

Mergenetic: a Simple Evolutionary Model Merging Library

Adrian Robert Minut^{1*}, Tommaso Mencattini^{2*}, Andrea Santilli¹, Donato Crisostomi¹, Emanuele Rodolà¹

¹Sapienza University of Rome ²Ecole Polytechnique Fédérale de Lausanne
minut@di.uniroma1.it

Abstract

Model merging allows combining the capabilities of existing models into a new one—post hoc, without additional training. This has made it increasingly popular thanks to its low cost and the availability of libraries that support merging on consumer GPUs. Recent work shows that pairing merging with evolutionary algorithms can boost performance, but no framework currently supports flexible experimentation with such strategies in language models. We introduce Mergenetic, an open-source library for evolutionary model merging. Mergenetic enables easy composition of merging methods and evolutionary algorithms, while incorporating lightweight fitness estimators to reduce evaluation costs. We describe its design and demonstrate that Mergenetic produces competitive results across tasks and languages using modest hardware. A video demo showcasing its main features is also provided¹.

 <https://github.com/tommasomncttn/mergenetic>

1 Introduction

Recent advances in large language models (LLMs) have shown that merging previously fine-tuned models can yield new systems with complementary strengths — often surpassing any single constituent (Yang et al., 2024). Rather than fully re-training from scratch or fine-tuning a large foundation model for every new task, merging techniques compose knowledge that is already encoded in existing checkpoints (e.g., specialized domain knowledge, multilingual abilities, or skills).

The accessibility of model merging has expanded significantly due to its inexpensive nature coupled with easy-to-use libraries like MergeKit (Goddard et al., 2024), enabling practitioners to produce competitive models from existing ones using standard consumer GPUs. Indeed, at the time of

* denotes equal contribution.

¹<https://youtu.be/lazoVeP7ku8>

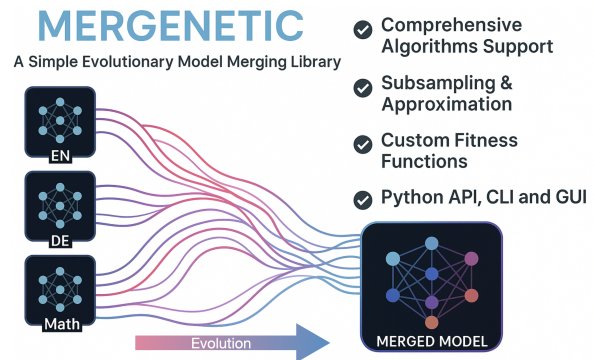


Figure 1: Mergenetic makes it easy to produce new state-of-the-art LLMs with minimal requirements.

writing, approximately 30% of models on the Hugging Face Open LLM Leaderboard (Fourrier et al., 2024) are merged models (Ilharco et al., 2022).

Recent research has shown that combining model merging with evolutionary algorithms can achieve superior performance (Akiba et al., 2025; Mencattini et al., 2025). However, this approach faces two key challenges: first, there is currently no library for experimenting with different evolutionary algorithms and merging methods; second, these methods typically require repeated computations on validation datasets to evaluate fitness functions, making them more computationally expensive than standard merging techniques. These limitations restrict access for the very user base that model merging was intended to empower.

In this paper, we introduce Mergenetic, a simple library to easily perform evolutionary model merging. Built on top of MergeKit (Goddard et al., 2024) and the widely used evolutionary framework PyMoo (Blank and Deb, 2020), our library provides:

1. **Python API, CLI, and GUI.** Mergenetic provides a flexible Python API for power users who wish to customize merging workflows, alongside a command-line interface (CLI) and a graphical user interface (GUI) for quick and

intuitive setup. Through the CLI or GUI, users can select models from the Hugging Face Hub, configure fitness functions, and launch merging experiments without writing code.

2. Comprehensive Algorithm Support.

Mergenetic integrates 19 evolutionary algorithms and a diverse set of merging strategies², enabling both single- and multi-objective optimization. This includes classical methods like genetic algorithms and state-of-the-art approaches such as NSGA-II (Deb et al., 2002a).

3. Subsampling & Approximation.

To reduce the overhead of fitness evaluations and support merging on consumer GPUs, Mergenetic allows for selective evaluation over dataset subsets and supports advanced approximation techniques for efficient fitness estimation (Mencattini et al., 2025; Polo et al., 2024).

4. Custom Fitness Functions.

The library seamlessly integrates with LM-Eval-Harness³ (Gao et al., 2024), offering out-of-the-box support for 8000+ tasks and metrics for fitness computation. Users can also define their own fitness routines tailored to specific needs.

Figure 1 and Table 1 summarize the key features of the library. By making evolutionary model merging more *efficient*, *configurable*, and *accessible*, Mergenetic expands the potential of merging as a truly *democratizing* technique.

