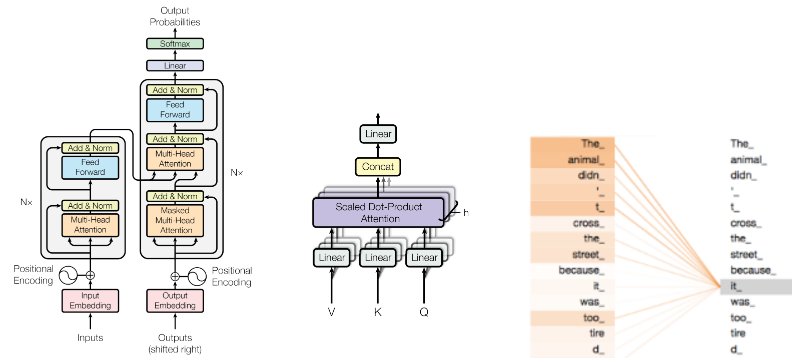
Synergies of Neuroevolution with Generative AI

Yujin Tang and Risto Miikkulainen

November 13, 2024

Introduction to Large Language Models (LLMs)

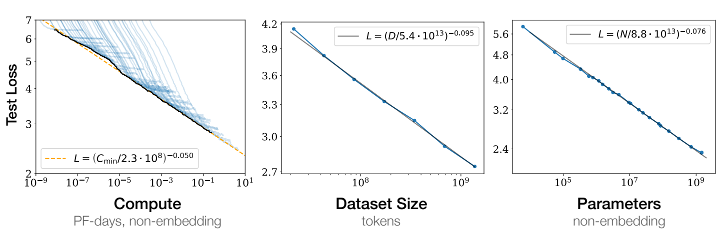
- Large Language Models (LLMs) such as GPT are key advancements in AI.
- LLMs generate human-like text and handle language-based tasks.
- Their foundation is the transformer architecture with self-attention.
- Self-attention helps manage long-range text dependencies efficiently.



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

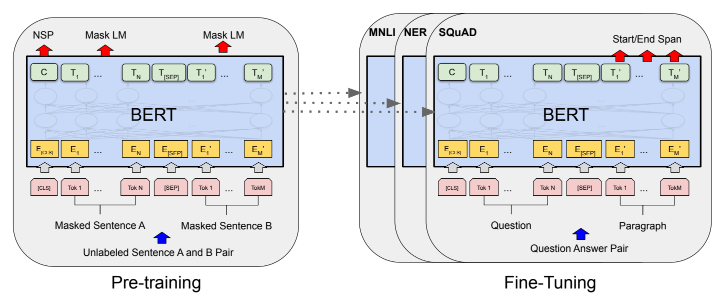
Scaling Laws and Applications of LLMs

- 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.



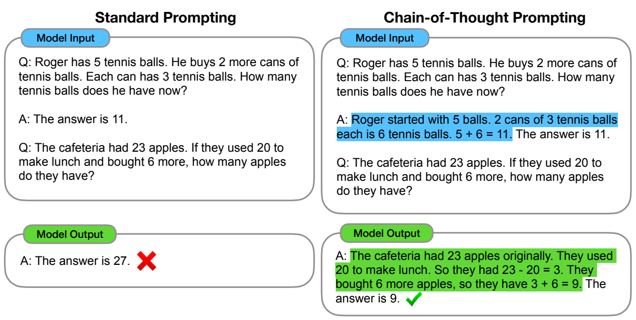
Training Large Language Models

- 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.



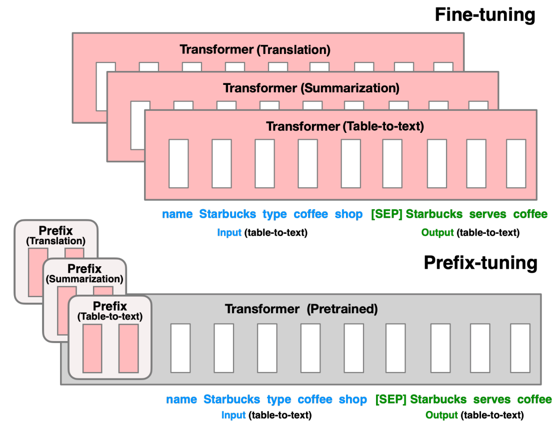
A New Field: Prompt Engineering

- 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.



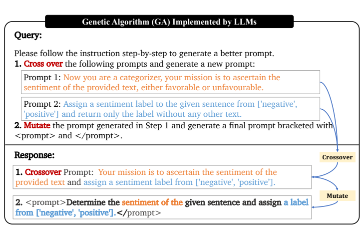
Challenges in Prompt Engineering

- 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.

