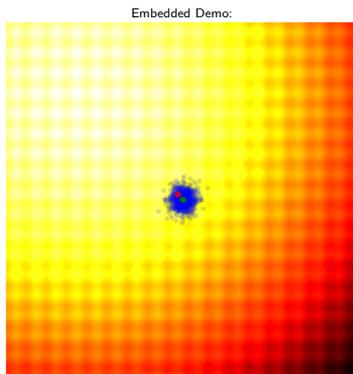
Demo link: <https://sakana.ai/evolutionary-model-merge/>

Current Fine-tuning Methods



- Reinforcement learning for human feedback (RLHF)
- Standard RL methods like PPO, GRPO
- Action-space exploration: gradients to favor actions with higher rewards
- Or, freeze the pretrained model and train adapters (e.g. LoRA)

Fine-Tuning with Evolutionary Strategy (ES)



(<https://blog.otoro.net/2017/10/29/visual-evolution-strategies/>)

- ▶ Early population-based search (Rechenberg, Schwefel 1970s)
 - ▶ Practical nonlinear optimization (CMA-ES; Hansen et al. 1990s)
 - ▶ Parallel RL (by OpenAI; Salimans et al. 2017)
- ▶ Does it scale to LLMs?

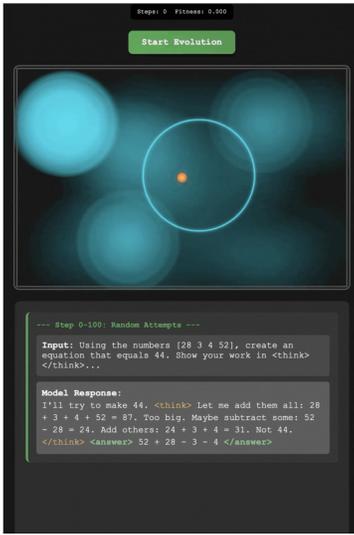
ES Fine-tuning with Billions of Parameters



(Gan 2025)

- Fine-tuning in parameter space
- Instead of action space like PPO, GRPO
- Finds principled and systematic solutions that underlie behavior
- Demonstrated in the Countdown task:
 - Find an expression using +*/ for 28 3 4 52 that = 44
 - Solution: $28 + 52/4 + 3 = 44$
- A simple reasoning task but difficult for LLMs

ES Fine-tuning with Billions of Parameters



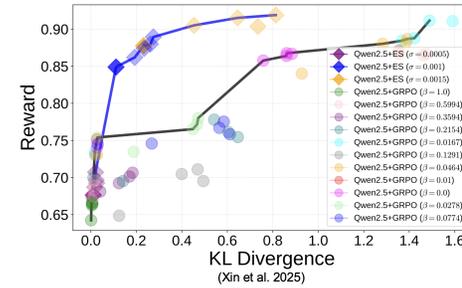
(Gan 2025)

ES fine-tuning improved accuracy 36%

- Vs. PPO 18%, GRPO 21%
- Can improve small and large models (Qwen, Llama 0.5-8B)
- Needs only 20% of the training examples

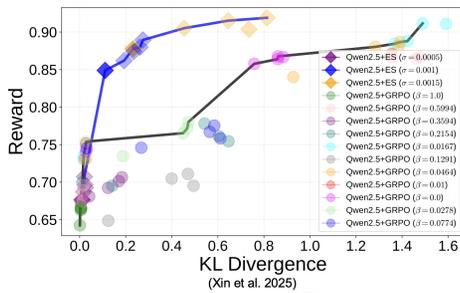
Base Model	Raw	RL			ES
		PPO	GRPO	GRPO30	
Qwen-2.5-0.5B-Instruct	0.1	0.3	0.3	0.5	14.4
Qwen-2.5-1.5B-Instruct	0.7	14.2	13.9	14.8	37.3
Qwen-2.5-3B-Instruct	10.0	20.1	30.9	32.5	60.5
Qwen-2.5-7B-Instruct	31.2	55.1	54.2	52.8	66.8
Llama-3.2-1B-Instruct	0.4	11.2	14.5	13.0	16.8
Llama-3.2-3B-Instruct	3.2	35.3	39.4	38.8	51.6
Llama-3.1-8B-Instruct	8.1	42.8	49.9	51.3	61.2

Principles of ES Fine-tuning



- Conciseness task: Answer questions with short answers
 - E.g. "Name a primary color" -> "Red".
- RL requires many examples, makes large changes
 - Reward hacking: breaks its performance

Principles of ES Fine-tuning

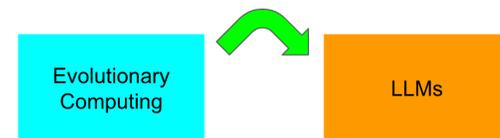


- ES fine tunes successfully with just two examples!
 - Implements principles underlying conciseness in parameter space
 - High rewards with minimal changes to the model
- Open-source:
 - <https://github.com/VsonicV/es-fine-tuning-paper>

Conclusion #1: Evolutionary Computing Enhances LLMs

- ▶ Evolutionary computing plays a significant role in optimizing LLM performance.
- ▶ Techniques like EvoPrompt and Promptbreeder show the power of evolutionary algorithms.
- ▶ Evolutionary model merging of multiple LLMs leads to more capable LLM systems.