In the remainder of this paper, we describe (i) the relevant background for Mergenetic, (ii) comparisons with existing solutions, (iii) its system architecture and workflow, and (iv) empirical evaluations featuring cross-lingual math merges and multi-task merges on publicly available LLMs. Finally, we conclude by discussing future extensions and potential broader impacts of this approach.

2 Background and Related Work

Model Merging. Model merging (Ainsworth et al., 2022; Crisostomi et al., 2025; Peña et al., 2023; Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2024; Matena and Raffel; Wortsman et al., 2022; Stoica et al.) has become a powerful and efficient alternative to ensembling, enabling the

²For all available merging methods refer to MergeKit.

³github.com/EleutherAI/lm-evaluation-harness

Features	Mergenetic (Ours)	MergeKit
Merging Algorithms	6	5 ⁴
Evolutionary Algorithms	19	1
Multi-objective	✓	✗
Dataset Subsampling	✓ (Random + Custom)	✗
Custom Fitness Functions	✓	✗
GUI	✓	✗

Table 1: Comparison of Mergenetic and MergeKit.

integration of existing models without requiring additional training. Mergenetic focuses on the multi-task scenario, where the aim is to merge different fine-tunings of a single pretrained model (Ilharco et al., 2022; Yadav et al., 2023; Yu et al., 2024; Matena and Raffel; Wortsman et al., 2022; Davari and Belilovsky, 2025; Wang et al., 2024; Zhou et al., 2024; Gargiulo et al., 2025; Akiba et al., 2025; Choshen et al., 2022).

Evolutionary Algorithms. Evolutionary Algorithms (EAs) are black-box optimization techniques that operate on a population of candidate solutions, evolving them over successive generations using operators such as selection, mutation, recombination, and crossover (Bäck and Schwefel, 1993; Pérowski and Ben-Hamida, 2017; Dasgupta and Michalewicz, 1997; Real et al., 2019; Vincent and Jidesh, 2023). A key component of EAs is the *fitness function*, which quantifies the quality of each candidate and steers the evolutionary process by promoting higher-performing solutions (Eiben and Smith, 2015). Applying EAs to model merging, evolutionary merging techniques (Akiba et al., 2025; Mencattini et al., 2025) automatically search for effective merging recipes using the performance of the merged model on a held-out validation dataset as the fitness function.

Comparison with other libraries. The most closely related library to Mergenetic is MergeKit (Goddard et al., 2024), which provides the underlying merging strategies (e.g., TIES, DARE, SLERP) that we build upon in our evolutionary pipelines. However, when it comes to search capabilities, MergeKit supports only a single evolutionary algorithm – CMA-ES (Hansen, 2023) – offering limited flexibility in how the optimization landscape is explored. In contrast, Mergenetic integrates with pymoo, enabling users to choose from a broad range of single- and multi-objective evolutionary strategies, as shown in Table 5.

⁴This number refers to the supported merging methods in evolutionary merging as per the [documentation](#).

```

from mergenetic.merging.linear_merger import LinearMerger
from mergenetic.optimization.merging_problem import MergingProblem
from pymoo.algorithms.soo.nonconvex.ga import GA
from mergenetic.searcher import Searcher

# Initialize the merger with base model, finetuned models, and output paths
merger = LinearMerger(run_id="demo_run",
                      path_to_base_model="my/base/model",
                      model_paths=["finetunedA", "finetunedB"],
                      path_to_store_yaml="configs/merging_config.yaml",
                      path_to_store_merged_model="merged_checkpoints/",
                      dtype="float16")

# Define the optimization problem for merging
problem = MergingProblem(
    merger = merger,           # Merger object
    searcher_df = my_dev_data, # Dataset used to compute fitness
    n_var = 2,                 # Number of variables (weights for the models)
    n_obj = 1                   # Number of objectives (usually a single metric)
)

algorithm = GA(pop_size=10) # Genetic algorithm with population size 10

# Create searcher to run GA over the merging problem
searcher = Searcher(problem, algorithm, results_path="results/",
                    n_iter=50, seed=42, run_id="demo_run")

searcher.search() # Run the evolutionary search for optimal weights
searcher.test()  # Evaluate the final merged model

```

Figure 2: Example on how to use the Python API for power users who wish to customize merging workflows.

Supported Merging Method	Multi-Model	Base Model
Task Arithmetic (Ilharco et al., 2023)	✓	✓
Model Soups (Wortsman et al., 2022)	✓	✗
SLERP	✗	✓
TIES (Yadav et al., 2023)	✓	✓
DARE (Yu et al., 2024) + TIES	✓	✓
DARE (Yu et al., 2024) + Task Arithmetic	✓	✓

Table 2: Tested merging methods in Mergenetic

Most importantly, MergeKit assumes that the fitness function must be computed over the full evaluation dataset, which significantly increases runtime and computational demands – often making the entire process impractical on consumer hardware. In contrast, Mergenetic supports sub-sampled evaluation and advanced fitness estimation techniques (e.g., IRT-based estimators (Polo et al., 2024; Mencattini et al., 2025)), dramatically reducing evaluation cost and enabling high-quality merging to be performed efficiently, even on a single GPU.