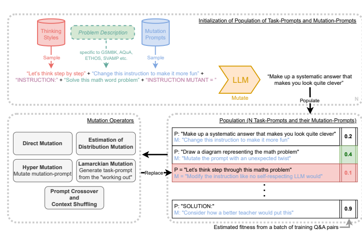


Evolutionary Algorithms for Prompt Optimization

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



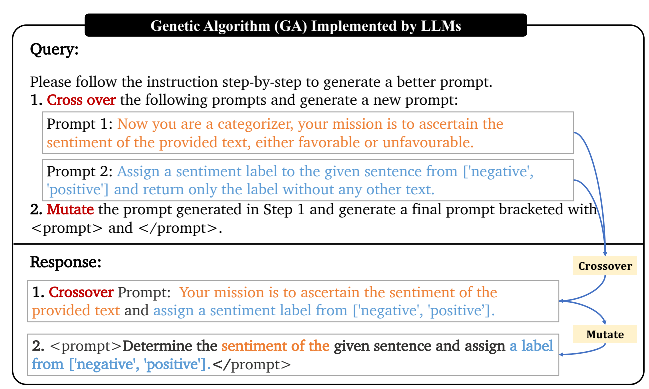
(a) EvoPrompt.



(b) Promptbreeder.

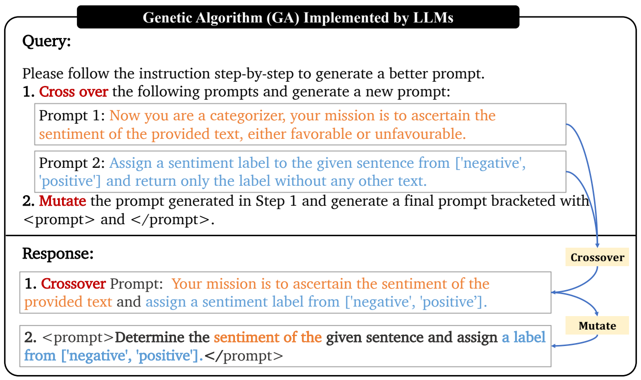
EvoPrompt Overview

- 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.



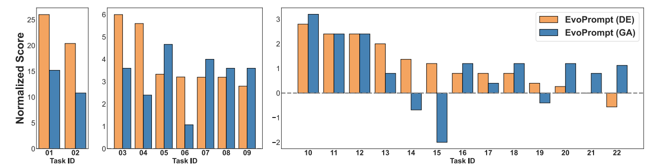
EvoPrompt: The Evolutionary Process

- 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.



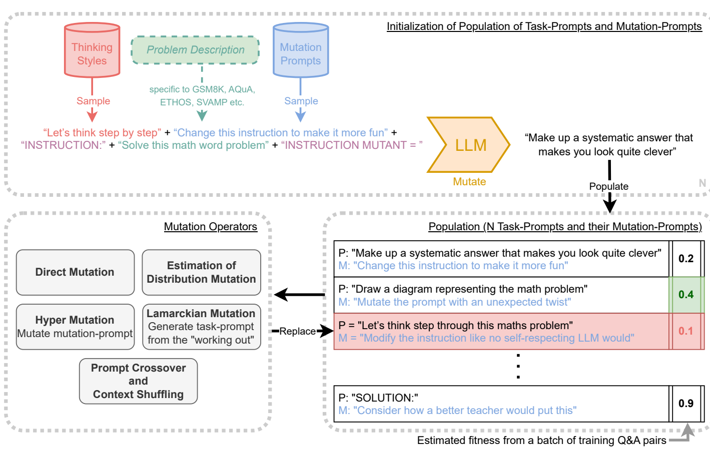
EvoPrompt Results on Benchmark Tasks

- 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%.



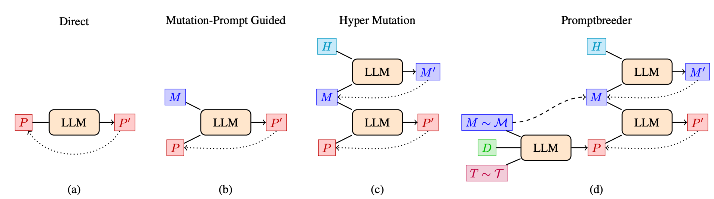
Promptbreeder: Optimizing Prompt Mutations as Well

- 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-	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 (ours)	PaLM 2-L	**100.0**	**98.9**	87.1	**93.7**	**80.2**	**85.9**	**64.6**	**83.5**

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."

