Improving Model Merging with Natural Niches

João P. Abrantes, Robert Tjarko Lange, Yujin Tang
Sakana AI
`{joao,robert,yujintang}@sakana.ai`

Abstract

Model merging is a powerful technique to combine specialized knowledge of multiple machine learning models into a single unified model. However, current methods require manually partitioning the model parameters into a fixed number of groups to be merged, which constraints the exploration of potential combinations and limits performance. To address these limitations, we propose an evolutionary algorithm with three key features: (1) dynamically adjustment of merging boundaries to progressively explore a broader range of parameter combinations; (2) a diversity preservation mechanism inspired by nature, which maintains a population of diverse, high-performing models that are particularly effective for merging; and (3) a heuristic-based *mate selection* strategy to identify the most promising pairs of models for merging. Our experimental results show, for the first time, that model merging can be used to evolve models from *scratch*. Specifically, we evolve MNIST classifiers from scratch using our method, and achieve comparable performance to CMA-ES, while being computationally cheaper. Additionally, we use our method to merge specialised language models and obtain state-of-the-art performance. Our code is available at `https://github.com/AnonScientist/natural_niches`.

1 Introduction

Open-source generative models have allowed the proliferation of thousands of specialised variants, fine-tuned by practitioners to solve their specific needs. In such an environment, where numerous diverse models are freely accessible, the ability to merge and aggregate that wealth of knowledge into a single model becomes important. This process, known as model merging [14], has gained popularity as evidenced by the current prevalence of merged models on the Open LLM Leaderboard [12].

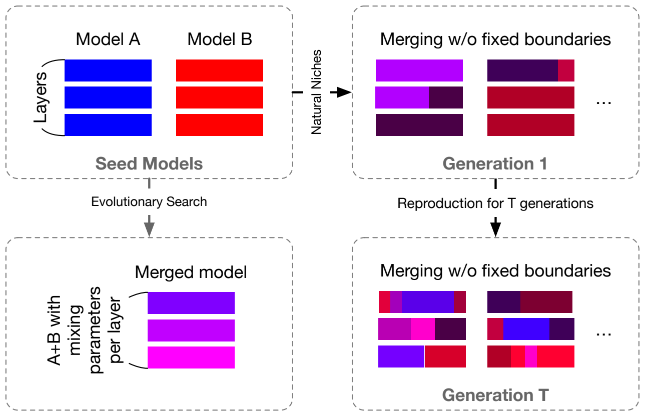
Model A
Model B
Layers
Seed Models
Natural Niches
Merging w/o fixed boundaries
...
Generation 1
Evolutionary Search
Reproduction for T generations
Merged model
A+B with mixing parameters per layer
Merging w/o fixed boundaries
Generation T

Figure 1: Left, previous methods group the parameters of each seed model according to fixed boundaries (e.g., model layers) and then search for a set of coefficients to mix each group. The shades of purple in the layers of the merged model represent how much the interpolation is close to parent A (blue) or parent B (red). Right, evolution of an archive of models using a random split-point explores a progressively larger number of coefficients and boundaries.

Model merging initially relied on manually adjusting coefficients to combine seed models, a process guided by intuition and requiring significant trial and error to optimize performance for specific tasks. Recently, this has been streamlined with evolutionary algorithms that automatically search for optimal coefficients [1] and increase the merging efficiency.

However, one manual step persists: developers must group model parameters into fixed sets before merging, which restricts the search for potential combinations (see Figure 1 left). To overcome this, we propose an evolutionary algorithm with three key features:

1. Evolving the Merging Boundaries. Existing methods partition the parameters of each seed model into fixed groups (e.g., layers) and search for optimal coefficients for merging these groups, which confines exploration to predefined boundaries. In contrast, our approach merges two models at a time, using arbitrary split points to divide parameters. Rather than working with fixed models, we maintain an evolving archive of models. As the number of generations increases, the method progressively explores a broader set of boundaries and coefficients (see Figure 1 right), allowing for increasingly complex combination if needed. This optional and gradual increase in complexity ensures a wider range of possibilities while maintaining computational tractability.

2. Managing Diversity. Merging models only makes sense when they differ, making it essential to maintain diversity within the evolving population. However, the challenge of diversity preservation lies in determining which characteristics should be diverse. While many approaches require developers to manually define a diversity metric, we employ a nature-inspired method to automatically preserve diverse high-performing models that are particularly effective for merging.

