# TOWARDS SCALING LAWS FOR LANGUAGE MODEL POWERED EVOLUTIONARY ALGORITHMS: CASE STUDY ON MOLECULAR OPTIMIZATION

**Tigran Fahradyan, Filya Geikyan, Philipp Guevorguian, Hrant Khachatrian**
YerevaNN research lab
{tigranfahradyan,filya,philipp,hrant}@yerevann.com

## ABSTRACT

The performance improvement of large language models (LLMs) mainly came from scaling pretraining. However, a new scaling paradigm emerged called *test-time compute*, which uses more computation at the inference time of language models to get better results. There have been extensive works suggesting various test-time compute scaling strategies, but derivation of laws describing the scaling dynamics of these methods is an open research question. In this work we try to bridge this gap, developing a parametric law for language model-enhanced evolutionary algorithms depending on the language model parameters ($N$) and number of evolutionary iterations ($k$) used. We show that in molecular optimization tasks, our law is able to accurately extrapolate $2.5\times$ in $N$ and $k$. Additionally, our law suggests that there is a tradeoff between $N$ and $k$, which we validate by matching the performance of a 3.2B model with a $8.5\times$ smaller 380M model using $2.3\times$ more evolutionary algorithms steps.

## 1 INTRODUCTION

Over the past years, the improvement of large language models (LLMs) primarily came from scaling pretraining. However, recently, a new scaling paradigm emerged called *test-time compute*, which tries to spend extra computation during the inference of language models to get better performance. Perhaps the simplest forms of using more test-time compute include methods such as, Consensus (Wang et al., 2022), best of N (Cobbe et al., 2021) and Chain-of-Thought (Kojima et al., 2022). More advanced methods utilizing test-time compute (in various forms) have demonstrated promising results in domains like games (Silver et al., 2018), (Brown & Sandholm, 2019), (Jones, 2021), general reasoning (Jaech et al., 2024), (Guo et al., 2025), (Muennighoff et al., 2025), mathematics problem solving (Xin et al., 2024), coding problem solving (alp).

Although a wide range of strategies have been developed for increasing test-time compute, to the best of our knowledge, there are no works developing laws describing the scaling dynamics of these strategies. In this work, we try to bridge this gap. We focus on the family of evolutionary algorithms powered by language models. The development of powerful LLMs has enabled their use in the implementation of crucial operations (such as crossover or mutation) in evolutionary algorithms. The synthesis of evolutionary algorithms and language models lead to a significant boost in their capabilities in applications, like prompt optimization (Guo et al., 2023), neural architecture search (Chen et al., 2023), math problems solving (Yang et al.) and molecular optimization (Wang et al., 2024), (Guevorguian et al., 2024).

Our contributions are the following:

- We introduce a parametric law for modeling the scaling dynamics of language model-powered evolutionary algorithms.
- We test our law on challenging multi-property molecular optimization tasks and find that our law successfully extrapolates 2.46 and 2.5 times along the language model parameters ($N$) and number of evolutionary algorithm iterations ($k$), respectively. Moreover, we find

that while accurate in the extrapolation regime, our law is unreliable at the earlier stages of the evolutionary algorithm.

- We find a tradeoff between the language model parameters ($N$) and the number of evolutionary algorithm iterations ($k$) and show that the performance of larger models can be accessible from smaller models (as derived by our law). Importantly, we validate the prediction of our parametric law that 380M language model should match the performance of a ($8.5\times$ larger) 3.2B model, through running $2.3\times$ more iterations.

## 2 Related work

Extensive work has been done to model the scaling dynamics of neural language models at different stages. Kaplan et al. (2020), Hernandez et al. (2021), and Hoffmann et al. (2022) model language model training log-likelihood loss, giving valuable insights into the optimal design choices for language models. Aghajanyan et al. (2023) extend the training loss modeling to the multi-modal setting. Muennighoff et al. (2023) study the training loss in data-constrained settings (i.e., when using the same data point multiple times).

Isik et al. (2025) suggest a logarithmic model for the BLEU score prediction. Dubey et al. (2024) and Bhagia et al. (2024) use a two-stage approach, firstly modeling the training loss, then the task performance based on it, using a sigmoid-based model. Chen et al. (2024) introduce a law for the probability of solution correctness for test-time compute strategies.