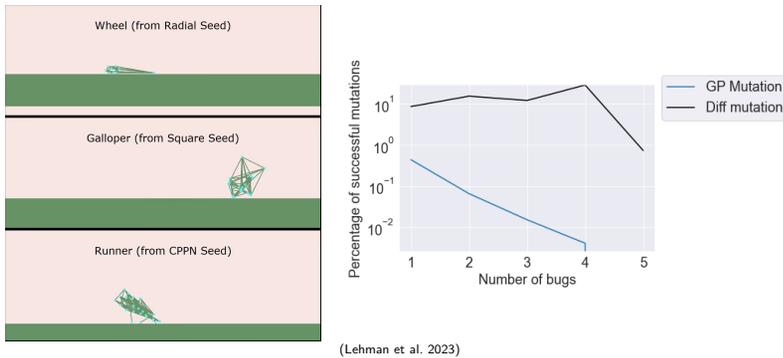
Enhance LLM performance via evolutionary prompt tuning, model merging, etc



(Tang 2025)

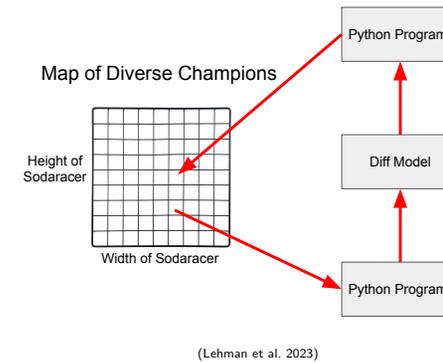
Going the Other Way: Evolution through LLMs

- ▶ E.g. ELM: Evolution through Large Models.
- ▶ Generating code for the Sodaracer domain:
 - ▶ LLM trained on large datasets of sequential code changes.
 - ▶ Then used to generate genetic operations.
- ▶ Three principles:
 - ▶ 1. LLM generates “diff” mutations, simulating meaningful code changes.
 - ▶ These mutations outperform traditional random mutations used in GP.



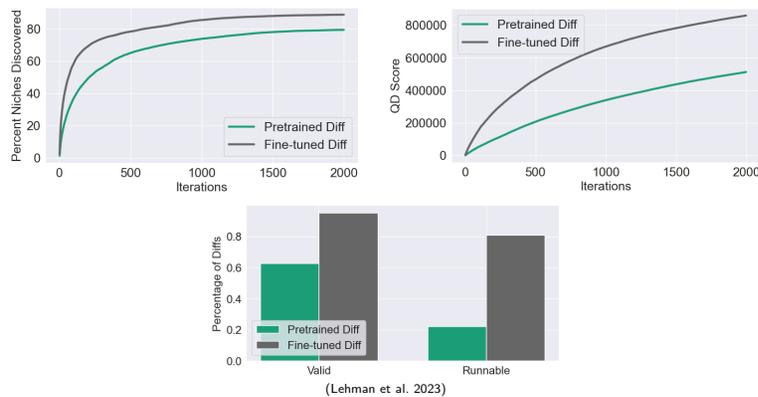
MAP-Elites with Diff Mutation

- ▶ 2. MAP-Elites categorizes solutions into different niches based on features.
 - ▶ LLMs generate targeted modifications to solutions using diff mutation.
 - ▶ Systematic exploration of diverse solution landscapes.



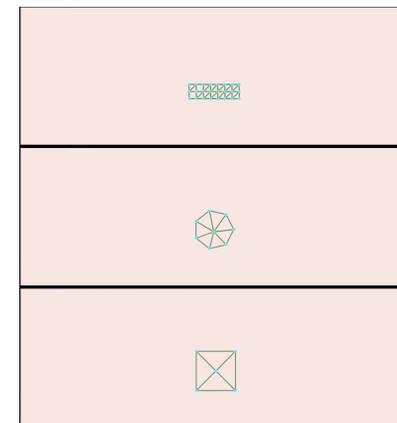
Fine-Tuning Diff Models

- ▶ 3. Fine-tuning LLMs on task-specific datasets, i.e. Sodaracer.
 - ▶ Refined diff models generate more contextually appropriate code changes.
 - ▶ Niches discovered, QD score, and code quality all improve with fine-tuning.



Seed Solutions in Evolutionary Search

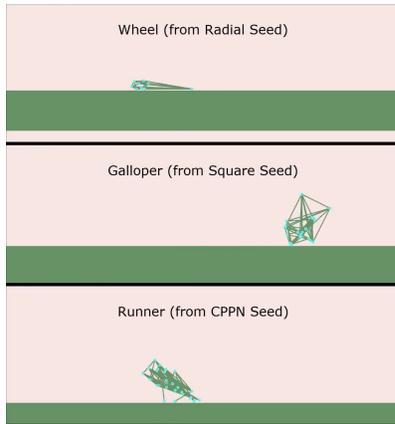
- ▶ Evolutionary search begins with seed solutions.
- ▶ LLMs enhance seed solutions by generating new body designs in evolutionary robotics.
- ▶ Seed solutions allow evolutionary exploration of complex designs.