3. Matchmaker. We introduce a heuristic for pairing models based on their complementary strengths, enhancing both the efficiency and quality of our method. *Mate selection* remains an under-explored area in genetic algorithms, becoming increasingly important as computational costs of crossovers (merging) rise. This work encourages further research on this topic.

We name our method *Natural Niches* and show that it performs well across two vastly different experiments: 1) evolving small classifiers *from scratch* and from pre-trained models (section 4.1), and 2) merging Large Language Models (LLMs) (section 4.2).

2 Related Work

2.1 Model Merging

Model merging introduces an innovative approach for integrating the strengths of multiple pre-trained models. In contrast to fine-tuning, which focuses on refining a single pre-trained model, model merging can leverage several models concurrently without requiring backpropagation. This has allowed the method to combine extremely large models for tasks involving subjective goals, like customizing an image generation model to reflect personal tastes.

Notably, the release of Stable Diffusion (SD) [21] and open-source interfaces [3] enabled practitioners to merge different SD fine-tunes manually, using techniques like linear and spherical linear interpolation (SLERP) [25]. These early efforts demonstrated the potential of model merging in combining specialized capabilities into a single unified model.

Subsequent research has approached the model merging problem from two complementary directions: minimizing interference between models and automating the merging process. Methods such as TIES [29] and DARE [31] introduced strategies to balance the contributions of individual models while minimizing interference, ensuring that the strengths of each model are preserved without mutual disruption.

Evolutionary algorithms like CMA-ES [11] were later applied to automate the search for optimal merging coefficients. As explored in [1], these methods not only automate what was previously a manual, iterative process but also significantly improve merging efficiency.

While previous research centered on merging pre-trained models, we show that merging can efficiently be used to evolve models from scratch. Additionally, unlike earlier methods that required manual partitioning of model parameters, we automate and optimize this process during the evolutionary process.

2.2 Overview of Diversity Preservation in Genetic Algorithms

Diversity preservation in Genetic Algorithms (GA) is crucial for finding multiple solutions to multimodal problems [26] and to prevent premature convergence. We believe this is particularly important when using crossover operations (such as model merging). These operations benefit from diversity while at the same time reducing it, which may lead to premature convergence if not counter-acted by a diversity increasing mechanism. In this section, we provide a quick overview of the two main methods for diversity preservation in GA: 1) crowding [7, 27] and 2) fitness sharing [10, 8, 9, 20].

Crowding methods involve first applying mutation and crossover to produce new candidate solutions. These candidates then compete for inclusion in the population, but only against other candidates that are similar, based on a predefined criteria such as genetic or phenotypic distances. This selective competition helps maintain diversity within the population by preventing any single solution type from becoming overly dominant. A similar mechanism for selective competition is used in the popular algorithm of MAP-Elites [18]. In MAP-Elites, the solution space is divided into a multidimensional grid, with each cell representing a species defined by one or more predefined behaviour descriptors. New candidates are placed into cells based on their descriptors and replace existing solutions only if they perform better. The real challenge of this method lies in defining descriptors that promote the desired type of diversity.

Fitness sharing requires each individual to share its rewards with others. In *explicit* fitness sharing, the researcher defines a distance function that is used to cluster similar individuals into a species, each individual then shares its fitness with other members of its species, making it more difficult for any single species to grow excessively large. A notable example is the NEAT [24] algorithm, known for evolving neural networks topologies, which clusters solutions into species by measuring genotypic differences (distances in network topologies). *Implicit* fitness sharing [23, 6], is seen as the more natural method because, as in Nature, it protects niches rather than species. A niche is a group of individuals that compete for the same resources, while a species is defined as group that can interbreed and typically have small genetic and phenotypic differences. Usually, members of the same species compete for the same resources (e.g., food, partners, shelter), but vastly different species can also compete for the same vital resources like nesting sites or food sources (e.g., birds and bats, lions and hyenas). *Implicit* sharing does not rely on custom distance metrics. Instead, it simulates natural competition for limited resources, promoting diversity as individuals who can derive their fitness from less contested resources are favoured. We provide more details on Section 3.

3 Natural Niches

In model merging, the goal is to find the optimal parameters θ^* for a merged model from a set of K seed models, each of which characterized by its model parameters θ_i ($i = 1 \cdots K$), such that optimization goal, normally in the form of task scores summation or average, is maximized. The following equation expresses this description mathematically:

$$\theta^* = \arg\max_{\theta} \sum_{j=1}^{N} s(x_j \mid \theta), \text{where}, \theta = h_w(\theta_1, \cdots, \theta_K) \tag{1}$$

where h_w is the model merging function parameterized by w's that correspond to fixed model merging boundaries (e.g., one scalar $w_{k,l}$ for the l-th layer in the k-th seed model), s is the score function for a certain task, x_j is a task example, and N is the number of examples to be evaluated. In Natural Niches (NN), we propose modifications to the merging function h to enable evolution of the merging boundaries, and adjustments to the optimization goal to promote diverse solutions.

