Agent Skill Acquisition for Large Language Models via CycleQD

So Kuroki, Taishi Nakamura, Takuya Akiba, Yujin Tang
Sakana AI, Japan
`{sokuroki,taishi,takiba,yujintang}@sakana.ai`

Abstract

Training Large Language Models (LLMs) to acquire various skills remains a challenging endeavor. Conventional training approaches often struggle with data distribution imbalances and inadequacies in objective functions that do not align well with task-specific performance. To address these challenges, we introduce CycleQD, a novel approach that leverages the Quality Diversity (QD) framework through a cyclic adaptation of the MAP-Elites algorithm. In this framework, each task's performance metric is alternated as the quality measure while the others serve as the behavioral characteristics. This cyclic focus on individual tasks allows for concentrated effort on one task at a time, eliminating the need for data ratio tuning and simplifying the design of the objective function. Empirical results indicate that applying CycleQD to 8-billion parameter models not only enables them to surpass traditional fine-tuning methods in coding, operating systems, and database tasks, but also achieves performance on par with GPT-3.5-TURBO across these domains. Our code is available at `https://github.com/SakanaAI/CycleQD`.

1 Introduction

Large Language Models (LLMs) have established themselves as powerful tools in the machine learning and artificial intelligence landscape, initially gaining prominence through their effectiveness in conversational tasks [Achiam et al., 2023]. As these models continue to evolve, there is an increasing interest in enhancing their abilities to integrate linguistic understanding with actionable outputs, leading to the production of intelligent LLM-based agents. Consequently, the training of LLMs to acquire various agent skills is attracting growing research attention [Gur et al., 2023, Liu et al., 2023b, Xi et al., 2023, 2024, Wang et al., 2024b].

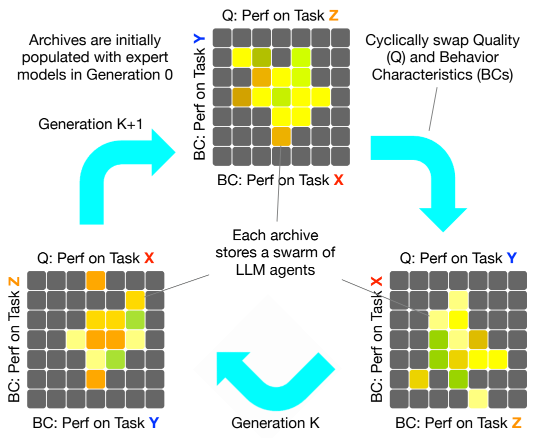

Figure 1: **Method Overview.** CycleQD uses QD in a cyclic manner to merge LLM agents. We initiate the archives with expert LLMs, each of which has been fine-tuned to excel in a specific task. Model merging is then conducted using QD, treating task performance as both quality (Q) and behavior characteristics (BC), which are cyclically swapped in each generation during the QD process.

Despite the potential, training LLMs to acquire various agent skills presents complex challenges. Beyond the basic hurdle of data collection, a significant difficulty arises from the need to balance data ratios during training—ensuring the model learns effectively from datasets representing different skills without favoring any particular one and without neglecting others. In addition, conventional objective functions, such as next token prediction, often fail to capture the nuances required for performance across diverse tasks, typically leading to sub-optimal training outcomes. This function, although effective for general language understanding, does not foster the development of specific agent skills. An alternative, such as reinforcement learning (RL), has been shown to align LLMs with user intents more effectively. Ouyang et al. [2022] demonstrated aligning LLMs with user intents by RL from human feedback, where a reward model is trained from a dataset of human-labelled model output rankings. Unlike general language tasks, the "rewards" are clearly defined in most agent tasks. On the other hand, managing the balance of rewards in learning tasks and the associated costs of fine-tuning pose additional challenges.

In response to these challenges, we propose a novel framework that leverages the Quality Diversity (QD) method [Pugh et al., 2016], specifically through a cyclic adaptation of the MAP-Elites algorithm [Mouret and Clune, 2015]. Our approach features a dynamic cycle where each agent skill's performance metric is optimized in isolation, with other skills represented as behavioral characteristics (BCs). This strategy not only simplifies the complex data ratio management by focusing on one task at a time but also addresses the inadequacy of traditional loss functions by directly optimizing task-specific performance metrics. Moreover, we utilize evolutionary model merging [Akiba et al., 2024] and a singular value decomposition (SVD)-based mutation method, which are essential for transferring skills from specialized experts to a new, cohesive model. The QD framework proves ideal for model merging, particularly in LLMs where the computational pipelines are long. By allowing locally sub-optimal solutions to persist, such as a temporarily under-performing layer, QD saves these configuration in its archive and provides them a chance to improve in the future. This method has shown that once these layers are effectively merged, the overall model performance can significantly increase, a phenomenon observed during our experiments. This technique serves as an alternative method to conventional fine-tuning, allowing for the preservation and enhancement of multiple agent skills without the common pitfalls of catastrophic forgetting typically seen in standard training processes. See Figure 1 for an overview of CycleQD. See Appendix A.1 for related works.

2 Methods

CycleQD builds on MAP-Elites, a recognized implementation of QD. It distinguishes itself from traditional MAP-Elites systems in three key ways (see Algorithm 1 for the pseudo-code):

1. **Alternating Quality and BCs:** Pugh et al. [2016] emphasizes the importance of well-aligned BCs with the quality (i.e., task metric that needs to be optimized). Traditional systems typically set the design at the onset and do not alter them throughout the QD process. In contrast, CycleQD uses task metrics as both quality and BCs, ensuring alignment and dynamically alternating them during the QD process (lines 6-7 in Algorithm 1 and detailed in Appendix A.2.1).
2. **Model Merging as Crossover:** Unlike existing systems that optimize model parameters directly, CycleQD leverages a model merging algorithm as a crossover operation, replacing the heuristic, hand-designed rules often seen in practice (line 12 in Algorithm 1 and detailed in Appendix A.2.2). In CycleQD, we fine-tune expert agents, each of which only needs to specialize in one task, and use them as the seed models to initialize the model archives (lines 2-4 in Algorithm 1).
3. **SVD-based Mutation:** Traditional mutation operations in genetic algorithms like MAP-Elites typically introduce random perturbations from a pre-defined distribution to explore new regions. In contrast, CycleQD utilizes perturbations aligned with the principal components of the models' parameters matrices, facilitating exploration outside the convex regions formed by the parent models while avoiding overfitting (line 13 in Algorithm 1 and detailed in Appendix A.2.3).

After CycleQD, we aggregated elite models from the archives of K target tasks into a model using softmax-weighted task vectors (detailed in Appendix A.2.4).

3 Experiments

3.1 Evaluation on Computer Science Tasks

3.1.1 Task Description

In this experiment, our goal is to train LLMs to master three agent skills in computer science: coding, Operation System (OS) manipulation and Database (DB) query generation. We adopt the MBPP+

(Mostly Basic Python Programming) dataset from EvalPlus [Liu et al., 2023a] for coding skill development, and utilize the OS and DB datasets from AgentBench [Liu et al., 2024] for training. The MBPP+ dataset, based on the original MBPP, uses a subset of 399 hand-verified problems from MBPP-sanitized to ensure well-formed programming tasks. The OS dataset evaluates LLMs in genuine interactive bash environments (Ubuntu Docker). The DB dataset assesses LLMs' abilities to operate on real databases via SQL with authentic SQL interfaces and multiple tables.