Demo link: <https://youtu.be/jeP8Nsulu48>

Novel Solutions in Evolutionary Search

- ▶ From the seeds, distinctly different solutions evolve.



Demo Links:

Seeded: <https://youtu.be/8C2K5fk28HI>

Varieties: <https://youtu.be/QNyNtvwA9FI>

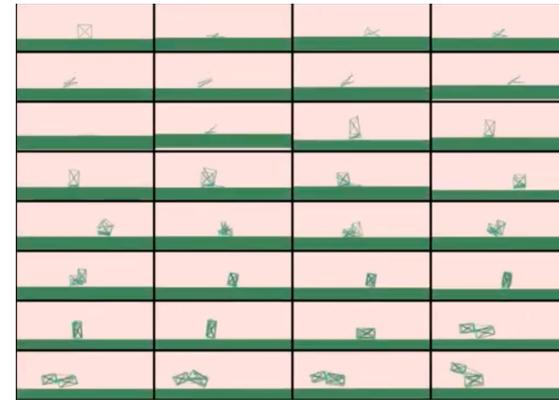
Blob: <https://youtu.be/JDUA18yrNcY>

Hopper: <https://youtu.be/noSPGF5m3M>

Centipede: <https://youtu.be/zhMsPzo22do>

Exploration Process

- ▶ The lineage of one of these solutions demonstrates exploration and refinement.

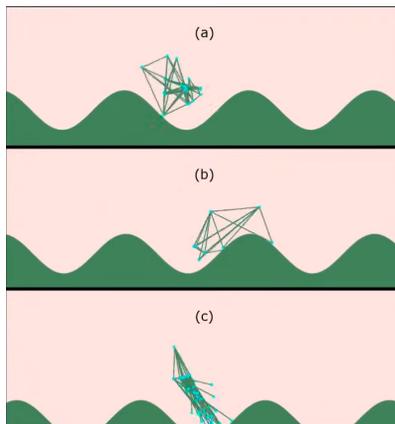


Demo link:

From square: <https://youtu.be/M9pAJuX6dyM>

Adjusting to Changes

- ▶ Evolution can adjust to changes in the environment.



Demo Links:

Bump Challenge: <https://youtu.be/Mo-rXnFq6vQ>

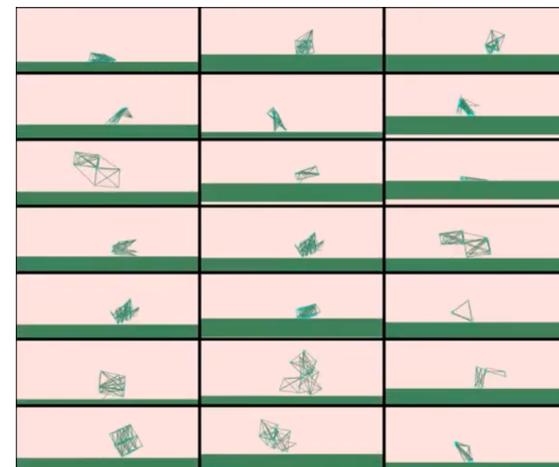
Solution 1: <https://youtu.be/15PVSLDknWM>

Solution 2: <https://youtu.be/WEM1dBtLLTw>

Tunnel Challenge: https://youtu.be/P9A1ruI3_tU

Conclusion: LLMs Enhance Evolutionary Computing

- ▶ Evolution discovers novel and complex body designs.
- ▶ LLM-based mutations can explore and refine.
- ▶ The combination can drive innovation in complex task-solving.



(<https://sakana.ai/evolutionary-model-merge/>)

LLMs as Evolution Strategies

- ▶ Can LLMs function as optimization tools for evolutionary strategies (ES)?
- ▶ E.g. LLM (Claude) optimizing the function $f(x) = 3 \times (x - 2024)^2 + 10$.
- ▶ Creating new points based on evaluations.
- ▶ With population size eight, getting close in five iterations:

I have a function $f(x)$ where x is a real number, and I need you to act like an evolutionary algorithm.

We are going to interact with each other for 5 rounds. In each round, you give me 8 numbers as x , and I'll give you the corresponding values of $f(x)$. At the end of the interaction, I'll ask you to give me a single x that is most likely to generate the lowest value for $f(x)$.

Your output should be in the format of "Iter=[iter], x=[x1], x2=[x2], ..., x8=[x8]", and I'll say "next" for the next iteration, at the end, I'll say "end" and you need to give me "x=[x]".

I understand you want me to act as an evolutionary algorithm to minimize an unknown function $f(x)$ through 5 rounds of interaction, with 8 samples per round. I'll use strategies like mutation and crossover, adapting my search based on the fitness values you provide. Let me start with an initial population spread across a reasonable range to explore the search space:

Iter=1, x1=-30.0, x2=-5.0, x3=-1.0, x4=0.0, x5=1.0, x6=5.0, x7=10.0, x8=15.0

12414178, 12350533, 12301885, 12289738, 12277897, 12229093, 12168598, 12108253

Based on these values, I can see there's a general downward trend as x increases, suggesting the minimum might be found in higher x values. I'll focus more sampling in that direction while maintaining some exploration:

Iter=2, x1=10.0, x2=12.5, x3=15.0, x4=17.5, x5=20.0, x6=22.5, x7=25.0, x8=30.0

12168598, 12138406.75, 12108253, 12078136.75, 12048058, 12018016.75, 11988013, 11928118

The trend continues downward with higher x values, and appears to be fairly smooth. I'll continue exploring higher values while keeping some points near our current best performers:

Iter=3, x1=25.0, x2=30.0, x3=35.0, x4=40.0, x5=45.0, x6=50.0, x7=55.0, x8=60.0

11988013, 11928118, 11868373, 11808778, 11749333, 11690038, 11630893, 11571898

The values continue to decrease steadily with higher x values. I'll keep exploring higher values while maintaining some exploration near our best performers.