3.1 Eliminating Fixed Model Merging Boundaries

In the formulation above, finding θ^* boils down to searching for the optimal model merging parameters w in h_w. To get rid of the constraints of fixed model merging boundaries and thus allows more flexibility, we propose to include these boundaries together with the mixing parameters into the evolutionary process. Concretely, NN maintains an archive of models, which is initialized with the K seed models. At each training step, NN randomly picks two models A and B from the archive, and samples two parameters (w_m, w_s) that determines the mixing ratio and the split-point in the models' parameters space. It then merges models A and B with the following formula, and inserts the new model into the archive if it outperforms the worst individual.

$$h_{\text{NN}}(\theta_A, \theta_B, w_m, w_s) = \text{concat}\big(f_{w_m}(\theta_A^{<w_s}, \theta_B^{<w_s}), f_{1-w_m}(\theta_A^{\geq w_s}, \theta_B^{\geq w_s})\big) \tag{2}$$

Here, $\theta^{<w_s}$ and $\theta^{\geq w_s}$ indicate the sub-arrays of model parameters before and after the split-point indexed by w_s. $f_t(\theta_A, \theta_B)$ is a spherical linear interpolation of rotations (SLERP) function that interpolates (θ_A, θ_B) with t. As shown in the right part of Figure 1, our method incrementally expands the search space by exploring a broader set of boundaries and coefficients. This gradual introduction of complexity ensures a wider range of possibilities while maintaining computational tractability.

3.2 Encouraging Diversity via a Modified Optimization Goal

Competing for limited resources **naturally** promotes diversity, favoring individuals who can tap into less contested resources. In the context of the optimization goal in Equation 1, where a sum of scores from all the examples is being maximised, each score is a "resource" that contributes to the fitness of a solution. By limiting the resource supply, NN sparks competition which naturally favors individuals that take over new niches. Concretely, we limit the total fitness a population can extract from a data point x_j by a capacity c_j. The amount of fitness a candidate solution derives from a data point is proportional to its score relative to the aggregate score of the population. Our modified goal becomes:

$$\theta^* = \arg\max_{\theta} \sum_{j=1}^{N} \frac{s(x_j \mid \theta)}{z_j + \epsilon} c_j, \text{where, } z_j = \sum_{k=1}^{P} s(x_j \mid \theta_k) \tag{3}$$

where ϵ in the denominator is a small number to prevent the zero-division error. In the term that defines z_j, P is the archive size. The capacity c_j, is task dependent and can be defined in multiple ways. For example, in binary scoring tasks (i.e. $s(\cdot) \in \{0, 1\}$) we simply set $c_j = 1$. In one experiment (WebShop) the environment offers a continuous reward from 0 to 1. Here, we define $c_j = \max_i s(x_j|\theta_i)$ to ensure that partially solved data points (where $\max_i s(x_j|\theta_i) < 1$) do not distribute the same amount of fitness points as fully solved data points (where $\max_i s(x_j|\theta_i) = 1$).

3.3 Sampling Parents via Matchmaker

Many evolutionary algorithms use the crossover operation to combine the strengths of both parents. In biology, this combination (i.e., reproduction) is very expensive, and therefore, animals invest many resources in the process of mate selection. We believe that as we make use of more expensive crossover operations, like model merging, algorithms for mate selection become increasingly important.

In contrast to conventional methods that put more sampling probability mass on top performing models in the archive, NN adds an extra layer of consideration that takes into account the complementarity of the parent models. Specifically, we sample the first parent based on their weighted sum of scores defined in Equation 3, and then sample the second parent based on a "matching score" generated by function g that is specifically tailored for the first parent. The equation below gives the definition of this matching score, it straightforwardly expresses a desire to choose a model B that performs well in the data points where model A performs less well, while giving an extra preference to resources with high capacity c_j and low competition z_j.

$$g(\theta_A, \theta_B) = \sum_{j=1}^{N} \frac{c_j}{z_j + \epsilon} \max\big(s(x_j \mid \theta_B) - s(x_j \mid \theta_A), 0\big) \tag{4}$$

4 Experiments

We verify the effectiveness of our proposed method on two challenging tasks: First, we evolve image classifiers from scratch and from pre-trained models on the MNIST dataset, and then we scale up the experiment to merging LLMs.

