Merging Enables Refinement of Generative Ecosystems: Building an Ecosystem, Not a Monolith

Alignment Lab AI

November 23, 2024

Abstract

We present a systematic approach to distributed language model development through continuous evolutionary merging. The framework enables sustainable improvement of model capabilities while reducing computational overhead and eliminating centralized bottlenecks. By implementing a comprehensive evaluation system with cryptocurrency-based incentives, we demonstrate that collaborative refinement through strategic merging can achieve state-of-the-art performance with minimal computational resources. Our results show sustained improvement through hundreds of successful merges, suggesting a viable path toward scalable, distributed AI development.

1 System Architecture Overview

The goal heuristic is to enable the public to continuously pursue their own use cases, whether in development or in the pursuit of rewarded tokens. As a consequence of this system, we aim to create a continuously improving performance baseline that ideally outpaces the ability of closed markets to maintain the state of the art through secrecy, scale, and funding alone. We believe this approach ensures that the alignment of model performance, usage, and behaviors remains oriented toward the public interest.

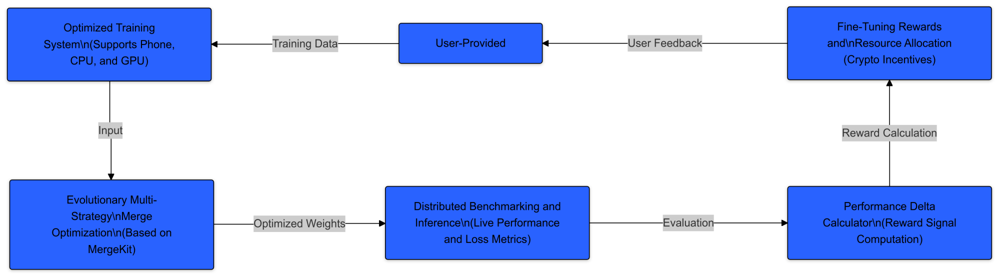
Optimized Training System\n(Supports Phone, CPU, and GPU)
Training Data
User-Provided
User Feedback
Fine-Tuning Rewards and\nResource Allocation (Crypto Incentives)
Input
Reward Calculation
Evolutionary Multi-Strategy\nMerge Optimization\n(Based on MergeKit)
Optimized Weights
Distributed Benchmarking and Inference\n(Live Performance and Loss Metrics)
Evaluation
Performance Delta Calculator\n(Reward Signal Computation)

Figure 1: The Performance Loop

1.1 Unified Data Processing

By assigning a single start/stop token of `\n{` and `}\n`, we ensure efficient pad-free training and fine-tuning. This approach provides consistent, high-quality structured output compatibility and generalization to the keys of the structured syntax. As a result, task-specific behaviors and semantically guided instructions are mutually able to steer and be steered by the format.

This approach is particularly useful for reward modeling, labeling, classification, and content extraction. It enables the base model to function as a versatile multitool, leveraging any JSON-structured data for meaningful improvement. At the same time, it avoids trade-offs traditionally associated with structured outputs, such as: - Increased computational cost and complexity caused by padding and parsing during training. - Performance degradation from reliance on chat-style templates. - Complexity for end users in generating and interpreting structured data outputs.

An additional advantage of this system is its ability to predict responses to user prompts as completions of structured rows with prepopulated labels. For example, the model can efficiently predict a high-reward response or analyze sentiment within the same JSON structure. This capability allows seamless integration with tasks requiring structured inputs and outputs.

The system eliminates the need for complex data preprocessing while maintaining natural sequence boundaries and enabling zero-shot generalization across formats. This streamlined processing method ensures compatibility with a wide range of use cases while significantly reducing overhead.

1.2 Continuous Merging Framework

We propose a continuously improving merging framework that leverages evolutionary optimization principles and heuristic-based strategy selection to dynamically improve models. The framework operates asynchronously, enabling large-scale distributed evaluations and adaptive parameter updates.

Algorithm 1 Async System for Iterative 0th Order Optimization

```
Require: Base parameters $\theta_{\text{base}}$, fine-tuned models $\{\theta_f^i\}_{i=1}^n$, strategy set $\mathcal{S}$
Ensure: Optimized parameters $\theta_{\text{optimized}}$, updated heuristic $\mathcal{H}(s)$
  Initialize $\theta_{\text{current}} \leftarrow \theta_{\text{base}}$
  Initialize $\mathcal{H}(s) \leftarrow$ uniform distribution over $\mathcal{S}$
  Generate initial population $\mathcal{X}_0 \leftarrow \{\mathcal{M}(\theta_{\text{base}}, \theta_f^i, s) \mid \forall \theta_f^i \in \{\theta_f^i\}_{i=1}^n, s \in \mathcal{S}\}$
  Evaluate current performance $S_{\text{current}} \leftarrow \mathcal{E}(\theta_{\text{current}})$
  for $t = 1$ to $T$ (until convergence) do
      Candidate Generation: Sample $s_t \sim \mathcal{H}(s)$
      Generate candidates $\mathcal{X}_t \leftarrow \{\mathcal{M}(\theta_{\text{current}}, \theta_f, s_t) \mid \theta_f \in \{\theta_f^i\}_{i=1}^n\}$
      Parallel Evaluation: Submit $\mathcal{X}_t$ to async evaluation pool $\mathcal{P}$
      Compute scores $\{S(x) \mid x \in \mathcal{X}_t\}$
      Heuristic Update: Compute $\Delta S = S(x_{\text{new}}) - S_{\text{current}}$
      Update $\mathcal{H}(s) \leftarrow \mathcal{H}(s) \cdot \exp(\beta \cdot \Delta S)$
      Normalize $\mathcal{H}(s)$
      Population Refinement: Select top candidates and apply mutation:
      $\theta_{\text{new}} \leftarrow \theta_{\text{best}} + \mathcal{N}(0, \sigma)$
      if $S(\theta_{\text{new}}) > S_{\text{current}}$ then
          $\theta_{\text{current}} \leftarrow \theta_{\text{new}}$
          $S_{\text{current}} \leftarrow S(\theta_{\text{new}})$
      end if
  end for
  return $\theta_{\text{current}}, \mathcal{H}(s)$
```