3.1.2 CycleQD Setups

Experts: We employ LLAMA-3-8B-INSTRUCT MODEL [Dubey et al., 2024] as our base model, and use supervised fine-tuning to create LLM experts in coding, OS and DB. For the OS and DB experts, we use their training datasets from the Agent-FLAN Chen et al. [2024]. The coding expert is fine-tuned on a combination of the MAGICODER-EVOL-INSTRUCT-110K and MAGICODER-OSS-INSTRUCT-75K datasets [Wei et al., 2024]. See Appendix A.3.1 for training parameters.

Datasets: We use the MBPP+ dataset, as well as the test and development datasets from OS and DB. To ensure that the experts have the same task metrics across tasks, each dataset is split evenly into training and test splits. For OS, problems that could not be solved by either the expert model, GPT-4, or GPT-3.5 are excluded beforehand to reduce computation cost.

See Appendix A.3.1 for CycleQD hyper-parameters and Appendix A.3.2 for baselines.

3.1.3 Results

Overall: We summarize our results from this experiment in Table 1. In addition to the baselines, we also include GPT-3.5-TURBO and the base model for references (highlighted with a lightgray background). The first thing to notice is that each expert model (models 2-4), fine-tuned specifically for its assigned task, performs better than the LLAMA3-8B-INSTRUCT base model (model 1), laying the stepping stones for CycleQD and other model merging based methods. After evolutionary computing, our method, averaged across the three tasks, outperforms all baselines, and is approaching the performance of GPT-3.5-TURBO. Specifically, CycleQD achieves the highest scores on the coding and the OS tasks among the baselines, and has only a mild performance drop on the DB tasks compared with the expert. Figure 2 gives the archives in CycleQD at the end of the experiment.

Comparison with Fine-tune Based Methods: A notable result from the table is that, the model fine-tuned on all the datasets (model 5) is only partially and marginally better than the three experts, despite being trained on a larger dataset. On the other hand, the expert models can be easily trained because they do not require the data ratio or object function adjustment. The drawback of these

Algorithm 1 CycleQD

```
 1: function TRAIN($A$, EXPERTS)          ▷ $A$ are the archives, EXPERTS are the expert LLM agents
 2:     for $i \leftarrow 1$ to $K$ do                                    ▷ $K$ is the number of tasks
 3:         $A[i] \leftarrow$ UPDATEARCHIVE($A[i]$, EXPERTS)     ▷ Place experts properly in $i$-th archive
 4:     end for
 5:     for $t \leftarrow 1$ to $N$ do                          ▷ $N$ is the total number of generations
 6:         $i \leftarrow t \mod K$              ▷ Cyclically swaps the task (archive) for the next QD step
 7:         $A \leftarrow$ QDSTEP($A, i$)                          ▷ Conduct one step of the QD algorithm
 8:     end for
 9: end function

10: function QDSTEP($A$, $k$)                       ▷ $A_k$ is the archive corresponding to the $k$-th task
11:     $p_1, p_2 \leftarrow$ SAMPLE($A[k]$)     ▷ $p_1$ and $p_2$ are the parents, SAMPLE() is detailed in Sec A.2.1
12:     $c \leftarrow$ CROSSOVER($p_1, p_2$)         ▷ $c$ is the child, CROSSOVER() is detailed in Sec A.2.2
13:     $c \leftarrow$ MUTATE($c$)                                  ▷ MUTATE() is detailed in Sec A.2.3
14:     for $i \leftarrow 1$ to $K$ do
15:         $A[i] \leftarrow$ UPDATEARCHIVE($A[i], c$)               ▷ Place $c$ properly in $i$-th archive
16:     end for
17: end function
```

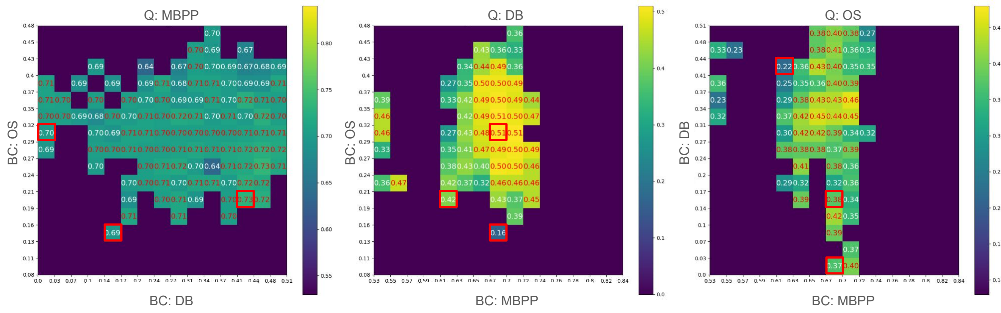

Figure 2: **CycleQD Archives from Computer Science Tasks.** In each archive, the two axes show the BCs, and the color intensities indicate the quality of the LLM agent in that grid. These archives are obtained after 1200 generations of CycleQD. The red bounding boxes indicate the grids where expert policies were present. See Appendix A.3.3 for archive development across generations.

Table 1: **Performance on computer science tasks.** We report pass@1 for MBPP and success rates for the DB and OS tasks. Performance of the base model and GPT-3.5 are highlighted for reference.

#	Methods	MBPP	DB	OS	Avg
0	GPT-3.5-TURBO	82.5	41.6	38.5	53.9
1	Llama3-8B-Instruct (base model)	67.3	5.3	25.2	32.6
	Fine-tuning Based Methods				
2	Fine-tuning (Coding expert)	70.4	21.2	20.7	37.4
3	Fine-tuning (DB expert)	65.8	**42.4**	28.5	45.6
4	Fine-tuning (OS expert)	66.3	0.0	30.4	32.2
5	Fine-tuning (All)	67.3	37.1	36.7	47.0
	Merging Based Methods				
6	Merging (w/o learning)	72.9	24.7	**42.6**	46.7
7	Merging (learning w/ GD)	69.3	41.2	29.6	46.7
8	Merging (learning w/ CMA-ES)	69.3	41.2	30.2	46.9
9	CycleQD (Ours)	**76.4**	38.2	**42.6**	**52.4**

experts is that they specialize in only one task, prohibiting their usage in wider scenarios. This result proves the difficulty with conventional methods, underscoring the contributions of CycleQD.

Comparison with Merging Based Methods: Conventional model merging based methods, learning-based or not, do not give results better than ours either. This is likely due to two reasons. The first is the lack of a mutation operation that helps the merged model break out the "convex region" formed by the seed models. For instance, model 8 shows the best average performance within the same category. However, all the three task scores are slightly lower than the experts. In contrast, CycleQD performs significantly better on the MBPP and the OS tasks than the experts. The second is the lack of a mechanism that revives temporarily under-performing models. Since LLMs contain very long computation pipelines (i.e., a large number of layers), it is highly likely that a merged model having only partial error (e.g., a malfunctioning layer) compromises an LLM's performance even though a majority of the other components are perfect. Such a model, though more promising in the future, is eliminated immediately. CycleQD compensates for both insufficiency, making it the top performer.