4.1 Experiment 1: Evolving MNIST classifiers

4.1.1 Setup

Model. The model being optimized is a two-layer feedforward neural network with 19,210 parameters in total. When starting from scratch, we randomly initialize the models. For pre-trained models, we develop two specialized models: one is trained on digits 0 through 4, and the other is trained on digits 5 through 9.

Dataset. We've used the MNIST [15] dataset from `scikit-learn` [19] where each digit is a 8x8 gray scaled image. 80% of the data was used as the training split, and 20% as the testing split.

Baselines. For the MAP-Elites algorithm, we used two diversity dimensions to create a 10 by 10 grid: the accuracy of the model in odd and even numbers. When starting from scratch, we use CMA-ES [11] as a baseline, even though it does not perform model merging here. Since the models are randomly initialized, optimizing mixing coefficients alone would be insufficient. Instead, CMA-ES directly optimizes model weights, which incurs a cubic computational cost $O(n^3)$ with respect to the number of parameters. While this method doesn't scale to larger models, it serves as a benchmark for how a popular evolutionary algorithm performs in this experiment. When working with pre-trained models, we use a brute-force search baseline that merges the two seed models by adjusting a mixing coefficient that ranges from 0 to 1 in increments of 0.00001. This baseline is first evaluated on the training data, and the best coefficient is subsequently evaluated on the test data.

Evolutionary Operators and Variables. All model merging methods (which excludes CMA-ES) sample a new candidate at a time and decide sequentially whether to insert the candidate into the archive. NN and GA use an archive of 20, sampling each candidate sequentially and deciding whether to insert it, similar to MAP-Elites. MAP-Elites, uses a 10x10 grid, resulting in an archive size of 100. CMA-ES uses a population of 20, sampling and updating its parameters in batches. When starting from scratch all model merging methods use the same mutation operation (Gaussian noise) and the same crossover operation (SLERP with split-point, as described in section 3.1). However, when dealing with pre-trained models, mutation is omitted because random alterations don't scale well to larger models. By avoiding mutation, we can assess which method is most promising for achieving efficient merging in larger pre-trained models.

Compute Resources. The 10 independent runs took about 15 hours for CMA-ES, and about 1h for each of the other methods. We ran this experiment using only CPUs.

4.1.2 Results

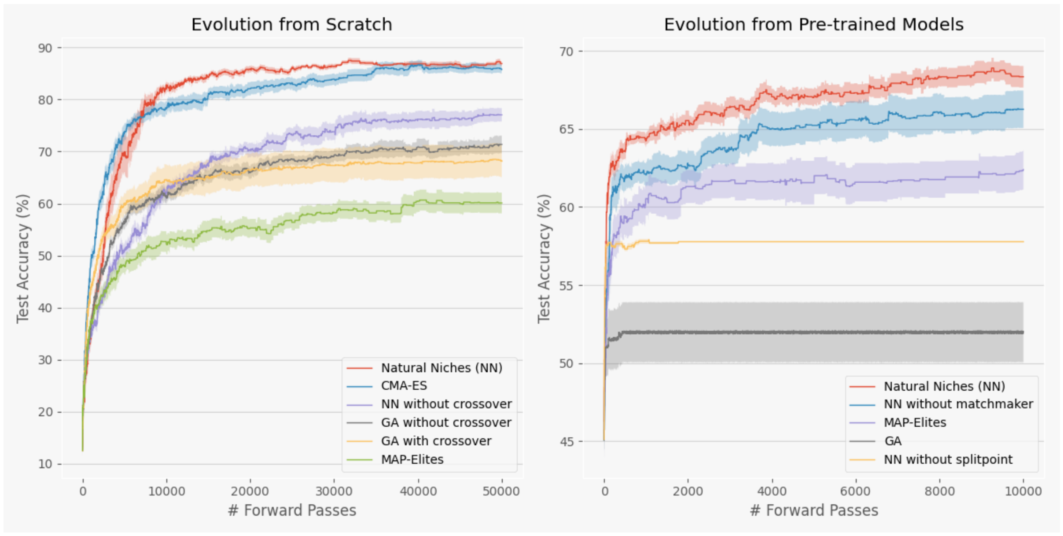

Figure 2: The plots show the accuracy on the test split vs the number of forward passes when starting from randomly initialised models (left) and when starting from the pre-trained models (right). The solid-lines represent the mean of ten independent runs and the shaded area around represents one standard error deviation.