Iter=4, x1=55.0, x2=60.0, x3=65.0, x4=70.0, x5=75.0, x6=80.0, x7=85.0, x8=90.0

11530893, 11571898, 11513053, 11454208, 11395363, 11336518, 11277673, 11218828

The pattern continues to show improvement with higher x values, with no sign of reaching a minimum yet. Let me explore even higher values:

Iter=5, x1=85.0, x2=90.0, x3=95.0, x4=100.0, x5=105.0, x6=110.0, x7=115.0, x8=120.0

11277673, 11218828, 11160333, 11101838, 11043343, 10984848, 10926353, 10867858

Based on all observed data points across the 5 iterations, the function appears to be strictly decreasing as x increases, with no signs of reaching a minimum yet. The lowest value we observed was at $x=120.0$, which gives us 108568. Since the trend suggests the minimum lies even further, I would predict:

$x=150.0$

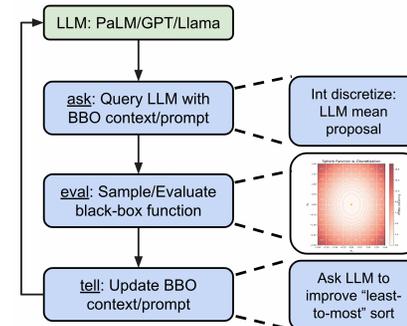
This prediction extrapolates beyond our sampled range since the function shows no sign of reaching a minimum within our sampled domain, and continues to decrease at a fast constant rate.

(Tang 2025)

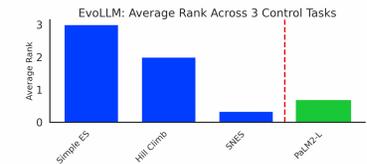
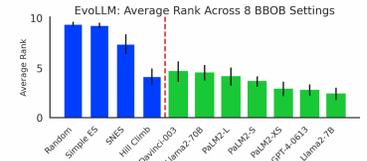


EvoLLM: LLMs in Evolutionary Cycles

- ▶ The EvoLLM strategy utilizes LLMs as a mutation operator:
 - ▶ LLMs suggest optimal sampling points for evaluation.
 - ▶ Points are evaluated; results consolidated; then iterated.
- ▶ Improves over standard ES.



(Lange et al. 2024)



EvoLLM Design: High-Level Prompt Design

- ▶ EvoLLM represents solution candidates as integers.
- ▶ The LLM prompt includes top M solutions from K generations.
- ▶ An improvement query is added to the prompt.

$$f_{K,1}^* : x_{1,K}^* ; x_{K,1}, x_{K,2}, \dots, x_{K,M}$$

...

$$f_{k,1}^* : x_{1,k}^* ; x_{k,1}, x_{k,2}, \dots, x_{k,M}$$

...

$$f_{1,1}^* : x_{1,1}^* ; x_{1,1}, x_{1,2}, \dots, x_{1,M}$$

$$f_{LLM}^{query} : < \text{LLM-Output} >;$$

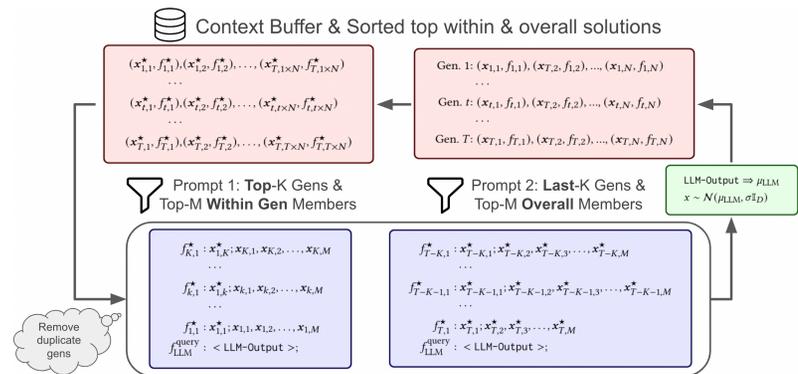
(Lange et al. 2024)

x_k^*, f_k^* denotes the best-performing solution and its fitness up to generation k .



EvoLLM Process

- ▶ Population forms a context buffer for LLM:
- ▶ Initialize through random search.
- ▶ Form two prompts: Top K and last K generations.
- ▶ Query LLM to get new solutions; add to context buffer.