4 Conclusions

We introduced CycleQD, a compelling modification of the conventional LLM fine-tuning pipeline, integrating evolutionary algorithms for agent skill acquisition. Through a process that cyclically swaps quality and BCs, coupled with a model merging crossover operation and an SVD-based mutation operator, CycleQD has successfully enabled LLMs to master computer science skills.

As additional studies, we test CycleQD's key components (see Appendix A.3.4), generalizability (see Appendix A.3.5), and applicability to other foundation models and domains (see Appendix A.4)

Author contributions

Yujin Tang and Takuya Akiba initiated the project. So Kuroki co-designed the algorithm, implemented CycleQD, conducted the experiments and the studies. Taishi Nakamura developed the agentic tasks, the expert models and the baseline methods. Takuya Akiba implemented the parallel distributed optimization and proposed the SAM experiment. Yujin Tang proposed the algorithm, designed and helped with the experiments, and managed the project. All authors contributed to the writing and approved the final version.

Acknowledgements

The authors would like to thank Robert Lange for providing valuable discussions and feedback while drafting the text, and Kou Misaki for providing infrastructure support. This paper is based on results obtained from a project, JPNP20017, subsidized by the New Energy and Industrial Technology Development Organization (NEDO).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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

Jorge Bernal, F Javier Sánchez, Gloria Fernández-Esparrach, Debora Gil, Cristina Rodríguez, and Fernando Vilariño. Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. *Computerized medical imaging and graphics*, 43:99–111, 2015.

Herbie Bradley, Andrew Dai, Hannah Benita Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente, Jeff Clune, Kenneth O. Stanley, Grégory Schott, and Joel Lehman. Quality-diversity through AI feedback. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=owokKCrGYr`.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. *arXiv preprint arXiv:2403.12881*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. arXiv:2110.14168, 2021.

Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Deng-Ping Fan, Ge-Peng Ji, Guolei Sun, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Camouflaged object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2777–2787, 2020a.

Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. Pranet: Parallel reverse attention network for polyp segmentation. In *International conference on medical image computing and computer-assisted intervention*, pages 263–273. Springer, 2020b.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=ZG3RaNIsO8`.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=d7KBjmI3GmQ`.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=6t0Kwf8-jrj`.

D Jha, PH Smedsrud, MA Riegler, P Halvorsen, T De Lange, D Johansen, and HD Johansen. Kvasir-seg: A segmented polyp dataset. multimedia modeling, 2019.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL `https://aclanthology.org/P17-1147`.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

Trung-Nghia Le, Tam V Nguyen, Zhongliang Nie, Minh-Triet Tran, and Akihiro Sugimoto. Anabranch network for camouflaged object segmentation. *Computer vision and image understanding*, 184:45–56, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze, editors, *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020. URL `https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html`.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL `https://openreview.net/forum?id=1qvx610Cu7`.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023b.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=zAdUB0aCTQ`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL `https://aclanthology.org/D18-1260`.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Thomas Pierrot, Guillaume Richard, Karim Beguir, and Antoine Cully. Multi-objective quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 139–147, 2022.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:202845, 2016.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=dHng2O0Jjr`.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 784–789, 2018. doi: 10.18653/V1/P18-2124. URL `https://aclanthology.org/P18-2124/`.

Sovit Ranjan Rath. Leaf disease segmentation dataset, 2023. URL `https://www.kaggle.com/datasets/sovitrath/leaf-disease-segmentation-with-trainvalid-split`.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob N. Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *CoRR*, abs/2402.16822, 2024. doi: 10.48550/ARXIV.2402.16822. URL `https://doi.org/10.48550/arXiv.2402.16822`.

Przemysław Skurowski, Hassan Abdulameer, J Błaszczyk, Tomasz Depta, Adam Kornacki, and P Kozieł. Animal camouflage analysis: Chameleon database. *Unpublished manuscript*, 2(6):7, 2018.

George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. In *ICLR*, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL `https://aclanthology.org/2023.findings-acl.824`.

Weihao Tan, Ziluo Ding, Wentao Zhang, Boyu Li, Bohan Zhou, Junpeng Yue, Haochong Xia, Jiechuan Jiang, Longtao Zheng, Xinrun Xu, Yifei Bi, Pengjie Gu, Xinrun Wang, Börje F. Karlsson, Bo An, and Zongqing Lu. Towards general computer control: A multimodal agent for red dead redemption II as a case study. *CoRR*, abs/2403.03186, 2024. doi: 10.48550/ARXIV.2403.03186. URL `https://doi.org/10.48550/arXiv.2403.03186`.

Alexey Tikhonov and Max Ryabinin. It's all in the heads: Using attention heads as a baseline for cross-lingual transfer in commonsense reasoning. In *Findings of the Association for Computational Linguistics (ACL)*, pages 3534–3546, 2021. doi: 10.18653/V1/2021.FINDINGS-ACL.310. URL `https://doi.org/10.18653/v1/2021.findings-acl.310`.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jiménez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. In *ICML*, page to appear, 2024a.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024b.

Ren-Jian Wang, Ke Xue, Haopu Shang, Chao Qian, Haobo Fu, and Qiang Fu. Multi-objective optimization-based selection for quality-diversity by non-surrounded-dominated sorting. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 4335–4343. ijcai.org, 2023. doi: 10.24963/IJCAI.2023/482. URL `https://doi.org/10.24963/ijcai.2023/482`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and LINGMING ZHANG. Magicoder: Empowering code generation with OSS-instruct. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=XUeoOBid3x`.

Ryan Wickman, Bibek Poudel, Michael Villarreal, Xiaofei Zhang, and Weizi Li. Efficient quality-diversity optimization through diverse quality species. In Sara Silva and Luís Paquete, editors, *Companion Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2023, Companion Volume, Lisbon, Portugal, July 15-19, 2023*, pages 699–702. ACM, 2023. doi: 10.1145/3583133.3590581. URL `https://doi.org/10.1145/3583133.3590581`.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022. URL `https://proceedings.mlr.press/v162/wortsman22a.html`.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

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.

Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, Philip Torr, Bernard Ghanem, and Guohao Li. CRAB: cross-environment agent benchmark for multimodal language model agents. *CoRR*, abs/2407.01511, 2024. doi: 10.48550/ARXIV.2407.01511. URL `https://doi.org/10.48550/arXiv.2407.01511`.

Ke Xue, Ren-Jian Wang, Pengyi Li, Dong Li, Jianye Hao, and Chao Qian. Sample-efficient quality-diversity by cooperative coevolution. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=JDud6zbpFv`.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html`.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. In *ICLR*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=WE_vluYUL-X`.

Chenchen Ye, Ziniu Hu, Yihe Deng, Zijie Huang, Mingyu Derek Ma, Yanqiao Zhu, and Wei Wang. MIRAI: evaluating LLM agents for event forecasting. *CoRR*, abs/2407.01231, 2024. doi: 10.48550/ARXIV.2407.01231. URL `https://doi.org/10.48550/arXiv.2407.01231`.

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 *ICML*, page to appear, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL `https://aclanthology.org/P19-1472`.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3053–3077. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.findings-acl.181`.