3 Design and guiding principles

The design of Mergenetic reflects our goal of supporting evolutionary model-merging experiments on consumer hardware. We outline the guiding principles that drove our design decisions before diving into key modules and functionalities in §4.

Research-Oriented A central motivation for Mergenetic is to enable *researchers* to easily explore and compare different evolutionary algorithms, merging strategies, and optimization objectives. Rather than locking users into a fixed routine, Mergenetic supports a flexible mix of merging methods (e.g., TIES, DARE, SLERP from MergeKit (Goddard et al., 2024)), evolutionary algorithms (e.g., GA, NSGA-II, DE from PyMoo (Blank and Deb, 2020)), and evaluation backends (e.g., LM-Eval-Harness or user-defined). This modularity supports systematic experimentation, such as comparing single- vs. multi-objective merges or testing different data sampling strategies – and allows defining custom objectives.

User-Friendly To democratize model merging for researchers and practitioners with standard GPU setups, Mergenetic is designed to be both configuration-centric and user-friendly. Users can define merges, tasks, algorithms, and evaluators using simple YAML files, a command-line interface, or an interactive GUI — minimizing the engineering overhead typical of large-scale experiments. The library is optimized for consumer GPUs by supporting approximate evaluation methods (e.g., IRT-

based estimators), dataset sub-sampling, and partial model loading. It integrates seamlessly with LM-Eval-Harness, supporting more than 8000 tasks and metrics already defined in the library (e.g., GSM8K and ARC), while also making it easy to plug in custom datasets and evaluations for fitness computation. Together, these features enable meaningful evolutionary merging on a single GPU, lowering the barrier for smaller research groups and individual practitioners.

4 Mergenetic

Modules and Functionalities The implementation relies on MergeKit (Goddard et al., 2024) for merging the models, PyMoo (Blank and Deb, 2020) for optimizing the objective function through evolutionary algorithms, and LM-Eval-Harness (Gao et al., 2024) for implementing some of the fitness functions. In table 2 we outline the supported merging methods, while in table 5 we outline the currently available evolutionary algorithms.

The Mergenetic library is divided into distinct modules that reflect the core stages of evolutionary model merging: (i) defining the workflow (Python API, CLI, GUI), (ii) performing the merge of the models (`Merger`), (iii) formulating the optimization problem (`Optimization`) as a `MergingProblem` , (iv) evaluating merged models (`Evaluator`), and (v) orchestrating the evolution loop (`Searcher`). Below, we briefly describe each module and link it to the broader system design.

4.1 Python API, CLI, and GUI

Python API. Figure 2 provides an example usage of the API. The `Searcher` and `Problem` classes form the core of the Python API. Users can instantiate an optimization *problem* (e.g., merging multiple language models), select an algorithm from PyMoo, and call `searcher.search()` to launch the evolutionary procedure. A typical workflow involves:

1. Defining *evaluation datasets* and relevant *performance metrics* through an `Evaluator` .
2. Instantiating a `Merger` to specify how weights are combined.
3. Passing these to a `MergingProblem` class, describing the evolutionary search space and the experiment’s objectives.
4. Choosing a `GeneticAlgorithm` (e.g., NSGA-II, GA, DE) from PyMoo.

5. Running the search through the `Searcher` , optionally calling `.test()` on the best solutions.

CLI. For users who prefer a command-line approach without manually writing scripts, the Mergenetic CLI is invoked via:

```
python mergenetic.py -eval-method <lm-eval|custom> -merge-type <single|multi>
```

Internally, it launches an interactive wizard to guide users through selecting *models*, *tasks*, *algorithms*, and *merging methods*. The CLI can handle four main modes: single- or multi-language merges, each with either LM-Eval-Harness or *custom* evaluations. By abstracting away many details, the CLI lets users prototype merges quickly with no code.

GUI. A Gradio⁵-based (Abid et al., 2019) graphical interface provides a further layer of accessibility, especially for non-technical users or demonstration purposes (See Fig. 3). It reuses the same core configuration concepts but wraps them in a step-by-step wizard: (1) load base model(s), (2) specify tasks/languages, (3) set evolutionary parameters, and (4) run merging with real-time logs. The GUI allows merging without coding.

4.2 Core components

The core components are as follows.