When starting from scratch, NN achieves the highest test accuracy by a substantial margin when compared to the other model merging methods, as shown in Figure 2 (left). Interestingly, GA with crossover performs better early on (before step 12,000) than GA without crossover, however, it converges faster to an inferior solution. The early convergence happens because GA can't maintain a diverse population which is crucial for effective crossover operations. Crossover reduces population diversity, and without a strong counteracting force, it diminishes exploration. In contrast, NN leverages the crossover operation effectively, benefiting significantly from the diversity it manages to retain. GA is an extreme case where there is no competition, we observed that by progressively decreasing competition in NN, we progressively converge earlier to worse solutions (section 4.1.3). MAP-Elites clusters individuals by their accuracy on odd and even numbers. This means it will always keep individuals who perform poorly on those tasks because there is a slot reserved just for them. Even though those individuals add to the diversity of the population, this is clearly not the type of diversity that leads to strong solutions and it highlights the difficulty of hand-engineering useful diversity metrics.

For models trained from scratch, the split-point and matchmaker have a minimal impact (ablations omitted for clarity). However, as seen in Figure 2 (right), the split-point becomes crucial when starting from pre-trained models, while the matchmaker significantly improves performance throughout the training process. GA has a low average test accuracy with large error bars as its performances is highly dependent on the quality of the first merges. Note that when starting from pre-trained models the mutation operator was not used (as explained in Section 4.1.1), and therefore, the performance is worse.

4.1.3 Analysing Diversity and Competition

This section focuses exclusively on the experiment where models were evolved from scratch, as the later sections will provide ample discussion on evolving models starting from pre-trained LLMs.

Diversity. Figure 3 left, shows the percentage of training data points that can be correctly labeled by at least one model in the archive, we call this percentage the training coverage. We observe that the archive in NN quickly spreads to cover the majority of the training data points and maintains this high coverage throughout the training process.

The right-hand plot shows how the diversity in the performance of the population evolves with training. If either all models correctly or incorrectly classify a data point, the entropy is 0 (no diversity). In contrast, when the models are evenly split on a prediction, entropy reaches its maximum value of 1. The plot displays the average entropy across all data points. For NN we see a sharp initial rise in entropy followed by a gradual decline as low-performing models go extinct. In contrast, MAP-Elites continually increases diversity by retaining lower-performing models, but it fails to achieve a high coverage. The Genetic Algorithms, lacking a diversity preservation mechanism, reduce coverage early on and show a sharp drop in entropy as they converge prematurely on the best solutions.

Overall, the graphs show that NN maintains an archive of models with complementary strengths that facilitate effective merging, while systematically discarding weaker models as training progresses.

Competition. Figure 4 left, shows that smaller archives perform better in the beginning but converge faster to inferior solutions. This suggests that we should scale the archive size along the number of forward passes we want to make. Note that in our plot the computational cost does not increase with the archive size since the number of forward passes remains the same, however, the memory footprint does increase with larger populations. For very large models we can always store the archive on disk instead of keeping them all in the RAM.

For a fixed population size P, we can adjust the intensity of competition by introducing a hyper-parameter $\alpha \geq 0$, as described in the fitness function in eq. 5.

$$f(\theta_i) = \sum_{j=1}^{N} \frac{s(x_j|\theta_i)}{{z_j}^{\alpha} + \epsilon} c_j \tag{5}$$

When $\alpha = 0$, there is no competition because the total fitness available per data point becomes unlimited. When $\alpha = 1$, the total fitness distributed among different individuals is limited to the capacity c_j. For $\alpha > 1$, the total fitness distributed decreases with increasing competition (z_j), this scenario can be thought of as individuals needing to "fight" for resources, spending some fitness points in the process. Figure 4 right, shows that smaller values of α (i.e. lower competition) have a similar effect to decreasing the population size: it performs better in the beginning but it converges faster to inferior solutions.