Zihan Zhong, Zhiqiang Tang, Tong He, Haoyang Fang, and Chun Yuan. Convolution meets lora: Parameter efficient finetuning for segment anything model. *ICLR*, 2024.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.

A Appendix

A.1 Related Works

LLM Skill Acquisition: Beyond conversational tasks, increasing attention has been given to enabling LLMs to acquire *skills* to take action. This allows LLMs to function as agents capable of solving problems autonomously, and this approach is known as *LLM-as-Agent* [Xi et al., 2023, Wang et al., 2024b]. Benchmarks for LLM-as-Agent have been developed, and research has demonstrated that LLMs can successfully perform tasks such as computer operations and web browsing to a certain extent [Gur et al., 2023, Liu et al., 2023b, Xi et al., 2024, Xu et al., 2024, Tan et al., 2024, Ye et al., 2024]. *ReAct* [Yao et al., 2023] is the most frequently used approach for constructing LLM-as-Agent systems, which integrates CoT reasoning with action. ReAct is typically employed through prompting and few-shot in-context learning, where models learn skills from a small number of examples. However, attempts to equip LLMs with such skills through actual training with more examples are still quite limited, with a few studies focusing only on standard fine-tuning [Zeng et al., 2024, Qin et al., 2024]. This fine-tuning approach often struggles with balancing multiple agent skills and suffers from the gap between next token prediction loss and actual task performance. Our proposed method uses QD techniques to optimize each skill independently and directly.

Quality Diversity: QD is an emerging paradigm in evolutionary computation that aims to discover a diverse collection of high-performing solutions rather than a single optimal result [Pugh et al., 2016]. QD algorithms balance two key aspects: *quality*, which represents a solution's performance, and *behavior characterization* (BC), which describes its observable properties. These algorithms maintain an *archive* of diverse, high-quality solutions, continuously updated through an evolutionary process. MAP-Elites [Mouret and Clune, 2015] is a prominent QD algorithm that maintains an archive defined by the BCs, discretizes it into tessellation, and stores the best-performing solutions in each cell to preserve diversity. In each generation, it samples parents from the archive, creates offspring through crossover and mutation, and evaluates their quality and BCs. Offspring replace existing elites if they perform better in their respective cells. This process simultaneously explores diverse behaviors and optimizes performance within each niche, resulting in a map of high-quality, diverse solutions. Recent improvements to MAP-Elites include a new selection method for better diversity [Wang et al., 2023] and an approach that removes the need for predefined niches [Wickman et al., 2023]. Additionally, MOQD [Pierrot et al., 2022] has been proposed to address problems with multiple, potentially conflicting objectives while maintaining diversity. Instead of the fitness, it maximizes a Pareto front hyper-volume in each niche of the descriptor space. CycleQD is a simplified form of MOQD in the sense that it uses task metrics as both BCs and the quality, the product of which approximates the hyper-volume, and the coordinate-ascent style of optimization allows it to avoid complex hyper-volume calculations for higher-dimensional archives.

Quality Diversity for LLMs: Evolutionary computation, particularly QD, has occasionally been applied to neural networks and LLMs [Guo et al., 2024, Xue et al., 2024, Akiba et al., 2024]. However, there has been no prior research that evolves the main weights of LLMs using QD. Samvelyan et al. [2024] proposed Rainbow Teaming, which applies QD to evolve diverse adversarial prompts for evaluating LLM safety. Bradley et al. [2024] introduced QDAIF, which applies QD with LLM feedback to evolve populations of generated creative texts. Our work represents the first attempt to merge LLMs through QD. We believe this approach holds great promise, as it allows individual components of the model to be preserved even when they are temporarily sub-optimal, with the potential to contribute to future performance improvements. Given the cost and complexity of optimizing the many layers of LLMs, QD offers a unique advantage by retaining useful configurations in archives that may later enhance overall model performance.

Model Merging: Model merging refers to combining multiple pre-trained models into a single model. This technique is gaining attention because it can enhance model performance and integrate multiple capabilities at a much lower training cost. Additionally, unlike ensemble methods, model merging does not increase inference costs either. This technique is particularly effective when applied to different fine-tuned versions of the same base model. The most basic method involves a linear combination of weights [Wortsman et al., 2022, Ilharco et al., 2023]. More advanced merging techniques have been proposed, incorporating methods such as election and sparsification [Yadav et al., 2023, Yu et al., 2024, Wang et al., 2024a, Stoica et al., 2024]. Furthermore, the applicability and performance of these methods have been significantly improved through optimization [Akiba et al., 2024, Yang et al., 2024]. Inspired by these merging strategies, CycleQD combines QD with model merging, allowing for more effective synthesis of multiple capabilities in LLMs.

A.2 Methods

A.2.1 Alternating Quality and Behavior Characteristics

CycleQD manages K archives, each dedicated to tracking the LLMs that specialize in one of the K agent skills. Each archive evaluates LLM performance using K tasks, which serve dual roles as both the quality and BCs. An archive is structured as a lattice, the size of which is defined by $\prod_{k \neq i} d_k$, where d_k is the number of bins defined by the k-th BC, and the i-th BC is excluded from this particular archive as it is treated as the quality metric. In generation t, the i-th archive is selected for QD computation where $i = t \mod K$ (see Figure 1 for an illustration with $K = 3$). This process

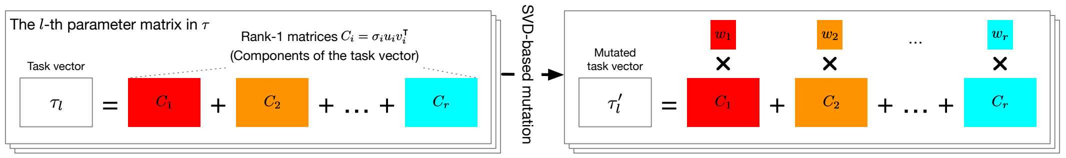
The l-th parameter matrix in τ
Rank-1 matrices $C_i = \sigma_i u_i v_i^\intercal$
(Components of the task vector)
Task vector
τ_l = C_1 + C_2 + … + C_r
SVD-based mutation
Mutated task vector
w_1 w_2 … w_r
τ_l' = C_1 + C_2 + … + C_r

Figure 3: **SVD-based mutations.** Left: A task vector τ contains multiple parameter matrices τ_l. E.g., the query projection matrix from attention in layer 1, the key projection matrix from attention in layer 2, etc. Those that have a rank $r > 1$ can be decomposed using SVD into r components. Right: SVD-based mutation samples a vector $w \in \mathbb{R}^r$ and scales each component by w_i, essentially adding perturbations that are aligned with the "directions" of the components.

utilizes all available data from the K tasks to evaluate and update the archives without the need for adjusting data ratios or objective functions.

CycleQD shares knowledge across K archives for efficient optimization. Specifically, in generation t, a new model is created from the i-th archive by sampling, crossover, and mutation. This model is used to update not only the i-th archive but all K archives (lines 14-15 in Algorithm 1). This allows for more effective utilization of the new model and helps facilitate optimization across K tasks.