4.2.1 Merger

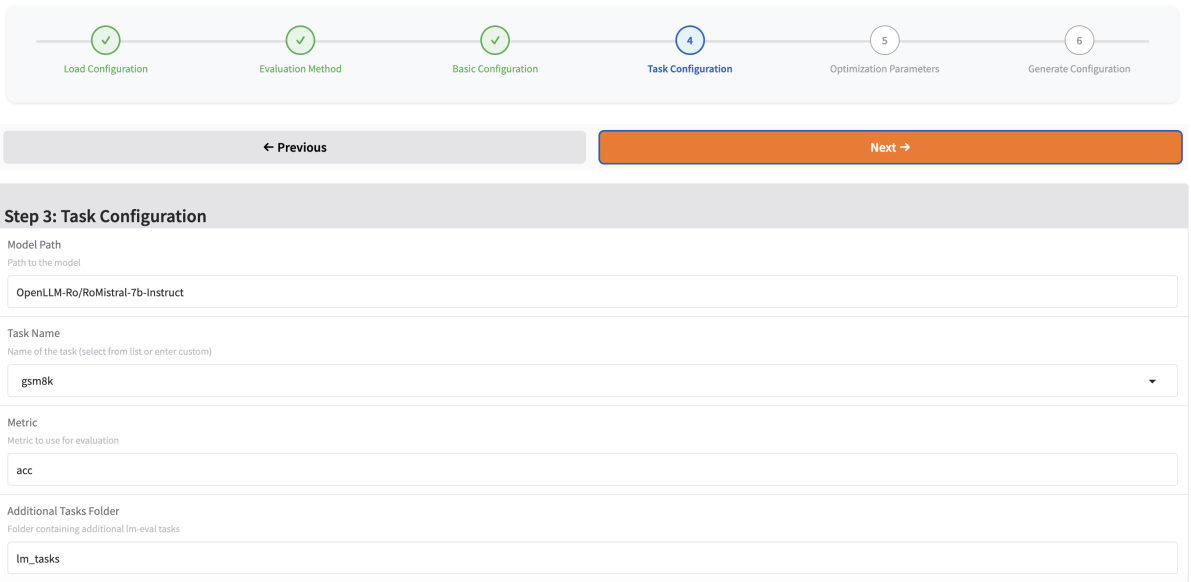
The `Merger` module handles the core weight-combination logic by interfacing with MergeKit. Each `merger` class (e.g., `SlerpMerger` , `TiesDareMerger` , `TaskArithmeticMerger`) generates a YAML configuration specifying the base checkpoints, interpolation method, and merge coefficients. This configuration is passed to MergeKit, which performs the actual merging and produces a new model checkpoint. The merger supports both standard and multi-model merges, including advanced strategies like TIES combined with DARE (Yadav et al., 2023; Yu et al., 2024). Additionally, Mergenetic manages GPU memory during the evolutionary search, helping avoid out-of-memory errors. During optimization, the evolutionary algorithm proposes weight combinations, which the merger translates into actual models ready for evaluation.

⁵<https://github.com/gradio-app/gradio>

Mergenetic Model Merging Interface

Configure and run model merging experiments with this visual interface.

Configuration Execution



Step 3: Task Configuration

Model Path
Path to the model
OpenLLM-Ro/RoMistral-7b-Instruct

Task Name
Name of the task (select from list or enter custom)
gsm8k

Metric
Metric to use for evaluation
acc

Additional Tasks Folder
Folder containing additional lm-eval tasks
lm_tasks

Figure 3: Screenshot of the Gradio-based GUI described in section 4.1. The user is guided through a step-by-step process to define every ingredient of the evolutionary merging pipeline.

4.2.2 Optimization

At the core of `Mergenetic`, the optimization module casts model merging as a *black-box optimization* problem. The decision variables correspond to the targeted parameters from the merging configuration file (the genotype), such as the interpolation or pruning coefficients. Objective functions define the fitness criteria to be optimized, such as accuracy, perplexity, or other task-specific metrics.

The `MergingProblem` class defines how to: (i) Convert a genotype to a merged model (by calling the `Merger`). (ii) Evaluate the merged model via an `Evaluator`. (iii) Return the resulting fitness or multi-objective scores to the algorithm.

Through `PyMoo` (Blank and Deb, 2020), we support both **single-** and **multi-objective** methods. Single-objective approaches optimize *one* metric (e.g., math accuracy). Multi-objective strategies balance multiple metrics, e.g., *math accuracy* and *general fluency*, and find a Pareto front of suitable models, allowing the final user to choose based on their preference of the individual metrics.