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.

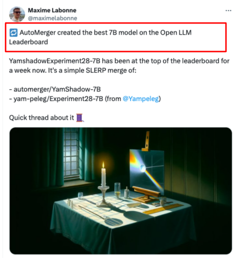
Aspect	EvoPrompt	PromptBreeder
Core Approach	Traditional genetic algorithm with simple text-based mutations and population-based evolution	Meta-learning system that evolves both prompts and mutation operators, using LLM for generating mutations
LLM Integration	Used primarily for fitness evaluation	Actively 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

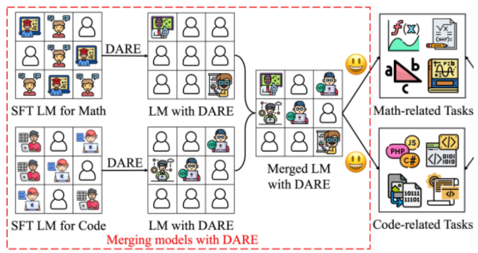
- AI development might follow a path similar to human collective intelligence.
- AI systems will consist of many small, specialized models instead of a single, all-knowing system.
- Open-source AI models demonstrate a promising trend of merging specialized models.

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.



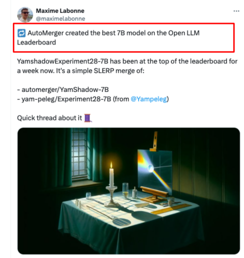
(a) Performance boost.



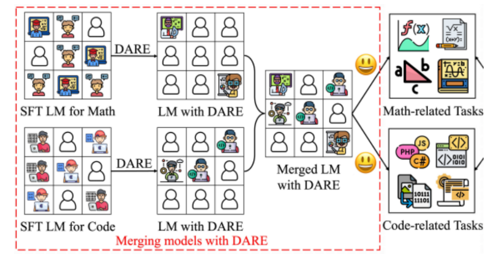
(b) Skill integration.

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.



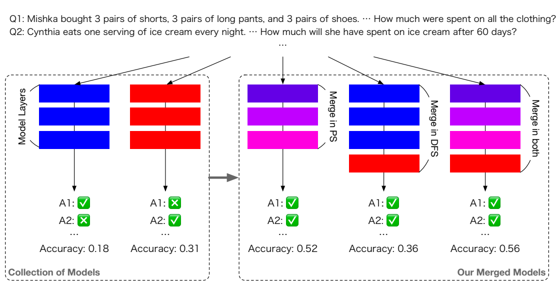
(a) Performance boost.



(b) Skill integration.

Evolutionary Model Merge Method

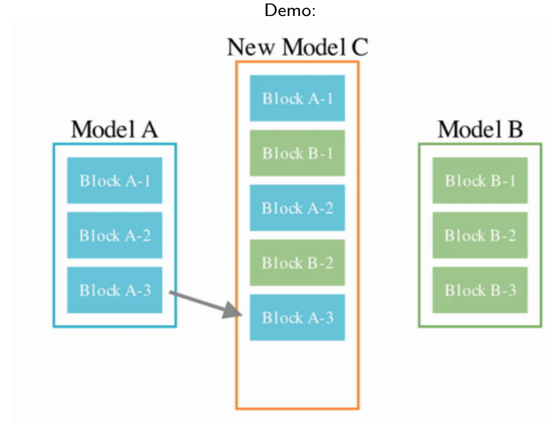
- Combines two approaches: Data Flow Space (Layer Merging) and Parameter Space (Weight Merging).
- Uses evolution to optimize how layers and weights from multiple models are merged.
- Capable of discovering architectural innovations beyond human intuition.



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.

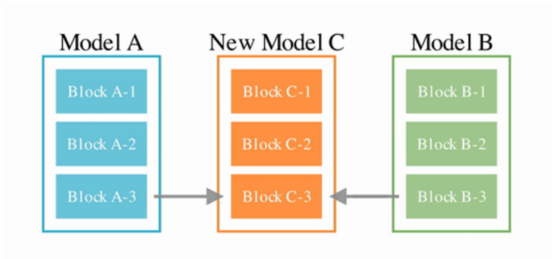
Demo:



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.