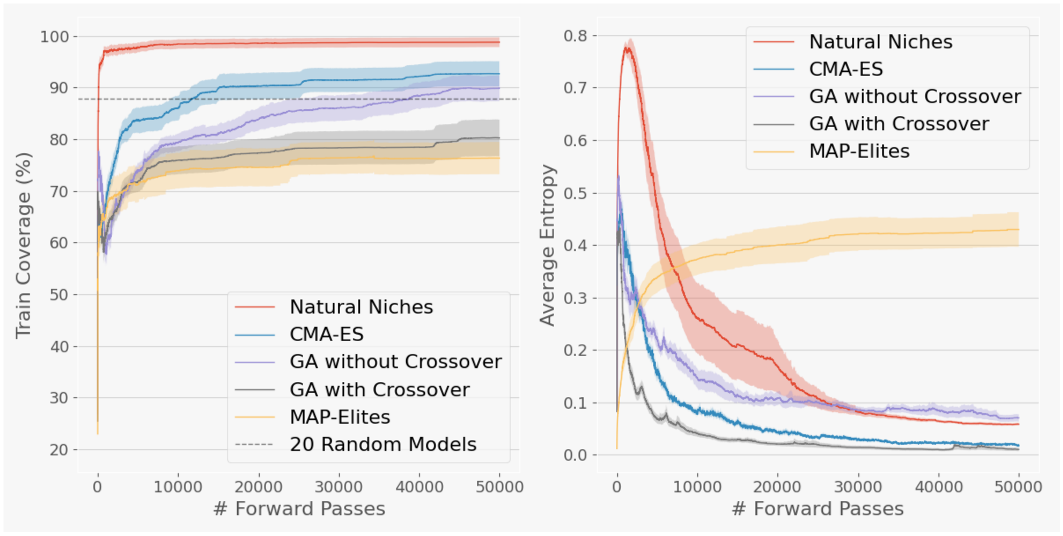

Figure 3: Left: The percentage of training data points that can be correctly labeled by at least one model in the population. Since there are 10 possible labels, 20 random models obtain an average coverage of $1-(\frac{9}{10})^{20}=87.8\%$. Right: The evolution of diversity in the population's performance, measured by entropy, over the course of training.

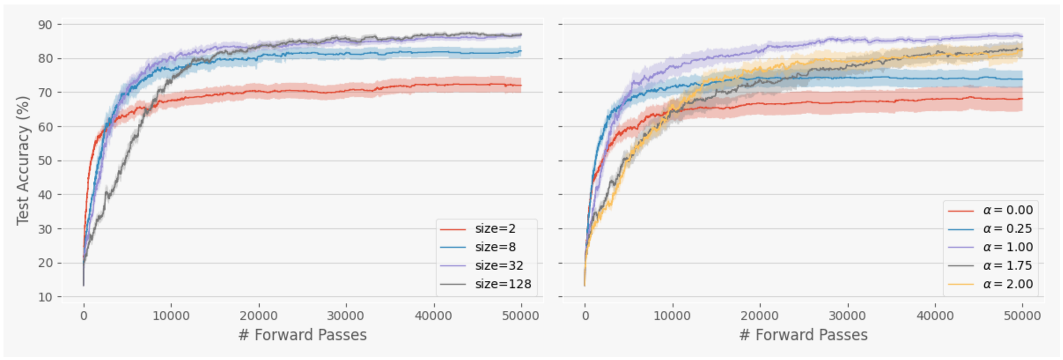

Figure 4: Left, test accuracy of Natural Niches on the MNIST across different archive sizes. Right, test accuracy for different α values while population size is 20.

4.2 Experiment 2: LLM Merging

In our LLM Merging experiments we don't use a mutation operator since random mutations don't work well on large models. Moreover, in these experiments we initialise the NN and GA archives with seed models, followed by a short warm-up period (50 iterations or less) where the seed models merge randomly amongst themselves and populate the archive. For these experiments, we used 4 H100 GPUs to run each method for around 24h.

4.2.1 Setup: Japanese Math LLM

Models. We reproduce the experiment done in [1], where one Japanese specialist LLM, `shisa-gamma-7b-v1` [2], and two Math specialists LLMs, `WizardMath-7B-V1.1` [16] and `Abel-7B-002` [4] are combined to create a hybrid model that can answer math problems in Japanese.

We used exactly the same datasets and evaluation method from [1], but for completeness, we provide the details here.

Dataset. The test split, consists of the Japanese test set of the MGSM dataset [22], which is a Japanese translation of a subset of the test set of the GSM8k dataset [5] consisting of 250 samples.

The training set, consists of a translation of the remaining 1069 (out of 1319 examples) of the GSM8k test set that were not included in the MGSM Japanese test set.

Baselines. The CMA-ES baseline implemented in [1] optimised the parameters for a TIES-Merging [29] with DARE [31] between the three seed models. CMA-ES used a population of 9 and it made 9,000 evaluations, while here we run NN and GA with an archive size of 20 but limited them to 4,000 evaluations.

Evaluation. A correct answer must meet the following criteria: 1) the final numerical value must be correct, and 2) the reasoning text must be written in Japanese. To determine the language of the output, the library `fasttext` was used [17, 13].