4.2.3 Evaluator

Evaluators compute a merged model’s performance on the chosen task(s). In `Mergenetic`, they appear both as *direct evaluators* (e.g., running on a small

dataset) or as *IRT-based estimators* using anchors (Mencattini et al., 2025). In particular, we highlight two categories of Evaluators:

LM-Eval-Harness Evaluators. `Mergenetic` can natively call out to the `LM-Eval-Harness` (Gao et al., 2024) library, passing the merged checkpoint and a chosen benchmark (e.g., ARC, GSM8K). This approach covers many standard tasks and yields consistent comparisons. However, it can be relatively expensive if one repeatedly evaluates large datasets on many candidate merges. To offset this problem, `Mergenetic` wraps `LM-Eval-Harness` and allows explicit subsamples through the plug-and-play `ConfigPE`, which allows subsampling without the need to instantiate a new `LM-Eval-Harness` config file.

Custom Evaluators. Users can alternatively define their own logic for computing correctness—e.g., `MultilingualMathFGEvaluator` that checks whether the extracted answer is correct *and* in the target language. Or a `MultipleChoiceEvaluator` that compares the chosen letter (A, B, C, D) to the ground truth. These evaluators easily allow advanced users to combine partial correctness checks with domain constraints (e.g., “the predicted chemical formula must be balanced”).

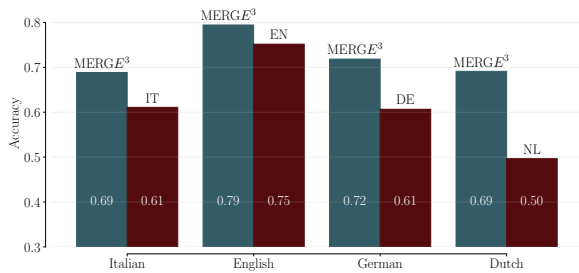


Figure 4: Evolving a multi-lingual model spanning Italian, English, German and Dutch.

4.2.4 Searcher

Finally, the `Searcher` orchestrates the evolutionary loop: it begins with the **initialization** of a population of random genotypes (weight vectors), followed by **merging/evaluation**, where each genotype is merged into a checkpoint and scored on user-specified tasks/datasets. Then comes **selection/variation**, where parent genotypes are chosen based on fitness and modified via crossover and mutation to produce children. Steps 2 and 3 repeat for T generations in the main **loop**. Therefore, the `Searcher` class essentially wraps all these elements (`Problem`, `Merger`, `Evaluator`, and PyMoo’s `Algorithm`) in an easy-to-use API.

During the search process, intermediate results (population genotypes, partial solutions, logs) are stored in `CSV` or `JSON` files, facilitating real-time monitoring. At completion, `test()` re-merges the best solutions and evaluates them on an unseen test set to quantify final performance.

5 Case Studies

To demonstrate the capabilities of the `Mergenetic` library, we reproduce here two evolutionary model merging pipelines: `MERGE3` (Mencattini et al., 2025) and `EvoLLM-JP` (Akiba et al., 2025). For an in-depth analysis of the performance improvements provided by evolutionary model merging over standard merging strategies, we refer the reader to the original works by Akiba et al. (2025) and Mencattini et al. (2025). Additionally, for a detailed treatment of estimator-based fitness approximations and their effectiveness, we refer to the estimator analysis in Mencattini et al. (2025).

5.1 Evolving a multi-lingual model

We demonstrate how `Mergenetic` can be used to merge individually fine-tuned models for four languages — Italian, English, German, and Dutch —

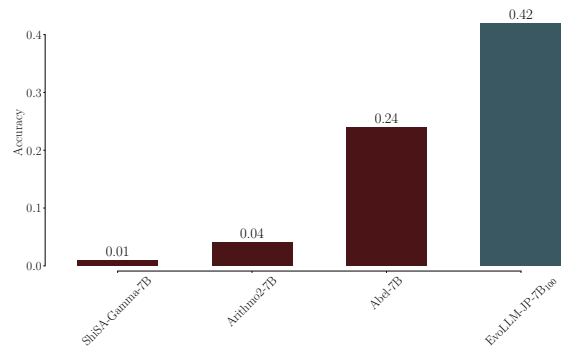


Figure 5: Cross-lingual transfer of math solving capabilities from English to Japanese.

into a single multilingual model. This setup formulates the objective function as explicitly multi-task, assigning one evaluation metric per language to promote balanced cross-lingual performance. Details on the specific models used per language are provided in Appendix A.2. As shown in fig. 4, the merged model consistently outperforms each of its language-specific constituents, achieving up to a 19% accuracy gain on the ARC-Challenge benchmark (Clark et al., 2018). Notably, it surpasses all endpoints across the board, highlighting the effectiveness of evolutionary merging in facilitating positive knowledge transfer across languages.

5.2 Cross-lingual transfer

To showcase the ability of `Mergenetic` to support cross-lingual skill transfer, we merge a math-specialized English model with a Japanese fine-tuned version of `Mistral-7B` (Jiang et al., 2023), and evaluate the result on the Japanese translation of the `GSM8K` dataset (Cobbe et al., 2021). This experiment follows the general setup proposed by Akiba et al. (2025), using a subset of 100 samples for the fitness evaluation instead of the full dataset. As shown in fig. 5, the merged model achieves a 10-20% accuracy improvement over each of its individual components, demonstrating effective cross-lingual transfer enabled by evolutionary merging.