We employ task performance metrics as our BCs, imbuing them with concrete performance implications. This allows for the design of an Elite sampling algorithm aimed to expand the Pareto frontier. This Elite sampling algorithm is specifically designed for the CycleQD setup, where both quality and BCs reflect task performance. In this context, it is more effective to prioritize models that push the frontier of higher performance. This frontier is defined by the models achieving high performance across these BCs. Concretely, when the i-th archive is active, a model j inside it is sampled with probability $P_j = \frac{\gamma_j}{\sum_{n=1}^{N} \gamma_n}$ where N is the number of models in this archive and γ_j is calculated as follows:

$$\gamma_j = \prod_{i=1}^{K} \left(\alpha_{\text{low}} + \frac{f_{j,i} - \min(f_{1:N,i})}{\max(f_{1:N,i}) - \min(f_{1:N,i})}(\alpha_{\text{high}} - \alpha_{\text{low}})\right)$$

Here, $f_{j,i}$ indicates the j-th model's performance on the i-th task. $(\alpha_{\text{low}}, \alpha_{\text{high}})$ are hyper-parameters for the purpose of normalization. Elite sampling strategically favors models that excel across the quality and various BCs, enhancing the efficiency and evolutionary potential of CycleQD.

A.2.2 Model Merging-based Crossover

Training an LLM to specialize in a single task does not suffer from the data ratio tuning and the inadequate objective function problems mentioned earlier. It is therefore straightforward to initially train a set of single-task experts using conventional fine-tuning methods, and then use model merging algorithms to develop more capable agents. In CycleQD, our model merging crossover operator employs the parameter space merging scheme described in Akiba et al. [2024], which creates a new model by merging task vectors at the model level [Ilharco et al., 2022]. Specifically, for a pre-trained base LLM with parameters $\theta_{\text{base}} \in \mathbb{R}^d$, we define a task vector for task k as $\tau_k = \theta_k - \theta_{\text{base}}$, where $\theta_k \in \mathbb{R}^d$ are the parameters of the LLM fine-tuned for that task. The crossover operator then generates a model's parameters by combining these task vectors: $\theta_{\text{child}} = \theta_{\text{base}} + \big(\omega_1/(\omega_1 + \omega_2)\big)\tau_{p_1} + \big(\omega_2/(\omega_1 + \omega_2)\big)\tau_{p_2}$, where τ_{p_1} and τ_{p_2} are the parents' task vectors. Here, ω_1 and ω_2 are i.i.d samples from $\mathcal{N}(\mu, \sigma^2)$, and (μ, σ) are predetermined hyper-parameters that remain fixed during the experiments. We normalize (τ_{p_1}, τ_{p_2})'s mixing coefficients to ensure the merged weights do not become outliers and cause problems for the downstream layers.

A.2.3 SVD-based Mutation

The formulation of our model merging-based crossover bears two obvious limitations: (1) the construction of θ_{child} is a linear combination of (τ_{p_1}, τ_{p_2}), which are themselves linear combination of their parents. The entire process then reduces to finding the optimal linear combination of the expert models' parameters, the combination coefficients of which can be effectively searched with gradient descent, eliminating the need for QD; (2) Due to its nature of construction, θ_{child} will likely be trapped in the "convex region" in the performance space formed by its parents, whereas extrapolation that leads to improved performance is desired. We introduce a mutation function $\theta_{\text{child}} = h(\theta_{\text{child}})$ after the crossover to get rid of these limitations.

In conventional methods, the mutation function $h(\theta_{\text{child}})$ is often an operator that samples perturbations from a pre-defined distribution (e.g., Gaussian with pre-determined mean and covariance matrix) and add them to the gene being mutated. We find this setting introduce excess freedom and lead to overfitting in the final model. Instead, we propose to sample perturbations along the "directions" of the components from the model's parameter matrices. This mutation operation is mathematically defined as: $h(\theta_{\text{child}}) = \theta_{\text{base}} + \text{concat}\big([U_l(\Sigma_l w)V_l^{\intercal}]_{l=1}^{L}\big)$, where $U_l \in \mathbb{R}^{m\times r}$, $\Sigma_l \in \mathbb{R}^{r\times r}$ and $V_l \in \mathbb{R}^{n\times r}$ are the SVD components of $\tau_l = U_l\Sigma_l V_l^{\intercal}$, the l-th parameter matrix in the task vector $\tau_{\text{child}} = \theta_{\text{child}} - \theta_{\text{base}}$ (i.e., τ_l is a reshaped subarray of τ_{child}). The perturbation vector $w \in \mathbb{R}^r$ is sampled from a uniform distribution of boundaries $[0, w_{\max}]$, where $w_{\max} \in \mathbb{R}^+$ is a hyper-parameter. Our SVD-based mutation becomes a pass-through operation for parameter matrices whose rank is one (e.g., those belonging to layer-normalization layers).

Intuitively, if task vectors are representative of the agent skills each fine-tuned LLM has mastered, their components are the building blocks and are therefore proxies in a finer granularity. As a result, our SVD-based mutation allows finer control of θ_{child} in the fundamental skill building blocks and reduces the risk of overfitting. See Figure 3 for an illustration.

A.2.4 Model Aggregation

CycleQD maintains K archives during the optimization process, LLMs in each of these archives are trained to acquire one of the K agent skills. In the end, CycleQD returns hundreds or even thousands of LLM agents with various specialties and characteristics. These LLM-based agents are versatile and can facilitate multi-agent researches in a diverse set of directions. On the other hand, an aggregated LLM agent that possesses the top skills acquired by the best models in each archive is useful on its own and is important for the evaluation of CycleQD.

Our model aggregation algorithm is straightforward, and can be summarized as: $\theta_{\text{agg}} = \theta_{\text{base}} + \sum_{k=1}^{K}\beta_k\tau_k$, where τ_k is the task vector from the elite model in the k-th archive, with ties broken by model age, and the most recently generated model is selected. $\beta_k = \exp(Q_k)/\sum_{i=1}^{K}\exp(Q_i)$ (i.e., softmax function) is the mixing coefficient calculated from the elite model's task performance Q_k in the k-th archive. In this paper, all experimental results from CycleQD are produced with this aggregated model θ_{agg}.

A.3 Evaluation on Computer Science Tasks

A.3.1 CycleQD Setups

Experts training details We adopt the AdamW optimizer [Loshchilov and Hutter, 2019] with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. A global batch size of 64 is used across all fine-tuning processes. We employ cosine learning rate scheduling with a range of $[4\times10^{-6}, 2\times10^{-5}]$, starting with a linear warmup for the first 10% of the total training steps. The OS and DB experts are trained for 1 epoch, while the code model is trained for 3 epochs due to larger training data size.

Hyper-parameters: We use the three experts' performance to determine the lower and upper bounds of the corresponding BC dimension. Specifically, the lower bound is set at 85% of the performance achieved by the least proficient expert, while the upper bound is set at 115% of the performance achieved by the most proficient expert. All BCs are then evenly divided into 15 bins between the lower and upper bounds. We limit the number of models in each bin to one, and run CycleQD for 1200 generations. Since the quality and BCs are alternated in each generation, this is equivalent to optimize for the three skills for 400 generations each. We set $\alpha_{\text{low}} = 0.5$ and $\alpha_{\text{high}} = 0.8$ in Elite sampling, $\mu = 1.0$ and $\sigma = 0.03$ in model merging-based crossover, and $w_{\max} = 0.3$ in our SVD-based mutations.