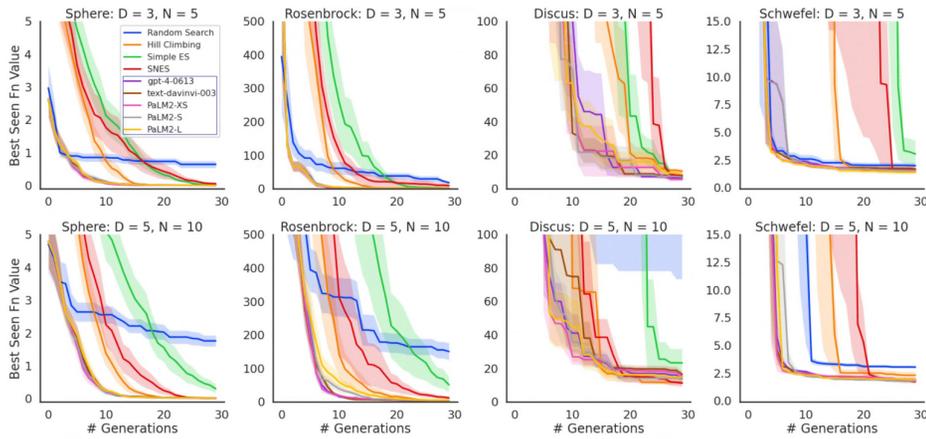


(Lange et al. 2024)



Evaluating EvoLLM Performance: BBOB Tasks

- ▶ EvoLLM outperforms Random Search, Hill Climbing, ES in BBOB tasks.
- ▶ More efficient in generating solutions, taking less than 10 generations.
- ▶ EvoLLM works with different LLMs such as PaLM2, GPT-4, and Llama2.

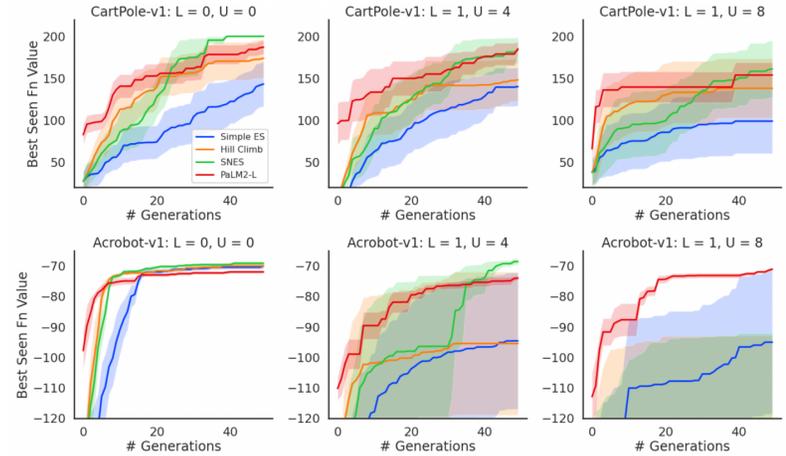


(Lange et al. 2024)



Evaluating EvoLLM Performance: Control Tasks

- ▶ Control tasks from OpenAI Gym (e.g., CartPole-v1 and Acrobot-v1).
- ▶ Tasks involve evolving 16 to 40 parameters of feedforward neural networks.
- ▶ EvoLLM achieves higher performance than baseline algorithms.

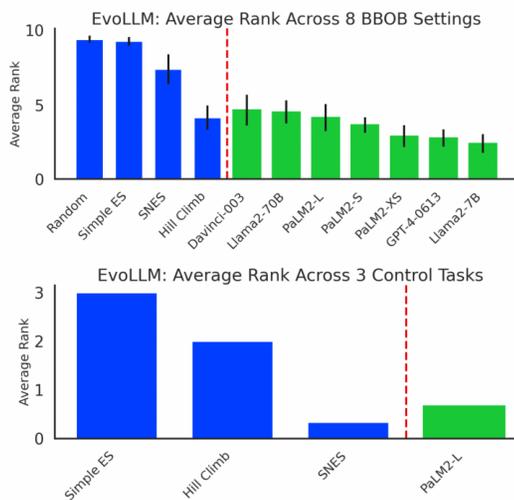


(Lange et al. 2024)



LLM Model Size and Compute

- ▶ Larger LLMs tend to perform worse than smaller models in EvoLLM.
- ▶ EvoLLM outperforms baselines even with smaller compute budgets.

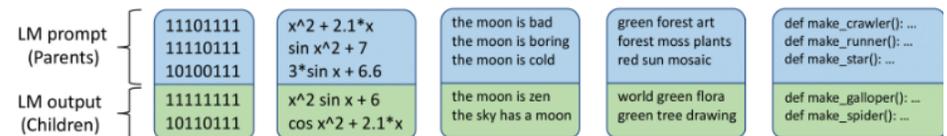


(Lange et al. 2024)



Generalizing to Language Model Crossover (LMX)

- ▶ LLMs can be used to implement crossover operations as well.
- ▶ They generate remarkably meaningful crossovers in various domains.
- ▶ Even when it is difficult to form formal crossover operators.