5.3 Technical Analysis

We conducted additional experiments to assess the practicality of evolutionary model merging using `Mergenetic` on different GPU models. Specifically, we measured evaluation and merging run-times across three common GPUs: `NVIDIA 3090`, `4090`, and `V100`, using `Mistral-7B` (Jiang et al., 2023) in 4-bit precision with `SLERP` on 10 examples. The results, summarized in Table 3, show

that Mergenetic achieves practical runtimes even on consumer architectures. While the NVIDIA 4090 yields the fastest evaluation (45s) and merging (160s), both the 3090 and V100 maintain feasible execution times, underscoring accessibility for users with varying hardware.

Table 3: Evaluation and merge times, in seconds, across different GPU models. We merged Mistral-7B fine-tuned models, using 10 samples per fitness computation.

GPU Model	Eval Time (s)	Merge Time (s)
NVIDIA 3090 24GB	65	135
NVIDIA 4090 24GB	45	160
NVIDIA V100 32GB	80	220

Table 4: Throughput in evaluated models per hour for different sample sizes per fitness computation on GSM8K. A single NVIDIA 4090 with 24GB of VRAM was used.

Sample size	1000	100	50	30	20
Throughput (Models/Hour)	0.67	8.33	14.17	16.67	17.08

To evaluate scalability, we also assessed the throughput of model evaluation under different dataset sizes using a 4090 with 24GB VRAM. For full evaluations on 1000 samples, throughput was 0.67 complete-model-evaluations per hour, while smaller sample sizes yielded up to 17 models/hour. While more extensive studies with both larger models (e.g., 70B parameters) and lower-end GPUs should be analyzed, these findings support the use of Mergenetic for efficient experimentation even on single consumer-grade GPUs, making evolutionary merging widely accessible.

6 Conclusions

Mergenetic bridges the gap between cutting-edge evolutionary model merging and practical usability on consumer hardware. By combining flexible merging strategies, diverse evolutionary algorithms, and lightweight fitness approximators, it empowers researchers and practitioners to explore high-quality model compositions without requiring large-scale infrastructure. While more extensive studies that include both larger models and lower-tier GPUs are still warranted, our current results already demonstrate that Mergenetic enables efficient experimentation even on a single consumer-grade GPU, making evolutionary merging broadly accessible. Through its Python API, CLI, and GUI,

Mergenetic supports both systematic experimentation and user-friendly workflows. We hope the library will serve as a stepping stone for future research in multilingual, multi-task, and efficient evolutionary model merging, and invite the community to build upon and extend its capabilities.

Limitations

While Mergenetic significantly lowers the entry barrier for evolutionary model merging, several limitations remain:

Dependence on Existing Fine-Tuned Models.

Model merging requires access to pre-trained or fine-tuned base models with relevant capabilities (e.g., math reasoning, language-specific fluency). As such, the technique currently cannot be directly applied to extremely low-resource languages or domains where such models are unavailable. This limits its immediate applicability in truly zero-resource settings. Future work could explore integrating lightweight fine-tuning or retrieval-based augmentation prior to merging to alleviate this dependency.

Hardware Requirements.

Although we designed Mergenetic for consumer-grade GPUs, it still requires relatively high-tier hardware (e.g., NVIDIA RTX 2080 or better) due to the size of language models involved and the need to load and evaluate them during evolution. Most laptops or low-memory GPUs may not have sufficient VRAM to support repeated merging and evaluation steps. We see this as a broader limitation of current LLM infrastructure and hope that advances in model quantization, sparse evaluation, and efficient loading techniques will further democratize access to frontier AI tools like Mergenetic.

LLM-Centric Design.

While the foundational methods behind model merging and evolutionary optimization are, in theory, applicable across various domains, the current implementation of Mergenetic is limited to large language models (LLMs). This constraint primarily arises from its dependence on MergeKit (Goddard et al., 2024) as the core merging backend, which is specifically tailored for transformer-based LLMs and lacks robust support for models in other modalities such as vision or speech. Consequently, Mergenetic inherits this modality-specific restriction. Future extensions could consider adapting or substituting the backend to enable broader applicability across diverse model architectures and domains.