4.2.2 Setup: Combining Math and Agentic Skills

Models. We combine a math specialist, `WizardMath-7B-V1.0` [16], with a specialist on agentic enviroments, `AgentEvol-7B` [28], to achieve an agent that performs well on the math benchmark GSM8k [5] and on the web shopping benchmark WebShop [30].

Datasets. For the math task, we used the test split of GSM8k as our test split (1319 samples). For the training split, we used the first 1319 samples of GSM8k train dataset. In the web shopping task, we used the WebShop environment implemented in [28]. The test split consistent of the first 100 tasks, while the training split were the next 100 tasks. We allowed the agents to take up to 7 steps.

Baselines. The CMA-ES optimised 32 mixing coefficients (one for each layer) for a SLERP merge between the two seed models. All methods were run for a 1000 evaluations on the training set. For the MAP-Elites we used two dimension to create a 4 by 4 grid: the accuracy on the math and on the web shopping training splits.

Evaluation. In this experiment, all methods used 1000 evaluations on the training set. NN and GA used an archive size of 15. CMA-ES used a population size of 25.

4.2.3 Results.

Tables 1 and 2 show that NN achieves the highest score. Both the matchmaker and the split-point techniques play a crucial role, however, the split-point seems to be slightly more important. Note that on Table 1 CMA-ES was run 2.25x longer than the other methods and used a more advanced merging technique (DARE-TIES), while on Table 2 all algorithms were run for the same amount of time and used the same merging method (SLERP). When combining the Math and Agentic skills, CMA-ES yielded a low score, likely due to suboptimal parameter partitioning, highlighting the need to include the merging boundaries in the optimization process.

Table 1: Accuracy of various methods on the Japanese Math benchmark MGSM-JA.

Methods	**MGSM-JA (acc ↑)**
Shisa Gamma 7B v1	9.6
WizardMath 7B v1.1	18.4
Abel 7B 002	30.0
Natural Niches (NN)	**54.4**
NN w/o matchmaker	47.2
NN w/o split-point	44.4
CMA-ES (DARE-TIES)	52.0
GA	44.8
LLama 2 70B	18.0
Japanese StableLM	17.2
GPT-3.5	50.4
GPT-4	78.8

Table 2: Scores of various methods on math (GSM8k) and web shopping (WebShop) benchmarks.

Methods	**GSM8k (acc ↑)**	**WebShop (score ↑)**	**Average (score ↑)**
WizardMath 7B v1.0	74.22	0.00	37.11
AgentEvol 7B	6.29	88.88	47.59
Natural Niches (NN)	40.74	86.17	**63.46**
NN w/o matchmaker	39.53	83.99	61.76
NN w/o splitpoint	33.31	87.91	60.61
GA	36.81	88.23	62.02
MAP-Elites	37.33	84.23	60.78
CMA-ES (SLERP)	46.21	43.49	44.85

4.2.4 Analysis

As shown in Figure 5, the findings from the MNIST dataset generalize to LLM merging. The Natural Niches method maintains high training coverage, as seen on the left side of the figure. The entropy rises early on as the models explore diverse niches (right), followed by a gradual decrease as low-performing models are removed, and the strengths of the models are aggregated. In contrast, MAP-Elites focuses on maximizing entropy at the cost of training efficiency and coverage, as it retains low-performing models. GA quickly reduces both coverage and entropy as it greedily converges on its top solution, ultimately collapsing the entire archive onto a single solution, with entropy nearing zero.

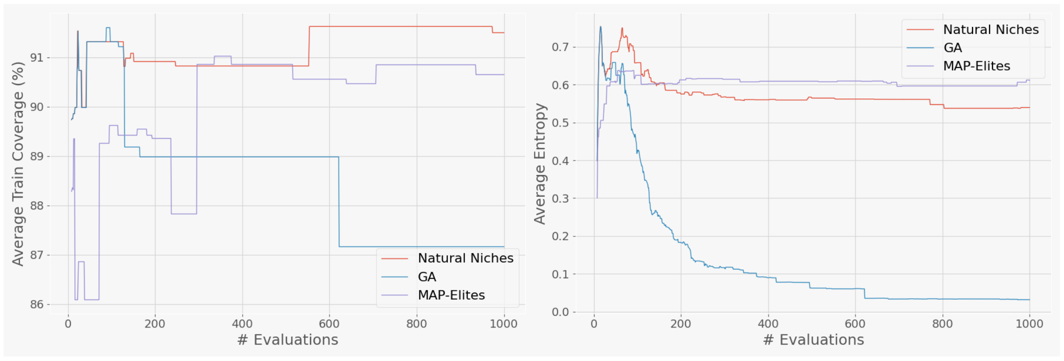

Figure 5: Left, the percentage of training data points that can be correctly labeled by at least one model in the population, averaged on the Math and Web shopping datasets. The right plot shows how the diversity in the performance of the population evolves with training.