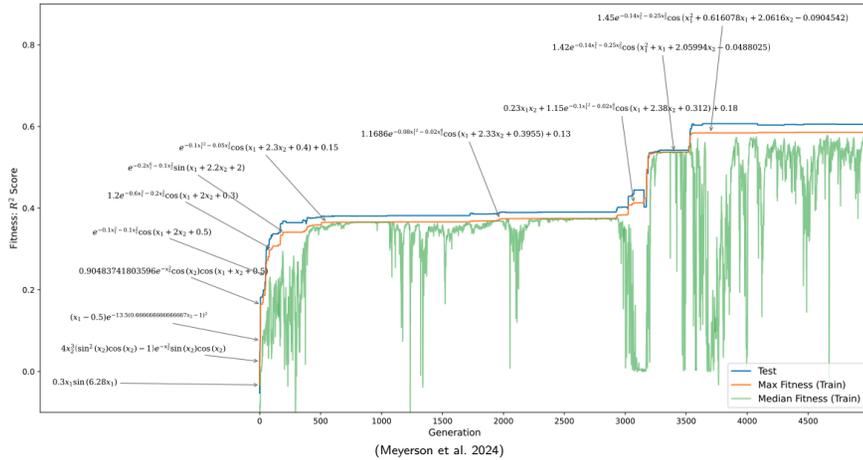


(Meyerson et al. 2024)



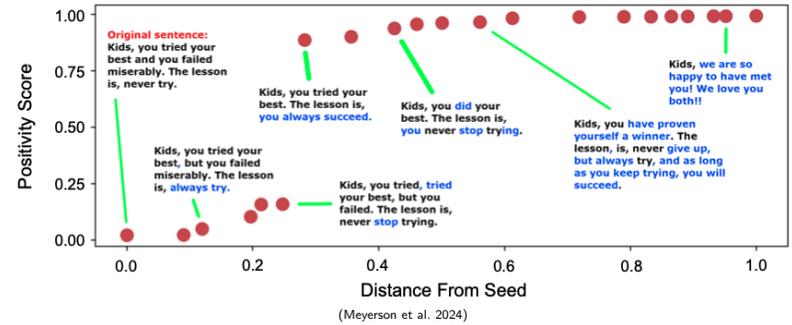
LMX in a Mathematical Domain

- ▶ Symbolic regression to a dataset (SRBench Banana problem; 5300 samples of 2 features).
- ▶ Settles on a skeleton quickly.
- ▶ Fine tunes the constants with surprising specificity.



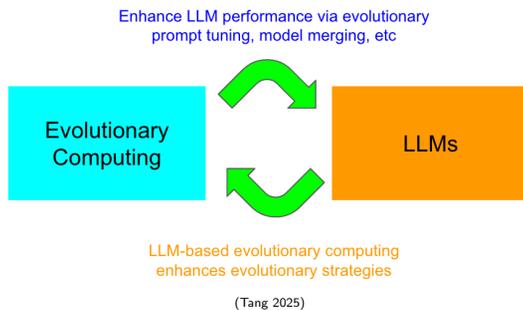
LMX in a Language Domain

- ▶ Make the sentiment of the statement more positive.
- ▶ Creating a variety of changes from meaningful to extreme.



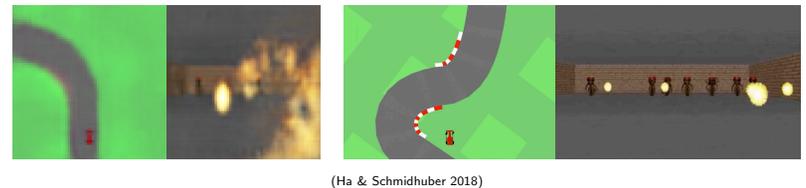
Conclusion #2: LLMs Enhance Evolutionary Computation

- ▶ EvoLLM demonstrates the potential of LLMs in enhancing evolutionary strategies.
- ▶ LMX utilizes LLM crossover in a wide variety of domains.
- ▶ More efficient and effective optimization, especially in black-box optimization.
- ▶ Expands the scope of evolutionary optimization.



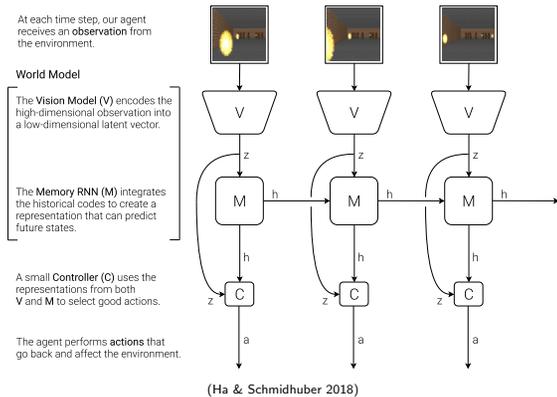
Introduction to World Models

- ▶ Deep generative models can learn to simulate complex environments.
- ▶ Such “world models” can serve as training environments for agents.
- ▶ Evolutionary algorithms can then optimize controllers.



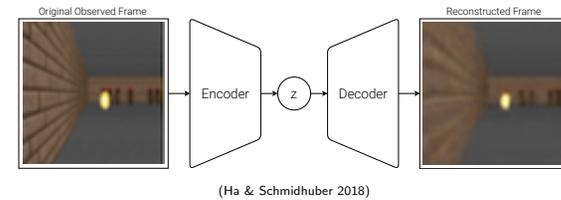
Simple World Model for Agents

- ▶ The agent has three core components:
 - ▶ **Vision (V)** compresses sensory input.
 - ▶ **Memory (M)** predicts future states.
 - ▶ **Controller (C)** decides actions based on V and M.