## 3 Scaling laws for evolutionary algorithms

We propose a parametric law for predicting the expected performance of the family of methods that use evolutionary algorithms powered by transformer-based language models. These approaches typically consist of training a language model and then defining an evolutionary algorithm in terms of the language model.

### 3.1 Parametric law

For $N$ parameter transformer, trained on a fixed number of tokens and $k$ evolutionary algorithm iterations, we propose the following parametric law for the expected performance of a task

$$\hat{S}(N,k) = 1 - A - \frac{B}{(\beta N^{\alpha})^{k^{\gamma}}} \tag{1}$$

where $A, B, \alpha, \beta$ and $\gamma$ are parameters to be fitted. In A.1, we provide the derivation of our law.

## 4 Scaling laws for molecular optimization

In this section, we test our parametric law with molecular optimization tasks. In the context of this work, molecular optimization refers to finding molecules that satisfy some desired properties. Our method consists of using 170M, 380M, 750M and 1.3B language models and up to 20 iterations of the evolutionary algorithm to estimate the parameters of our law, then for testing it we use 3.2B parameter model and up to 50 evolutionary iterations.

### 4.1 Chemical language models

The Llama 3 (Dubey et al., 2024) transformer architecture is chosen for training chemical language models. We train transformers in the sizes of 170M, 380M, 750M, and 1.3B for scaling law derivation. Since Llama 3's smallest architecture has 1B parameter transformer, we manually derive 170M, 380M and 750M parameter architectures. A.2 contains the architecture configurations used for each model size. We train the language models with AdamW Loshchilov & Hutter (2019) and with Warmup-Stable-Decay (WSD) learning rate scheduler Hu et al. (2024) using Fully Sharded Data Parallel (FSDP) Zhao et al. (2023) distributed training method.

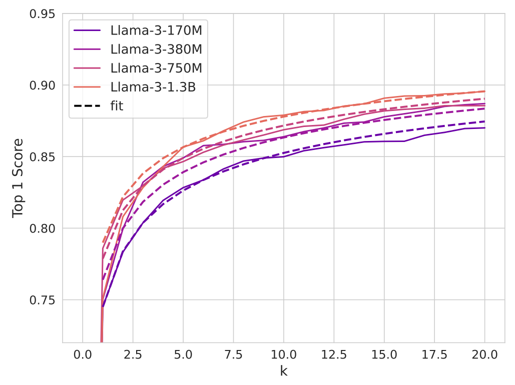

Figure 1: The empirical (solid lines) and predicted (dashed lines) top-1 scores for 170M-1.3B models.

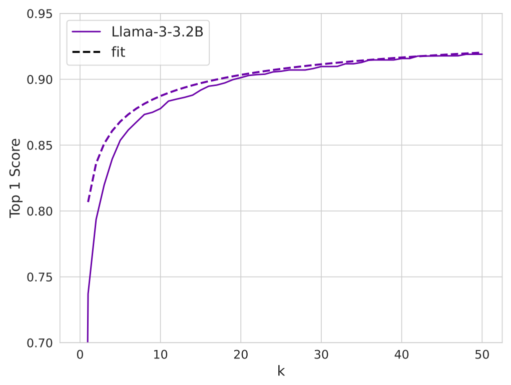

Figure 2: The result of testing our law derived with 170M-1.3B language models and 20 evolutionary iterations with a 3.2B parameter language model with 50 evolutionary iterations. The empirical and predicted top-1 scores are represented in solid and dashed lines, respectively.

We employ the extensive chemical dataset curated in Guevorguian et al. (2024), based on the PubChem knowledge repository, which includes around 100 million molecules, their properties, and their relationships. We utilize the tokenizer that comes with Llama 3 pre-trained models and, after some data processing, obtain a dataset consisting of 40B tokens for training our models.

Additionally, we use a special case of the evolutionary algorithm developed in Guevorguian et al. (2024), which has demonstrated competitive performance in a variety of molecular optimization tasks. This evolutionary algorithm relies on the notion of similarity defined in terms of molecule fingerprints. The algorithm keeps a pool of the best molecules so far and, at each step, queries the language model to generate new molecules similar to the existing good ones, hence defining the crossover and mutation operations. Algorithm 1 provides a high-level overview of the evolutionary algorithm used, but for further clarification, we ask the reader to refer to Guevorguian et al. (2024).