References

- Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*.
- Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2022. Git Re-Basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*.
- Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2025. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*.
- J. Blank and K. Deb. 2020. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509.
- Thomas Bäck and Hans-Paul Schwefel. 1993. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23.
- Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. 2022. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Donato Crisostomi, Marco Fumero, Daniele Baieri, Florian Bernard, and Emanuele Rodolà. 2025. c^2m^3 : Cycle-consistent multi-model merging. In *Advances in Neural Information Processing Systems*, volume 37.
- Dipankar Dasgupta and Zbigniew Michalewicz. 1997. Evolutionary algorithms: an overview. *Evolutionary algorithms in engineering applications*, pages 3–28.
- MohammadReza Davari and Eugene Belilovsky. 2025. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pages 270–287. Springer.
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002a. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002b. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. 2007. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07*, page 11871194, New York, NY, USA. Association for Computing Machinery.
- A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing*. Springer.
- Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.
- Antonio Andrea Gargiulo, Donato Crisostomi, Maria Sofia Bucarelli, Simone Scardapane, Fabrizio Silvestri, and Emanuele Rodolà. 2025. Task singular vectors: Reducing task interference in model merging. *Preprint*, arXiv:2412.00081.
- Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee’s MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.
- N. Hansen. 2023. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.
- G. Ilharco, M.T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. 2023. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hananeh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *The Eleventh International Conference on Learning Representations*.
- A.Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D.S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L.R. Lavaud, M.-A. Lachaux, P. Stock, T. Le Scao, T. Lavril, T. Wang, T. Lacroix, and W. El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

- Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging.
- Tommaso Mencattini, Adrian Robert Minut, Donato Crisostomi, Andrea Santilli, and Emanuele Rodolà. 2025. [Merge³: Efficient evolutionary merging on consumer-grade gpus](#). *Preprint*, arXiv:2502.10436.
- Fidel A Guerrero Peña, Heitor Rapela Medeiros, Thomas Dubail, Masih Aminbeidokhti, Eric Granger, and Marco Pedersoli. 2023. Re-basin via implicit sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20237–20246.
- Alain Pétrowski and Sana Ben-Hamida. 2017. *Evolutionary algorithms*. John Wiley & Sons.
- Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinybenchmarks: evaluating llms with fewer examples. In *Forty-first International Conference on Machine Learning*.
- Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.
- Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.
- George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*.
- Klaudia Thellmann, Bernhard Stadler, Michael Fromm, Jasper Schulze Buschhoff, Alex Jude, Fabio Barth, Johannes Leveling, Nicolas Flores-Herr, Joachim Köhler, René Jäkel, and Mehdi Ali. 2024. [Towards cross-lingual llm evaluation for european languages](#). *Preprint*, arXiv:2410.08928.
- Amala Mary Vincent and P Jidesh. 2023. An improved hyperparameter optimization framework for automl systems using evolutionary algorithms. *Scientific Reports*, 13(1):4737.
- Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. 2024. [Localizing task information for improved model merging and compression](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50268–50287. PMLR.
- Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. [Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time](#). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.
- Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, pages 7093–7115. Curran Associates, Inc.
- Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.
- Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. [Language models are super mario: Absorbing abilities from homologous models as a free lunch](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 57755–57775. PMLR.
- Luca Zhou, Daniele Solombrino, Donato Crisostomi, Maria Sofia Bucarelli, Fabrizio Silvestri, and Emanuele Rodolà. 2024. [Atm: Improving model merging by alternating tuning and merging](#). *arXiv preprint arXiv:2411.03055*.

A Additional Details

A.1 Cross-Lingual Case Study Details

For the cross-lingual case study, we conduct evolutionary search on the Japanese subset of the MGSM dataset (Shi et al., 2022), a multilingual extension of GSM8K (Cobbe et al., 2021). The final merged model is evaluated on the MGSM test set, following the evaluation protocol of Akiba et al. (2025). Unlike their setup, which used 1069 search datapoints (the remaining part of the GSM8K test set that was not included in MGSM), we use only a subset of 100 examples for computational efficiency. Our approach employs a single-objective evolutionary algorithm based on a Genetic Algorithm (Dasgupta and Michalewicz, 1997), incorporating a Simulated Binary Crossover (SBX) operator (Deb et al., 2007) for recombination and a Polynomial Mutation operator (Deb et al., 2007) for exploration. We set the population size to 25 and run the algorithm for 7 generations. Fitness and evaluation metrics are computed by extracting the final numeric answer using a regular expression and verifying both the mathematical correctness and the linguistic accuracy of each response. Language identification is performed using the method described in (Joulin et al., 2016). Only responses that are both mathematically and linguistically correct are considered valid. The models evaluated in this experiment include Arithmo2-Mistral-7B, Abel-7B-002, and shisa-gamma-7b-v1.