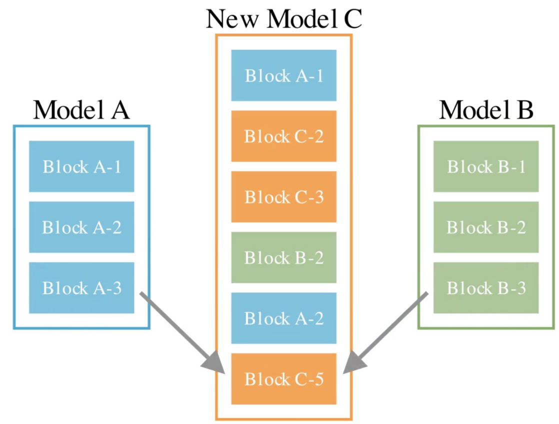
Demo:



Merging in Both Spaces

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

Demo:



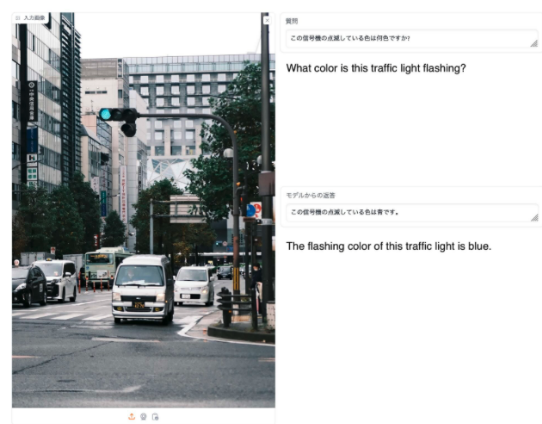
Case Study: Japanese Math LLM

- 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

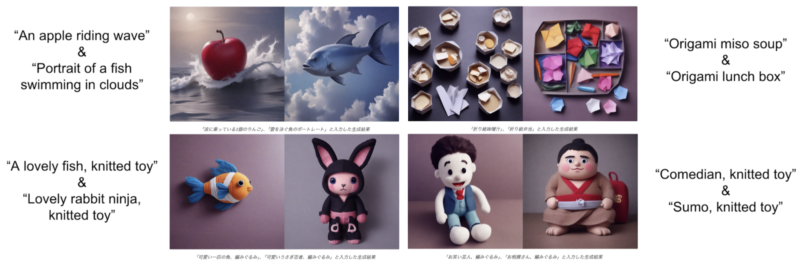
Case Study: Japanese Vision-Language Model

- 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.



Case Study: Merging Diffusion Models

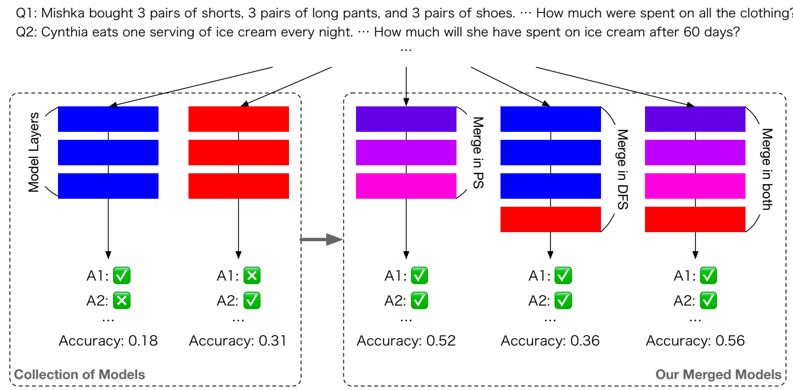
- Evolutionary merging applies to diffusion models (DM) as well.
- E.g. forming a 4-step DM that supports Japanese native prompts.
- Demonstrates the power of evolutionary algorithms in combining diverse models.



`https://sakana.ai/evolutionary-model-merge/`

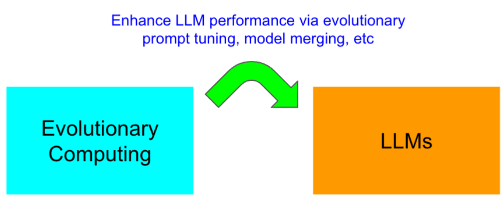
Conclusion: Evolutionary Model Merging

- Evolutionary algorithms can optimize model merging in both data flow and parameter spaces.
- Combined models can address niche domains with improved capabilities.
- Evolutionary merging enables AI models to go beyond human intuition and manual merging.