A.3.2 Baselines

Our baselines can be categorized into fine-tuning based and merging based models.

Fine-tuning Based Models: These are models 2-5 in Table 1. In addition to the expert LLMs mentioned earlier, we also combine all the data from the three tasks (including the extra MAGICODER-EVOL-INSTRUCT-110K and MAGICODER-OSS-INSTRUCT-75K datasets) and continue training the LLAMA3-8B-INSTRUCT base model with supervised fine-tuning to develop an extra baseline. In

Table 2: **Ablation studies.** We add and highlight a different treatment from the previous trials.

#	Trials	MBPP	DB	OS	Avg
0	QD + No mutation + Random sampling	70.4	28.8	**43.7**	47.6
1	CycleQD + No mutation + Random sampling	72.9	33.5	41.9	49.4
2	CycleQD + Gaussian mutation + Random sampling	73.4	30.0	42.2	48.5
3	CycleQD + SVD mutation + Random sampling	75.9	**38.2**	41.1	51.7
4	CycleQD + SVD mutation + Elite sampling	**76.4**	**38.2**	42.6	**52.4**

this baseline, we don't tune the data ratio and use cross-entropy (i.e., next token prediction) as our objective function.

Merging Based Models: These are models 6-8 in Table 1. We first introduce a naive merging method where the merged model is produced by taking an average of the three experts' task vectors (model 6). Similar to this baseline, we introduce two additional learning based model merging methods where, instead of taking an average, the coefficients of a linear combination of these task vectors is learned through gradient descent on policy gradients (model 7), and CMA-ES, an evolutionary algorithm on the raw rewards (model 8) separately.

A.3.3 CycleQD development of archives across generations.

Figure 4 shows the development of archives across generations. The archives are shown in increments of 300 generations from top to bottom. The corresponding generation for each archive is displayed on the left side of the figure. The red bounding boxes indicate the grids where expert policies were present in each archive. It can be observed that the frontier of the archives expands with each passing generation.

A.3.4 Ablation Studies

To get insights to the design choices in CycleQD and show how they contribute its effectiveness, we conduct a series of ablation studies and summarize the results in Table 2.

CycleQD vs QD: We compare conventional QD with CycleQD in trials 0 and 1. Specifically, we run QD for three runs. Within each run, one of the task's metric is treated as the quality while the others are regarded as the BCs. Each run lasts for 400 generations, an identical computing budget as CycleQD, and all other hyper-parameters are identical to CycleQD. We use the same model aggregation method to merge the best models from these three runs (see Appendix A.2.4). The better performance from CycleQD suggests the importance of alternating the quality and BCs.

Mutation Methods: We focus on the impact of different mutation operations with trials 1-3. Although we mentioned earlier that mutation helps the merged model extrapolate in the performance space, naively designed mutations give excess freedom and can lead to overfitting. This is what happens to trial 2, the performance of which is even worse than trial 1 that does not have any mutation operations. On the other hand, our SVD-based mutation (trial 3) successfully avoids the problem and delivers a much better performance.

Sampling Methods: Finally, we demonstrate the importance of Elite sampling in CycleQD in trial 4, which has the identical settings as model 9 in Table 1, and gives the best score in the ablation studies. Elite sampling is effective because it prioritizes merging high-performing models, the result of which helps expand the Pareto frontier in each archive more effectively.

A.3.5 Generalization and Language Capabilities

We evaluate our model across diverse categories of tasks to assess its performance in various aspects of language understanding, reasoning, and code generation. Our evaluation framework encompasses six main categories: In the Coding category, we employ two key benchmarks: HumanEvalLiu et al. [2023a], evaluated in a zero-shot setting, and BigCodeBench. BigCodeBench, introduced by Zhuo et al. [2024], is designed to assess code generation with diverse function calls and complex instructions. We report the average Pass@1 scores for both BigCodeBench-Complete and BigCodeBench-Instruct variants in the full setting, using zero-shot evaluation with temperature set to 0. For the General Knowledge and Reasoning (GK & Reasoning) category, we utilize two comprehensive benchmarks:

Table 3: **Generalization performance.** The scores are normalized against the base model. CycleQD generalizes well to other coding tasks unseen in the training while retaining its language capabilities.

Model	Coding Tasks		Language Tasks				Avg
	HUMANEVAL+	BigCodeBench	Reasoning	GSM8K	RC	CommonSense	
MBPP expert	1.18	0.97	0.57	0.82	0.94	1.03	0.92
DB expert	0.80	0.84	0.84	0.87	0.98	0.98	0.89
OS expert	0.94	0.90	0.98	0.93	0.99	0.99	0.95
CycleQD	1.10	1.03	0.95	0.88	0.98	1.02	0.99

MMLU [Hendrycks et al., 2021] with a 5-shot evaluation, and BBH (Big Bench Hard) [Suzgun et al., 2023] using a 3-shot chain-of-thought [Wei et al., 2022] prompting approach. These benchmarks provide a holistic assessment of our model's ability to handle a wide range of knowledge-based and reasoning tasks. To evaluate Mathematical Reasoning, we employ the GSM8K benchmark [Cobbe et al., 2021] with a 4-shot evaluation, challenging our model's ability to solve complex mathematical problems. For Reading Comprehension (RC), we use two established benchmarks: SQuAD2 [Rajpurkar et al., 2018] and TriviaQA [Joshi et al., 2017], both evaluated using a 4-shot approach. These datasets assess the model's capacity to understand and reason over complex textual information. In the Commonsense Reasoning category, we employ three diverse benchmarks: HellaSwag [Zellers et al., 2019] for commonsense inference, OpenBookQA [Mihaylov et al., 2018] for open-domain question answering, and XWinograd (English version) [Tikhonov and Ryabinin, 2021] for cross-lingual commonsense reasoning. All these evaluations are conducted with 4-shot prompts. The results of these evaluations are presented in Table 3, offering a detailed view of our model's capabilities in each category, including an overall average across all benchmarks.

Table 3 shows the generalization results across various tasks. On average, our method achieves the highest score of 0.6077, outperforming the OS expert (0.6021), DB expert (0.5643), and MBPP expert (0.5528).This demonstrates that our method provides better overall generalization across tasks.

The expert models, such as the MBPP expert, show significantly lower performance on specific tasks like BBH-CoT (0.1054). This sharp decline in performance suggests the occurrence of catastrophic forgetting during fine-tuning, where the model loses its ability to generalize across tasks outside its specialization. In contrast, CycleQD maintains strong overall performance across tasks while effectively balancing optimization for the target tasks (MBPP, OS, DB).

A.4 Evaluation on Image Segmentation Tasks

A.4.1 Task Description

In addition to its applications in LLMs, CycleQD serves as a versatile method for integrating expert models across various data modalities beyond text. In this experiment, we extend CycleQD to the fusion of multiple Segment Anything Models (SAM) [Kirillov et al., 2023], which are state-of-the-art computer vision models designed for image segmentation tasks. Specifically, our objective is to merge pairs of SAM models, A and B, to create models whose capabilities encompass the skill sets of both A and B.