A.2 Multilingual case study details

For the multilingual case study, we perform evolutionary model merging across four languages — Italian, Dutch, German, and English — using the translated ARC dataset from the Hugging Face repository (Thellmann et al., 2024)⁶. We employ a multi-objective optimization setup with NSGA-II (Deb et al., 2002b), configuring the evolutionary process with a population size of 25 and 7 iterations. As the merging strategy, we use a combination of TIES and DARE. The fitness and test evaluations are performed by extracting the final answer choice (A, B, C, or D) from the model’s output using a regular expression. For each language, we use a reduced dataset of 20 translated examples from ARC to compute fitness scores, keeping the process efficient and GPU-friendly. The models used in this experiment are Mistral-Ita-7B, GEITje-

7B-ultra, leo-mistral-hessianai-7B, and the base model is Mistral-7B-v0.1.

A.3 Supported evolutionary algorithms

Table 5 lists all the evolutionary algorithms provided by PyMoo and hence supported in Mergenetic, stating whether they are single- or multi-objective and if they allow constraints to be defined, along with a brief description.

A.4 Performance Estimator

To reduce the computational cost associated with evaluating the fitness of candidate models during evolutionary merging, the Mergenetic library supports estimator-based approximations inspired by Mencattini et al. (2025) and Polo et al. (2024). These methods allow us to estimate model performance using a reduced subset of the evaluation dataset, significantly accelerating the evolution process without sacrificing accuracy.

In particular, Mergenetic provides implementations of both standard and model merging-specific IRT-based estimators, which leverage latent ability inference to approximate full-dataset correctness. These estimators vary in their assumptions and complexity, offering a trade-off between computational efficiency and estimation fidelity.

Table 6 provides an overview of the currently supported estimators, including a brief description and a qualitative rating of their performance.

A.5 License

The library is licensed under Apache 2.0. This means that it can be freely used, modified, and redistributed by anyone, including for commercial purposes. The license is designed to promote widespread adoption by offering a permissive legal framework that imposes minimal restrictions on end users. Developers are allowed to modify the source code and distribute derivative works under different terms, provided that the original license and copyright notice are retained. Mergenetic builds upon two key dependencies: PyMoo and MergeKit. The former is distributed under the same Apache 2.0 license as Mergenetic, ensuring compatibility and permissive use. However, MergeKit introduces additional licensing constraints in versions beyond v0.1.0. Specifically, it adopts a Business Source License, which restricts production use based on organizational scale and revenue. Users intending to deploy Mergenetic for commercial purposes are advised

⁶<https://huggingface.co/openGPT-X/arcx>

Algorithm	Class	Obj.	Constr.	Description
Genetic Algorithm	GA	single	✓	Customizable evol. operators for broad problem categories
Differential Evol.	DE	single	✓	Variants for continuous global optimization
BRKGA	BRKGA	single	✓	Advanced variable encoding for combinatorial opt.
Nelder Mead	NelderMead	single	✓	Point-based algorithm using simplex operations
Pattern Search	PatternSearch	single	✓	Iterative approach with exploration patterns
CMAES	CMAES	single		Model-based sampling from dynamic normal distribution
Evol. Strategy	ES	single		Real-valued optimization strategy
SRES	SRES	single	✓	ES with stochastic ranking constraint handling
ISRES	ISRES	single	✓	Improved SRES for dependent variables
NSGA-II	NSGA2	multi	✓	Non-dominated sorting and crowding
R-NSGA-II	RNSGA2	multi	✓	NSGA-II with reference points
NSGA-III	NSGA3	many	✓	NSGA-II for many-objective problems
U-NSGA-III	UNSGA3	many	✓	NSGA-III optimized for fewer objectives
R-NSGA-III	RNSGA3	many	✓	NSGA-III with aspiration points
MOEAD	MOEAD	many		Multi-objective optimization via decomposition
AGE-MOEA	AGEMOEA	many		Estimates Pareto-front shape instead of crowding
C-TAEA	CTAEA	many	✓	Sophisticated constraint-handling for many objectives
SMS-EMOA	CTAEA	many	✓	Uses hypervolume during environmental survival
RVEA	RVEA	many	✓	Reference direction with angle-penalized metric

Table 5: Supported Optimization Algorithms and their description.

Estimator	Description	Performance
Random	Baseline estimator using random sample correctness. Simple but noisy and unreliable.	★★
P-IRT	Standard Item Response Theory estimator, uses subset to estimate ability, not tailored for merging.	★★★
GP-IRT	Generalized P-IRT with better smoothing but still not designed for merging.	★★★
MP-IRT	MERGE3’s merged-performance IRT estimator assuming linear combination of abilities.	★★★★
GMP-IRT	Generalized version of MP-IRT, combines predictions and observations with learned weights.	★★★★
Full Dataset	Ground truth performance by running evaluation on the full dataset.	★★★★★

Table 6: Comparison of different performance estimators.

to review these terms carefully and install a version of MergeKit that aligns with their intended usage scenario. If unrestricted commercial use is required, it is recommended to use version 0.1.0 of MergeKit, which remains under the Apache 2.0 license, or to contact the licensor for alternative licensing arrangements.