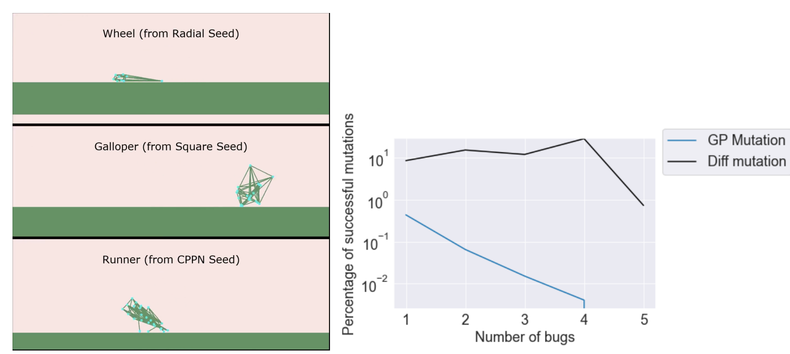
Conclusion #1: Evolutionary Computing Enhances LLMs

- 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.



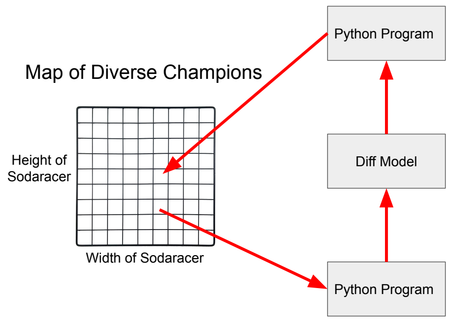
Going the Other Way: Evolution through LLMs

- 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.



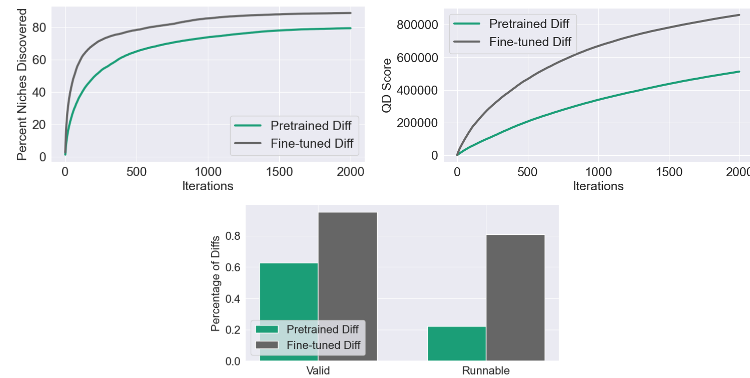
MAP-Elites with Diff Mutation

- 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.



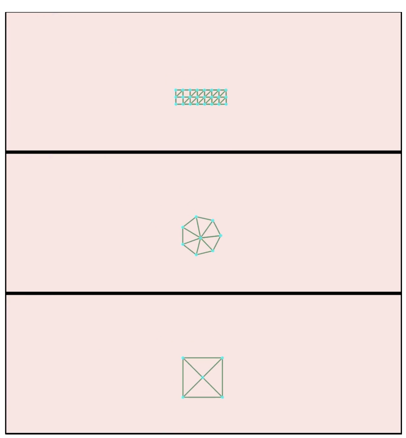
Fine-Tuning Diff Models

- 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

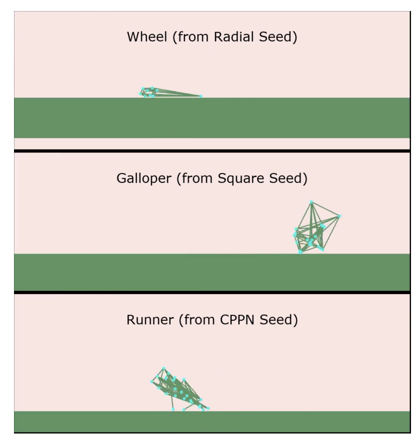
- Evolutionary search begins with seed solutions.
- LLMs enhance seed solutions by generating new body designs in evolutionary robotics.
- Seed solutions allow evolutionary exploration of complex designs.



`https://youtu.be/jeP8Nsulu48`

Novel Solutions in Evolutionary Search

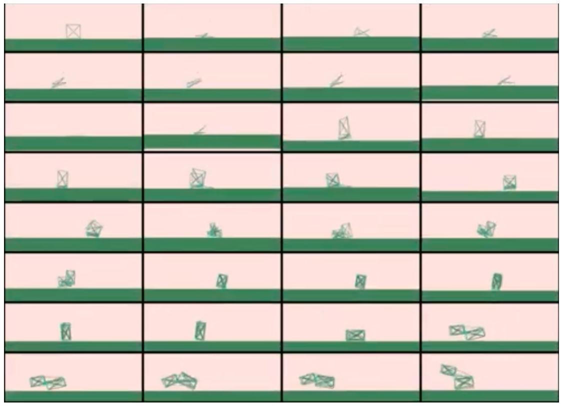
- From the seeds, distinctly different solutions evolve.