A.4.2 CycleQD Setups

Experts: We select four tasks within the image segmentation domain, each supported by extensive datasets, for training specialized models: Camouflaged Object Segmentation (CAM), Polyp Segmentation (POL), Skin Lesion Segmentation (SKL), and Leaf Segmentation (LEA). Camouflaged Object Segmentation focuses on detecting objects hidden within complex or cluttered backgrounds, presenting a greater challenge than traditional object segmentation. Polyp Segmentation identifies abnormal growths (polyps) in gastrointestinal endoscopic images, which is crucial for early colorectal cancer diagnosis. Skin Lesion Segmentation targets the identification of various types of skin lesions in medical images. Lastly, Leaf Segmentation involves detecting individual plant leaves in agricultural images, aiding plant disease control and improving food production quality. We employ the SAM-ViT Huge as our base model, which we fine-tune with the datasets from Zhong et al. [2024] to develop these experts.

Datasets: For Camouflaged Object Segmentation, we use three datasets: COD10K [Fan et al., 2020a], CHAMELEON [Skurowski et al., 2018], and CAMO [Le et al., 2019]. Following Fan et al. [2020a] we train on a combined dataset consisting of the 4040 training images from COD10K and CAMO for 20 epochs, randomly splitting 10% of the images from the training set for validation. The model is then tested on the 250 COMO test images. For Polyp Segmentation, we use two datasets: Kvasir [Jha et al., 2019] and CVC-ClinicDB/CVC-612 [Bernal et al., 2015]. Following Fan et al. [2020b], we divide the images into a 9:1 ratio for training and testing, resulting in 1450 training images. We then randomly split 20 % of the training set for validation. The model is trained for 30 epochs and tested on the 101 Kvasir test images. For Skin Lesion Segmentation, we use the ISIC 2017 dataset [Codella et al., 2018]. We train the model on the 2000 training and 150 validation images for 30 epochs and evaluate it on the 600 test images. For Leaf Segmentation, we use the Leaf Disease Segmentation Dataset [Rath, 2023]. We train the model on the 498 training image, using 80% for training and 20% for validation for 30 epochs, and evaluate it on 90 test images.

Hyper-parameters: CycleQD's hyper-parameters remain the same as in Appendix A.3.1.

A.4.3 Results

Overall: Table 4 shows the performance of the models merged by CycleQD, where scores are normalized against the expert models. In general, CycleQD is able to merge the experts successfully, with top models retaining more than 90% of the experts' performance (e.g., models 0, 1, and 3). On the other hand, models 2, 4 and 5 are less successful. This leads us to conduct further analysis and we report the findings next.

Analysis: We report the similarities between experts A and B in the last column in Table 4. A strong correlation of 0.83 between the averaged scores and these similarities indicate both the limitations and potential areas for enhancement in CycleQD. We define the similarity between models A and B as the average cosine similarity of the singular value vectors (derived from the task vectors) across all layers in both models: $s = (1/L)\sum_{i=1}^{L} \cos\left(\text{diag}(\Sigma_{i,A}), \text{diag}(\Sigma_{i,B})\right)$. Here, L is the number of weight matrices with a rank greater than 1, and $\Sigma_{i,*}$ denotes a diagonal matrix of singular values from the i-th weight matrix in the task vector. Although this summarizing metric does not fully encapsulate the models' characteristics, CycleQD generally performs well when the models exhibit high similarity. This observation underpins our approach of incorporating this similarity metric as a regularization technique during model fine-tuning, which we anticipate will encourage a cyclical process of continuous learning and model integration. Notably, a similar metric has been employed as the "proximal term" in optimizing heterogeneous networks within the realm of federated learning, as evidenced by research documented in [Li et al., 2020]. This precedent lends credence to our strategy, suggesting it is grounded in established methodologies.

Table 4: **Performance on image segmentation tasks**. The scores are normalized against the performance of the corresponding expert models. Model similarity shows how close the pair of expert models are (the larger the similarity, the closer the models).

#	Expert A	Expert B	Score A	Score B	Avg Score	Model Similarity
0	CAM	POL	0.95	0.99	0.97	0.98
1	CAM	SKL	0.85	0.99	0.92	0.97
2	CAM	LEA	0.51	0.89	0.70	0.88
3	POL	SKL	0.98	0.95	0.96	0.99
4	POL	LEA	0.40	0.84	0.62	0.93
5	SKL	LEA	0.83	0.84	0.83	0.95

A.4.4 Visualization result

The visualization results are shown in Figure 5 for SKL and POL, and Figure 6 for SKL and LEA. In both figures, "Ours" refers to a single merged model via CycleQD, identical to model #3 (for SKL and POL) and model #5 (for SKL and LEA) in Table 4. In Figure 5, our model retains near expert-level performance for both tasks after merging. However, in Figure 6, while our model generally performs at an expert level for both tasks, a slight drop is observed. Notably, in the bottom images of SKL, the model seems to detect the edges of the skin lesions instead of the lesions themselves, as if confusing the task with leaf disease detection. The expert model similarities are high—0.99 for SKL and POL, and 0.95 for SKL and LEA. This corresponds with the observation that the merged model from the highly similar SKL and POL performs better than the one from SKL and LEA.