## 4.2 Molecular optimization task selection

We model the average top-1 score of the sitagliptin_mpo, perindopril_mpo and ranolazine_mpo multi-property optimization tasks from the Guacamol Benchmark (Brown et al., 2019). Additionally, to obtain a more reliable performance estimate, for each task, ten independent runs are executed (with 10 different seeds), and the results are averaged.

## 4.3 Parameter estimation

The evolutionary algorithm is run for 20 iterations with 170M, 380M, 750M, and 1.3B sized models, and data is collected for each model size and per iteration, totaling 80 data points. Following the

**Algorithm 1**

```
Input: $k$ (number of iterations), $u$ (pool update frequency), $p$ (pool size)
$pool = \{\}$                                  // create a pool
for $i = 1, \dots, k$ do
    $m_1, \dots, m_u \sim pool$                 // sample $u$ molecules from the pool
    $m_1^{new}, \dots, m_u^{new} \sim LM(m_1, \dots, m_u)$     // sample $u$ new molecules from the LM
    $pool = top\text{-}p(pool \cap \{m_1^{new}, \dots, m_u^{new}\})$   // keep the best $p$ molecules in the $pool$
end for
```

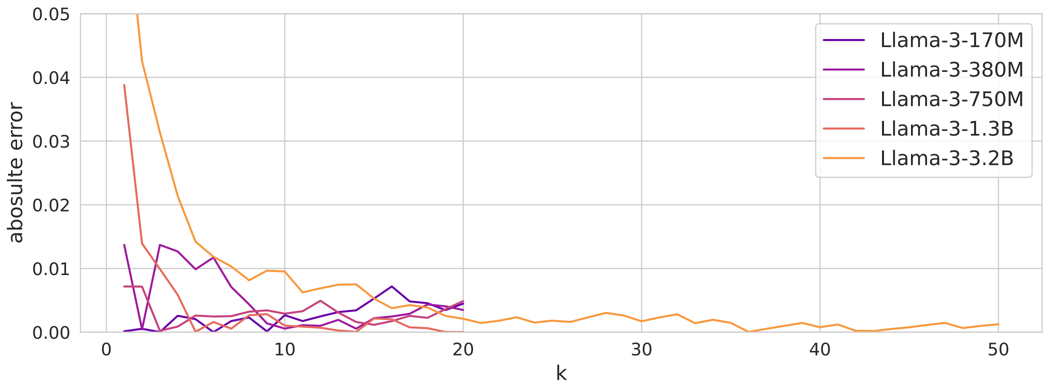

Figure 3: Absolute error between the observed and predicted top-1 score for 170M-3.2B models. The general trend is that the error decreases as $k$ grows.

methodology in Hoffmann et al. (2022), we minimize the *Huber* loss between the predicted and observed log scores, to estimate $A, B, \alpha, \beta$ and $\gamma$.

$$\min_{A,B,\alpha,\beta,\gamma} \sum_i \textit{Huber}(\textit{LSE}(a, b - k_i^\gamma \log \beta N_i^\alpha)) - \log(1 - S_i)) \tag{2}$$

where *LSE* refers to the log-sum-exp function, and then we set $A = e^a, B = e^b$. $\delta = 0.001$ is used for *Huber* loss. We optimize our parametric law with the L-BFGS algorithm and a grid of initialization to avoid local minimum points, resulting in $A = 0.03, B = 142, \alpha = 0.11, \beta = 78$, and $\gamma = 0.042$ estimates. Figure 1 visualizes the observed and predicted top-1 scores.

### 4.4 Extrapolation

To test our derived model in the extrapolation regime, we train a 3.2B parameter language model and execute the evolutionary algorithm for 50 iterations. Figure 2 shows the predicted and the observed top-1 scores.

At $k = 50$, the predicted and observed top-1 scores are 0.918 and 0.920 respectively, with 0.002 absolute error. This suggests that our law simultaneously extrapolates in the number of language model parameters ($N$) and in the number of evolutionary algorithm iterations ($k$) 2.46 and 2.5 times, respectively.

## 5 Analysis

Firstly, we find that the top-1 score is hard to model for smaller values of $k$ due to the lack of regularity. Figure 4 illustrates the observed top-1 scores for all model sizes up to 7 iteration, which shows the order of language models with respective parameter counts is inconsistent, making it challenging to predict. We hypothesize that this is due to high stochasticity in the evolutionary algorithm at the earlier stages and leave the investigation of this phenomenon as a future work.