Seeded: `https://youtu.be/8C2K5fk28HI`
Varieties: `https://youtu.be/QNyNtvwA9FI`
Blob: `https://youtu.be/JDUAI8yrNcY`
Hopper: `https://youtu.be/noSPGFX5m3M`
Centipede: `https://youtu.be/zhMsPzo22do`

Exploration Process

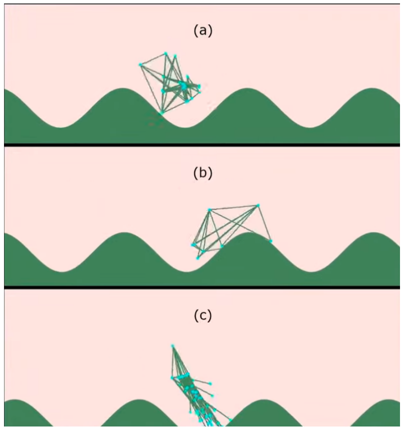
- The lineage of one of these solutions demonstrates exploration and refinement.



From square: `https://youtu.be/M9pAJuX6dyM`

Adjusting to Changes

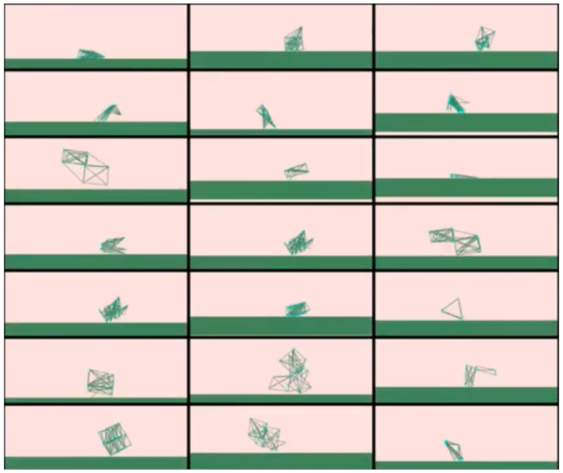
- Evolution can adjust to changes in the environment.



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`

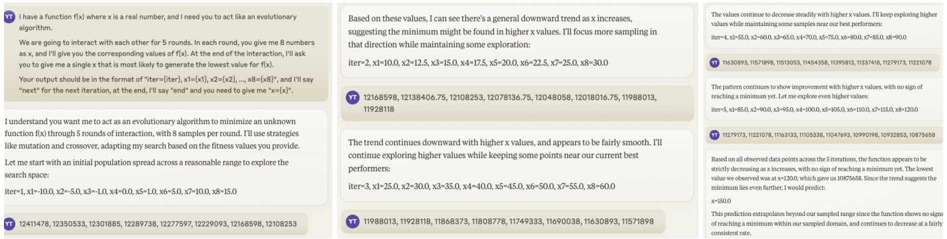
Conclusion: LLMs Enhance Evolutionary Computing

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



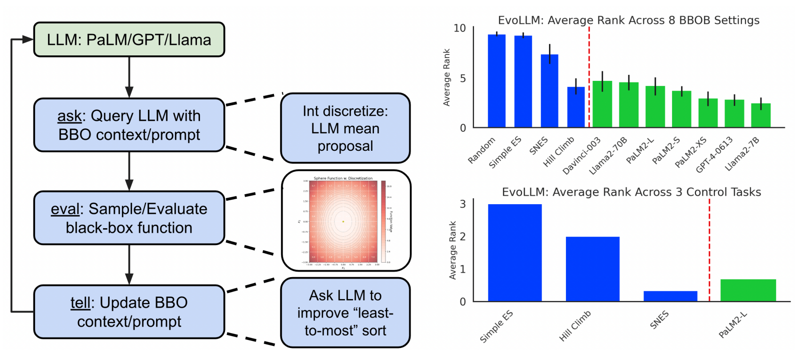
LLMs as Evolution Strategies

- 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:



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.