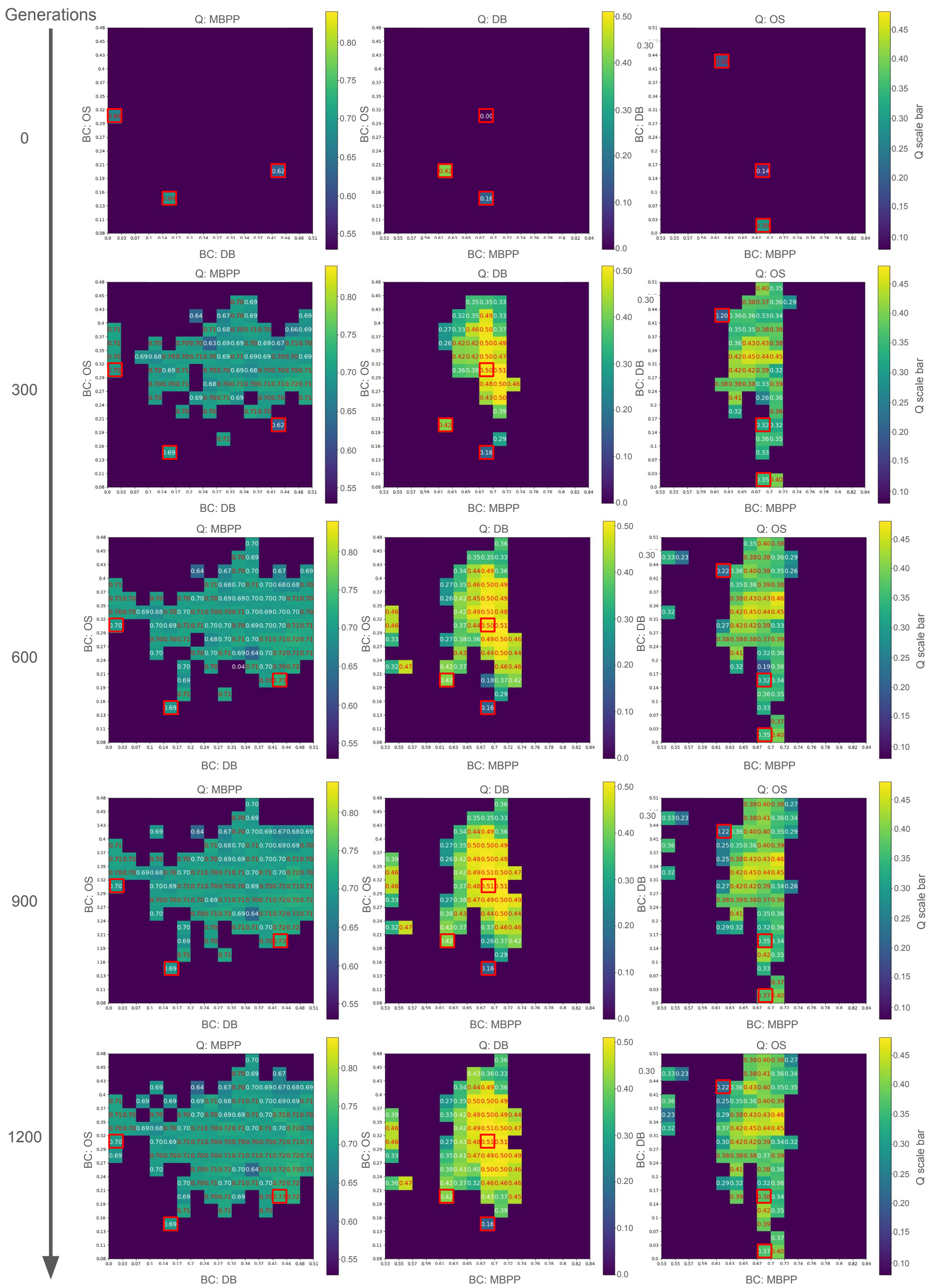

Figure 4: **CycleQD Development of Archives Across Generations.** In each archive, the two axes represent the BCs, and the color intensities in each grid indicate the quality of the LLM agent in that grid. The archives are shown in increments of 300 generations from top to bottom. The corresponding generation for each archive is displayed on the left side of the figure. The red bounding boxes indicate the grids where expert policies were present in each archive. The experiment shown in this figure is the same as in Figure 2, and the archive for 1200 generations is identical to that in Figure 2.

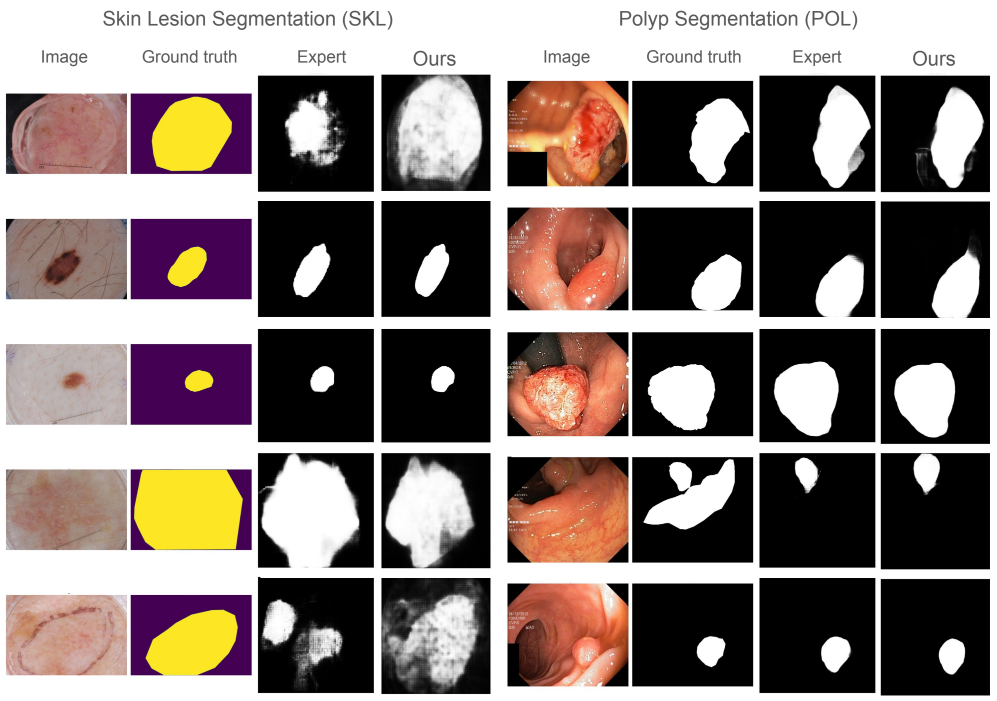

Figure 5: **CycleQD of SAM for SKL and POL.** Ours is a single merged model via CycleQD, and it is the same model as the one in Table 4 (# 3). The similarity between the two expert models is high (0.99), and the model maintains expert-level performance across both tasks.

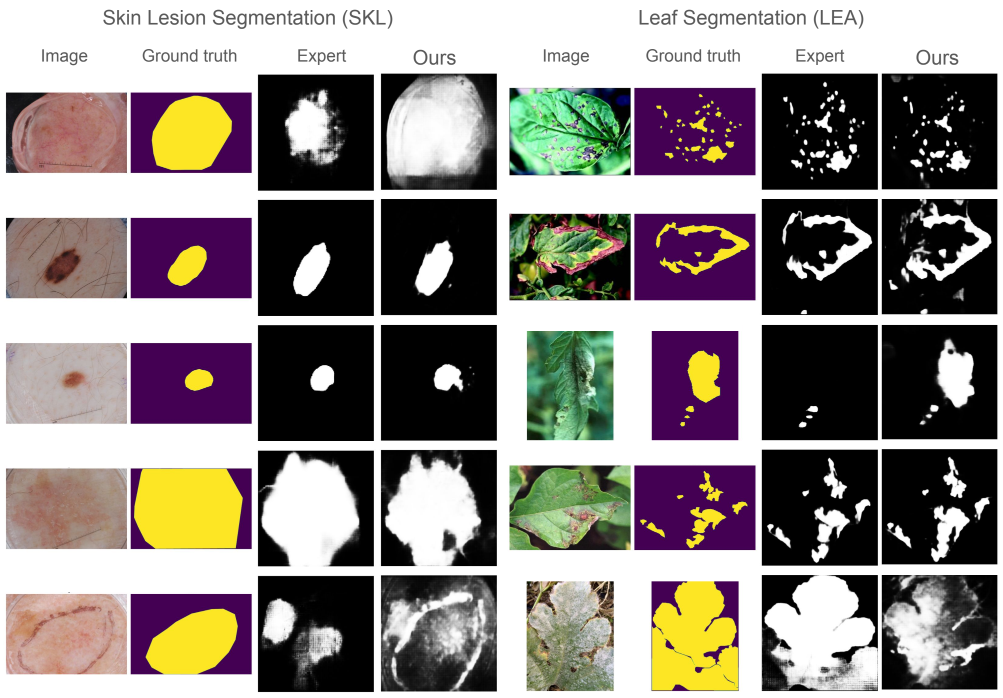

Figure 6: **CycleQD of SAM for SKL and LEA.** Ours is a single merged model via CycleQD, and it is the same model as the one in Table 4 (# 5). The similarity between the two expert models is 0.95, which is lower than #3 (0.99) from Table 4, resulting in a slight decrease in performance, as observed when comparing the bottom images in SKL.