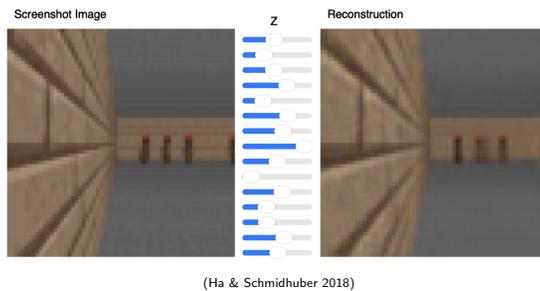
Vision Model (V)

- ▶ The Vision Model learns an abstract, compressed representation of each observed frame.
- ▶ Variational Autoencoder (VAE) is used for compressing image data.
- ▶ Compresses high-dimensional input into a low-dimensional latent vector.



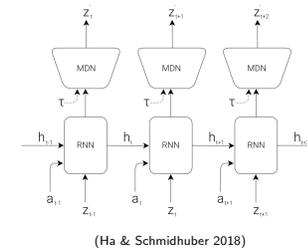
VAE Example on VizDoom Environment

- ▶ Trained VAE compresses screenshots of the VizDoom environment into a 16-dimensional latent vector.
- ▶ The vector can be used to reconstruct the original image.



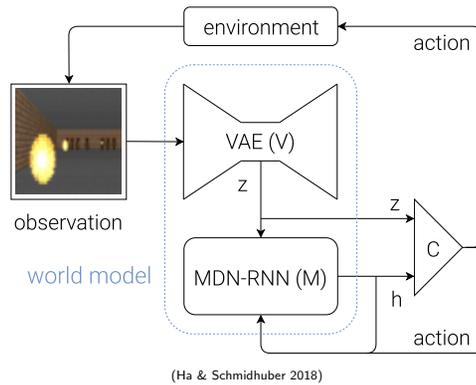
Memory Model (M)

- ▶ Predicts future latent vectors.
- ▶ Uses Recurrent Neural Networks (RNN) to predict future states based on past information.
- ▶ Mixture Density Network (MDN) outputs parameters of a Gaussian used to sample the predictions.
- ▶ Helps represent temporal aspects of the environment.



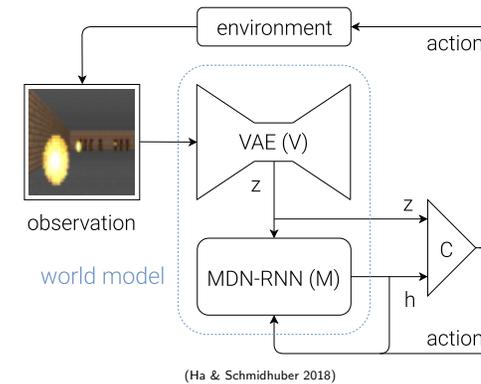
Controller Model (C)

- ▶ Decides actions that maximize rewards.
- ▶ The simplest controller is a linear model.
- ▶ More complex controllers can include hidden layers.
- ▶ Can be evolved e.g. with NEAT.



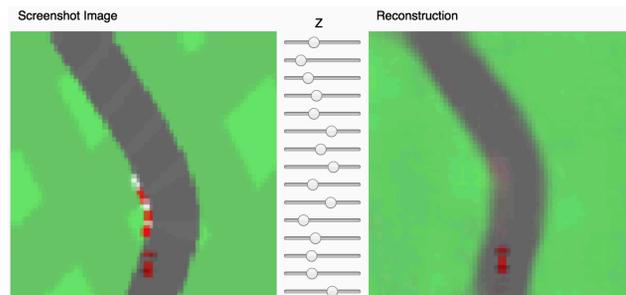
Putting Everything Together

- ▶ V (Vision), M (Memory), and C (Controller) interact to guide the agent's behavior.
- ▶ The vision model processes inputs, the memory model predicts future states, and the controller outputs actions.



Feature Extraction with World Models: CarRacing Task

- ▶ Use world models to extract features from environments.
- ▶ In the CarRacing task, a variational encoder compresses high-dimensional input into a 16-dimensional latent vector.
- ▶ A simple controller is trained using evolutionary algorithms.



(Ha & Schmidhuber 2018)



Performance of Simple Controllers

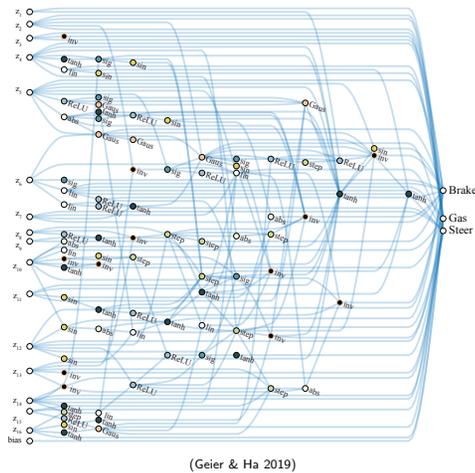
- ▶ Vision-only (z -only) models achieved moderate success with a linear and nonlinear controller
- ▶ NEAT controller improved performance significantly.
- ▶ Full world model ($z + h$) solved the task even with a linear controller.