EvoLLM Design: High-Level Prompt Design

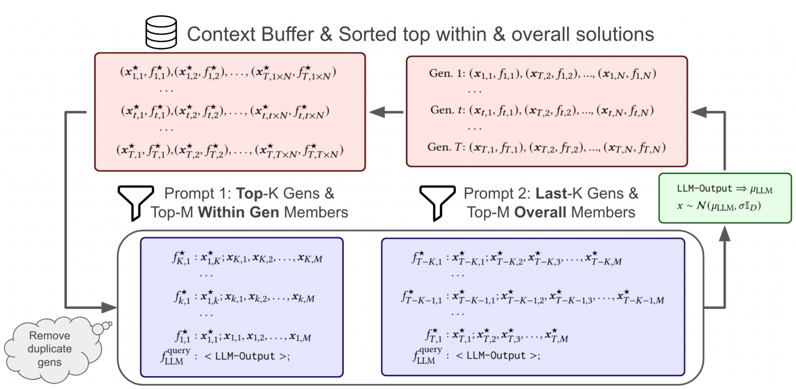
- EvoLLM represents solution candidates as integers.
- The LLM prompt includes top M solutions from K generations.
- An improvement query is added to the prompt.

$$
\begin{aligned}
f^{\star}_{K,1} &: \boldsymbol{x}^{\star}_{1,K}; \boldsymbol{x}_{K,1}, \boldsymbol{x}_{K,2}, \ldots, \boldsymbol{x}_{K,M} \\
&\ldots \\
f^{\star}_{k,1} &: \boldsymbol{x}^{\star}_{1,k}; \boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, \ldots, \boldsymbol{x}_{k,M} \\
&\ldots \\
f^{\star}_{1,1} &: \boldsymbol{x}^{\star}_{1,1}; \boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}, \ldots, \boldsymbol{x}_{1,M} \\
f^{\text{query}}_{\text{LLM}} &: < \texttt{LLM-Output} >;
\end{aligned}
$$

Figure: $x^{\star}_k$, $f^{\star}_k$ denotes the best-performing solution and its fitness up to generation k.

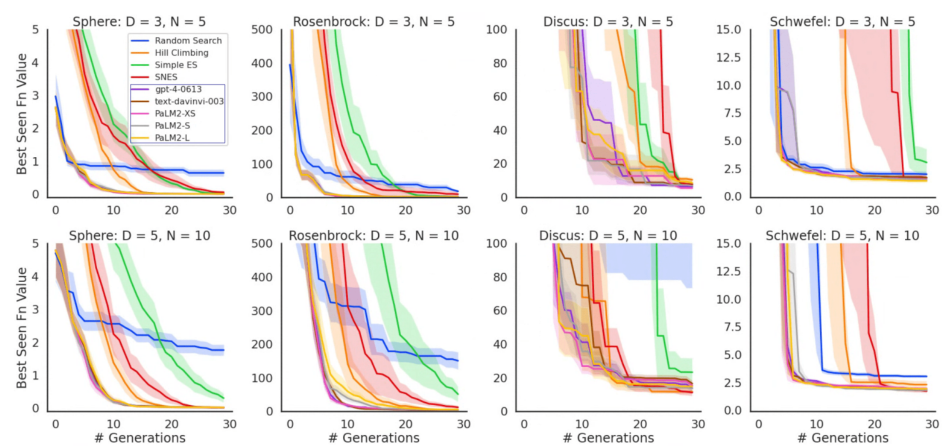
EvoLLM Process

- 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.



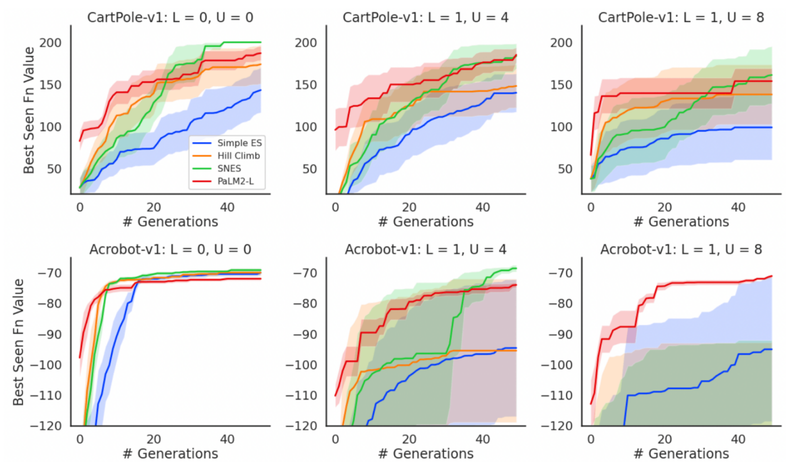
Evaluating EvoLLM Performance: BBOB Tasks

- 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.