5 Limitations & Future Work

The feasibility of model merging strongly depends on the degree of similarity between models. As demonstrated in [31], when fine-tuned models deviate significantly from their base models—often due to extensive, divergent training—merging becomes impractical. We hypothesize that models with divergent *state representations* are incompatible for merging. However, a standardized metric for model *compatibility* has yet to be established. Defining such a metric could allow it to be used as a form of regularization during preprocessing (e.g., fine-tuning), enabling better control over model compatibility and ensuring the success of merging.

Moreover, we believe there is a strong evolutionary pressure for models that are co-evolving together to remain compatible for merging. Should one model, diverge and become incompatible with others, it would no longer produce viable offspring, halting its improvement and leading to its eventual extinction. Testing this hypothesis through further research would provide valuable insights into the dynamics of model co-evolution.

Finally, incorporating a *compatibility* metric into the matchmaker heuristic could facilitate the co-evolution of distinct *species* of models, defined as groups that can merge with one another but not with others.

6 Conclusion

We've shown that model merging can significantly speed up the evolution of image classifiers when combined with a diversity-preservation technique. Moreover, this technique scales up to pre-trained LLMs. Our ablation studies show that both our proposed matchmaker heuristic and the use of crossover with split-point significantly improve the performance of the proposed method and have the potential to improve other evolutionary algorithms that use the crossover operation.

References

[1] Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*, 2024.

[2] augmxnt. Shisa-gamma-7b. `https://hf.co/augmxnt/shisa-gamma-7b-v1`, 2023.

[3] AUTOMATIC1111. Stable diffusion webui, 2022.

[4] Ethan Chern, Haoyang Zou, Xuefeng Li, Jiewen Hu, Kehua Feng, Junlong Li, and Pengfei Liu. Generative ai for math: Abel, 2023.

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[6] Paul Darwen and Xin Yao. Every niching method has its niche: Fitness sharing and implicit sharing compared. In *Parallel Problem Solving from Nature—PPSN IV: International Conference on Evolutionary Computation—The 4th International Conference on Parallel Problem Solving from Nature Berlin, Germany, September 22–26, 1996 Proceedings 4*, pages 398–407. Springer, 1996.

[7] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan, 1975.

[8] Kalyanmoy Deb and David E Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third international conference on Genetic algorithms*, pages 42–50, 1989.

[9] David E Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In *PPSN*, volume 2, 1992.

[10] David E Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, volume 4149, pages 414–425. Cambridge, MA, 1987.

[11] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[12] HuggingFace. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023. HuggingFace.

[13] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[14] Maxime Labonne. Merge large language models with mergekit. Hugging Face Blog, 2024.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[17] Yuval Marton, Ning Wu, and Lisa Hellerstein. On compression-based text classification. In *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27*, pages 300–314. Springer, 2005.

[18] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[20] Alain Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE international conference on evolutionary computation*, pages 798–803. IEEE, 1996.

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[22] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

[23] Robert E Smith, Stephanie Forrest, and Alan S Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary computation*, 1(2):127–149, 1993.

[24] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[25] Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

[26] Ka-Chun Wong. Evolutionary multimodal optimization: A short survey. *arXiv preprint arXiv:1508.00457*, 2015.

[27] Ka-Chun Wong, Chun-Ho Wu, Ricky KP Mok, Chengbin Peng, and Zhaolei Zhang. Evolutionary multimodal optimization using the principle of locality. *Information Sciences*, 194:138–170, 2012.

[28] Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

[29] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models, 2023.

[30] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

[31] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have evolved MNIST classifiers from scratch and achieved comparable performance to CMA-ES. Additionally, we applied our method to LLM merging and achieved state-of-the-art performance.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a section that addresses the limitations of this work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include code to reproduce the MNIST experiment which is the main experiment in this paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The link to our code repo is on the abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe that on our Setup sections and it can also be verified by inspecting our code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: On the MNIST experiment we did 10 independent runs for each method and report the error bars. On LLM merging, each run is very computationally expensive so we only did one run for each method and do not report error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In our Setup subsections we specify the compute resources used.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We present a generic algorithm to improve evolutionary algorithms that use model merging or crossover operations.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.