Although hard to predict at earlier steps, as $k$ gets larger, the top-1 score gets more stable, and our model is able to accurately capture the top-1 performance, as can be seen in Figure 1 and 2. To facilitate a more comprehensive understanding, Figure 3 represents the absolute error between the predicted and observed top-1 scores for all model sizes.

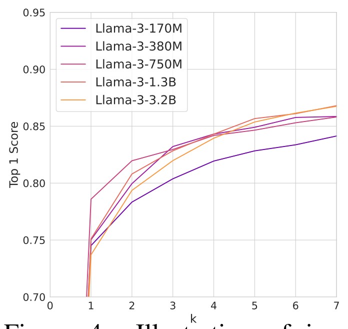

Figure 4: Illustration of inconsistent ordering between different model sizes at the initial stages.

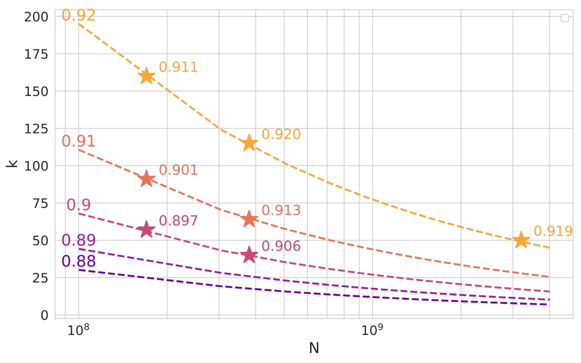

Figure 5: This figure represents the tradeoff between $N$ and $k$, by showing the configurations of $N$ and $k$ to choose for achieving top-1 $0.88 - 0.92$ scores. Figure inspired from Jones (2021)

### 5.1 Model size and evolutionary steps tradeoff

In this section we show the tradeoff between $N$ and $k$. Figure 5 $x$ axis represents the number of language model parameters to choose, and the $y$ axis shows the number of evolutionary iterations to use to get the desired performance. The dashed lines represent the prediction of the scaling law. To validate this prediction, we run 170M and 380M models for the appropriate number of iterations to achieve the predicted $0.9, 0.91$, and $0.92$ top-1 scores. The stars represent the exact configuration used for testing along with the empirical result. We find that for $N = 170$M model our law correctly predicts the top-1 score for $k = 57$, but makes a $0.01$ error for 91 and 160. For $N = 380$M model, it correctly predicts the top-1 score at $k = 115$. These findings suggest that, indeed, there is a tradeoff between $N$ and $k$.

In practice training a larger language model can be inaccessible due to compute or data quality or quantity constraints. However, our experiments suggest that the capabilities of larger language models may be accessible from smaller language models through more evolutionary iterations.

## 6 Discussion and future work

While in this work we focused on molecular optimization task, it is natural to conjecture that our law would be possible to extend to other problem formulations as well. Additionally, it would be interesting to model metrics other than top-1 scores, spreading the applicability of these methods.

We think that an exciting line of work would be to introduce the notion of *compute optimality*, which will require some unification of the oracle and language model computation to optimize against. This would enable us to find the choices for $N$ and $k$, maximizing the performance under some fixed *compute budget*.

As the next steps in this work, we plan on including more challenging and realistic molecular optimization problems to embrace the usefulness of our method. We plan to construct tests to check further extrapolation for a range of different molecular optimization tasks and enhance our understanding of the strengths and limitations of our approach.

## 7 Conclusion

We have developed a parametric law for modeling the scaling dynamics of language model-enhanced evolutionary algorithms. We tested it on the average top-1 performance of multi-property molecular optimization tasks and showed that our law accurately extrapolates in the language model parameter count and number of evolutionary algorithm iterations at the same time. As suggested by our law, we found that smaller models can reach the performance of larger ones by running more evolutionary algorithm steps.

## REFERENCES

Alphacode 2 technical report. URL `https://api.semanticscholar.org/CorpusID:266058988`.

Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. Scaling laws for generative mixed-modal language models. In *International Conference on Machine Learning*, pp. 265–279. PMLR, 2023.

Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A Smith, Dirk Groeneveld, Pang Wei Koh, et al. Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*, 2024.