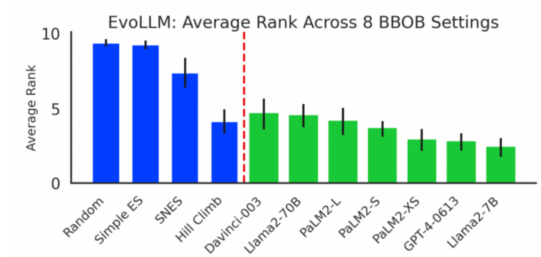
Evaluating EvoLLM Performance: Control Tasks

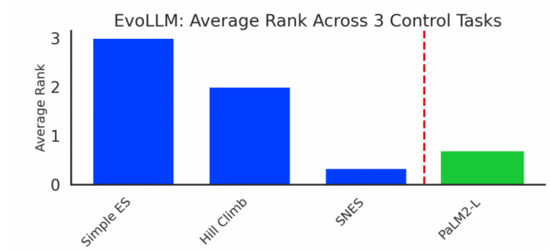
- Control tasks from OpenAI Gym (e.g., CartPole-v1 and Acrobot-v1).
- Tasks involve evolving 16 to 40 parameters of feedforward neural networks.
- EvoLLM achieves higher performance than baseline algorithms.



LLM Model Size and Compute

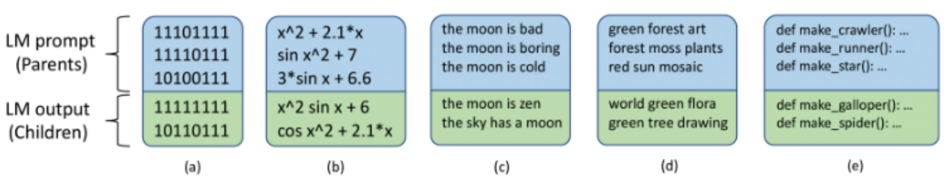
- Larger LLMs tend to perform worse than smaller models in EvoLLM.
- EvoLLM outperforms baselines even with smaller compute budgets.





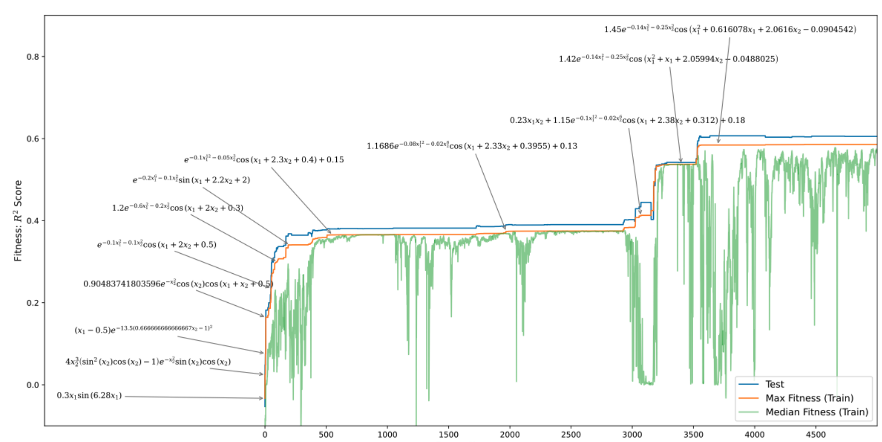
Generalizing to Language Model Crossover (LMX)

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



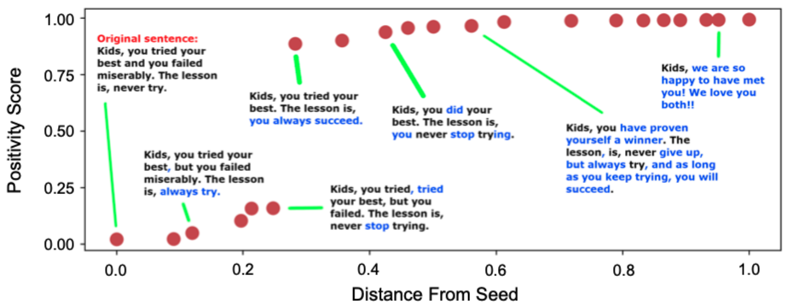
LMX in a Mathematical Domain

- 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.



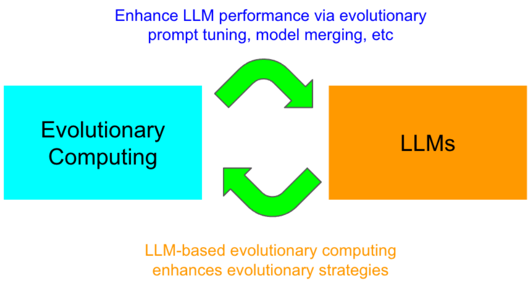
LMX in a Language Domain

- 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.



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.