Controller	Average Score
1-Layer, z -only	632 ± 251
2-Layer, z -only	788 ± 141
NEAT, z -only	893 ± 74
1-Layer $z + h$, linear	906 ± 21



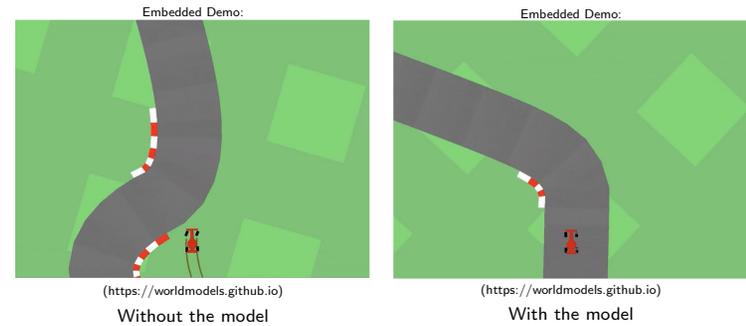
Evolved NEAT Controller

- ▶ Weight agnostic approach, only the structure evolved.
- ▶ Could take advantage of different activation functions.



Impact of World Model

- ▶ Without the model wobbly and error-prone.
- ▶ With the model more stable and efficient.



VizDoom Environment: Take Cover Task

- ▶ VizDoom environment is used to train an agent to avoid fireballs.
- ▶ The task is to maximize survival time.



(Ha & Schmidhuber 2018)



Using CMA-ES to Train Controllers in VizDoom

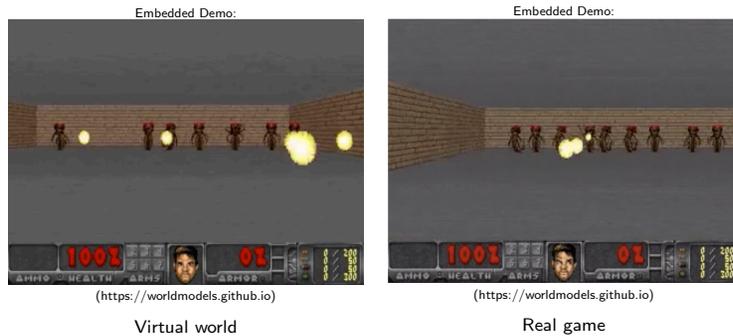
- ▶ Evolutionary algorithms (CMA-ES) are used to train the controller in a virtual environment.
- ▶ Performance is evaluated in both the virtual and real environments.
- ▶ Temperature τ controls exploration.
- ▶ With high τ , transfers very well.

Temp (τ)	Virtual Score	Actual Score
0.10	2086 \pm 140	193 \pm 58
0.50	2060 \pm 277	196 \pm 50
1.00	1145 \pm 690	868 \pm 511
1.15	918 \pm 546	1092 \pm 556
1.30	732 \pm 269	753 \pm 139



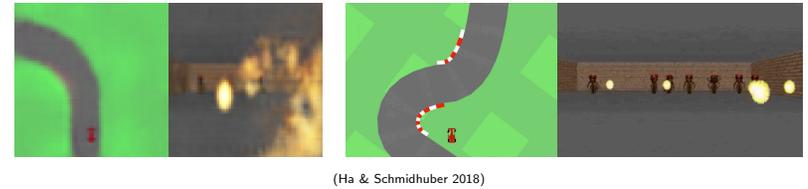
Transfer from the World Model to Real World

- ▶ Performance is qualitatively similar as well.
- ▶ World models can thus be used to create agents for the real world.



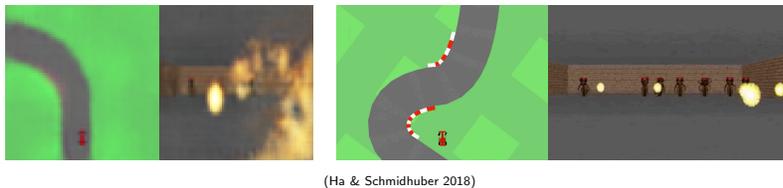
Future Opportunities

- ▶ Iterative data collection to refine world models.
- ▶ Incorporating novelty search to encourage exploration.
- ▶ Multi-agent systems and adversarial training in simulated environments.



World Model Conclusion

- ▶ World models allow agents to simulate, plan, and learn effectively.
- ▶ Combining deep generative models with evolution leads to significant advancements.
- ▶ Can transfer well into the real world.



Conclusion: Synergies of Neuroevolution with Generative AI

- ▶ Neuroevolution and generative AI form a powerful synergy, combining the creative capacity of generative models with the optimization power of evolutionary algorithms.
- ▶ **Key successes:**
 - ▶ Efficient model adaptation through evolutionary prompt engineering.
 - ▶ Novel model merging strategies creating unique capabilities.
 - ▶ LLMs aiding evolutionary computing through intelligent mutation and optimization.
- ▶ **Future opportunities:**
 - ▶ Extending evolutionary optimization to previously infeasible domains.
 - ▶ Evolutionary optimization of LLM architectures and parameters.