Nathan Brown, Marco Fiscato, Marwin H.S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59 (3):1096–1108, 2019. doi: 10.1021/acs.jcim.8b00839. URL `https://doi.org/10.1021/acs.jcim.8b00839`. PMID: 30887799.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

Angelica Chen, David Dohan, and David So. Evoprompting: Language models for code-level neural architecture search. *Advances in neural information processing systems*, 36:7787–7817, 2023.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. A simple and provable scaling law for the test-time compute of large language models. *arXiv preprint arXiv:2411.19477*, 2024.

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.

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.

Philipp Guevorguian, Menua Bedrosian, Tigran Fahradyan, Gayane Chilingaryan, Hrant Khachatrian, and Armen Aghajanyan. Small molecule optimization with large language models. *arXiv preprint arXiv:2407.18897*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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. *arXiv preprint arXiv:2309.08532*, 2023.

Jun He, Yu Chen, and Yuren Zhou. A theoretical framework of approximation error analysis of evolutionary algorithms. *arXiv preprint arXiv:1810.11532*, 2018.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=3X2L2TFr0f`.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance in machine translation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=vPOMTkmSiu`.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

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

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=j5BuTrEj35`.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Haorui Wang, Marta Skreta, Cher-Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, et al. Efficient evolutionary search over chemical space with large language models. *arXiv preprint arXiv:2406.16976*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. arxiv 2023. *arXiv preprint arXiv:2309.03409*.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

## A Appendix

### A.1 Derivation of the parametric law

In this section, we provide the derivation of our scaling law for the task performance obtained by evolutionary algorithms powered by a language model.

We frame a task to be solved as a minimization problem given by

$$S^* = \min_x \mathbb{S}(x)$$

where $\mathbb{S}$ is the objective function and refers to the metric of interest. We examine evolutionary algorithms augmented by transformer-based language models $\overline{f}_{N,D}$ with $N$ number parameters, trained with $D$ number of tokens (terminology adopted from Hoffmann et al. (2022)). We seek to find the expected value of the best task performance achieved by running the evolutionary algorithm for $k$ iterations with a language model $\overline{f}_{N,D}$, and denote it by $S^k_{\overline{f}_{N,D}}$, more rigorously

$$S^k_{\overline{f}_{N,D}} = \mathop{\mathbb{E}}_{X^k}[\min(\mathbb{S}(x) : x \in X^k)]$$

where $X^k$ denotes the population after $k$th iteration

We write the expected performance in the following form

$$S^k_{\overline{f}_{N,D}} = S^* + (S^k_{\overline{f}_{N,D}} - S^*)$$

where $S^*$ is the minimum achievable performance and the $(S^k_{\overline{f}_{N,D}} - S^*)$ term refers to the approximation error of the evolutionary algorithm. He et al. (2018) developed a theoretical framework for approximation error analysis for a class of evolutionary algorithms and proved that the error is upper bounded by $e_0 e_1^k$ for $k$th iteration, where $e_0$ and $e_1$ are some constants. We approximate the $(S^k_{\overline{f}_{N,D}} - S^*)$ term in accordance with the theoretical bound and obtain the following expression.

$$S^k_{\overline{f}_{N,D}} = A + \frac{B}{(\beta N^\alpha)^{k^\gamma}}$$

where $A, B, \alpha, \beta$ and $\gamma$ are parameters to estimate. We added the $\gamma$ in the exponent of $k$ for an additional degree of freedom, and we note that without it, the model failed to capture the empirical observations with high accuracy.

Finally, as many task metrics are in the $[0, 1]$ range and require to maximize performance, our expression can be easily converted to model these cases by subtracting our expression from $1$

$$1 - A - \frac{B}{(\beta N^\alpha)^{k^\gamma}}$$

### A.2 Smaller Llama 3 models.

Table 1 shows the architecture parameters used for obtaining smaller models corresponding to Llama 3 architecture.

| model size | $d_{model}$ | $n_{layers}$ | $n_{heads}$ | $n_{kv\_heads}$ |
|---|---|---|---|---|
| 170M | 768 | 8 | 16 | 8 |
| 380M | 1024 | 16 | 16 | 8 |
| 750M | 1536 | 16 | 24 | 8 |

Table 1: Architecture configurations used for smaller versions of Llama 3 architecture.