Formally we can define the process as follows:

$$\Theta_m = \mathcal{M}(\Theta_b, \{\Theta_f^i\}_{i=1}^n, s) \tag{1}$$

where Θ_m represents the merged parameters, Θ_b the base parameters, $\{\Theta_f^i\}_{i=1}^n$ the fine-tuned parameters, and $s \in \mathcal{S}$ the selected merging strategy. The merging function $\mathcal{M}$ dynamically adapts based on heuristics and evolves over iterations to optimize performance metrics.

By coupling heuristic updates with evolutionary optimization, this framework supports continuous model improvement, balances exploration and exploitation of strategies, and ensures scalability in distributed environments.

To address parameter interference, we implement a multi-stage resolution process:

$$\theta_r = \begin{cases} \theta_b & \text{if } \Delta\theta < \epsilon, \\ \text{sign}(\theta_c) \cdot \text{mean}(|\theta_k|) & \text{if consensus is achieved,} \\ \text{OptimizationStep}(\theta_k) & \text{otherwise,} \end{cases} \tag{2}$$

where $\Delta\theta$ represents parameter change magnitude, ϵ is the sensitivity threshold, θ_b is the base parameter, θ_c is the consensus parameter, and θ_k are the candidate parameters.

2 Performance Evaluation Framework

The system implements a comprehensive evaluation framework that quantifies improvements across multiple dimensions while maintaining computational efficiency. This framework provides both the foundation for the reward system and the continuous optimization process.

2.1 General Improvement Quantification

Given a task set τ from the Dharma benchmark distribution with associated weights w_i, performance improvement is quantified through four complementary metrics that capture different aspects of model capability:

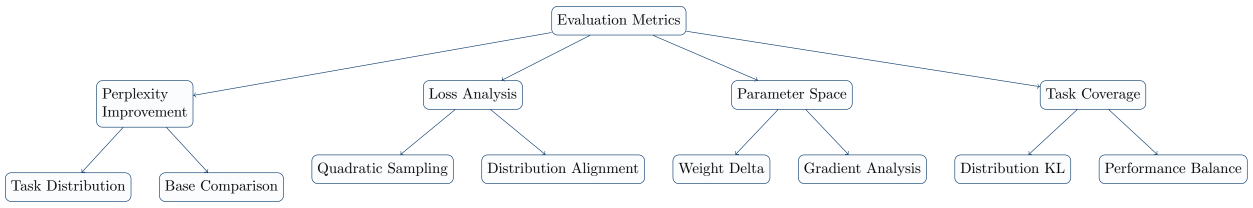

Figure 2: Evaluation Framework Components

2.1.1 Perplexity-Based Evaluation

For n datapoints across distribution τ:

$$P = \sigma\left(\sum_i w_i \cdot \frac{\text{PPL}_{b,\tau_i} - \text{PPL}_{f,\tau_i}}{\text{PPL}_{b,\tau_i}}\right) \tag{3}$$

with perplexity defined:

$$\mathrm{PPL}_{x,\tau_i} = \exp\left(\frac{1}{n}\sum_{j=1}^{n} -\log(p_{x,\tau_i}(y_j|x_j))\right), \tag{4}$$

where PPL_{b,τ_i} and PPL_{f,τ_i} represent perplexities of the base and fine-tuned models, respectively.

2.1.2 Loss Distribution Analysis

Across n^2 benchmark datapoints:

$$L = \sigma\left(\sum_i w_i \cdot \frac{L_{b,\tau_i} - L_{f,\tau_i}}{L_{b,\tau_i}}\right) \tag{5}$$

where task-specific loss is:

$$L_{x,\tau_i} = \frac{1}{n^2}\sum_{j=1}^{n^2} \mathrm{loss}(f_{x,\tau_i}(x_j), y_j). \tag{6}$$

Quadratic sampling ensures robust evaluation across task distributions.

2.1.3 Parameter Space Metrics

The weight delta measure quantifies parameter space movement while accounting for relative magnitude:

$$W = 1 - \sigma\left(\frac{\|\theta_f - \theta_b\|_2}{\|\theta_b\|_2}\right), \tag{7}$$

where $\|\theta_f - \theta_b\|_2$ represents the L_2-norm difference between fine-tuned and base parameters.

2.1.4 Task Distribution Alignment

Performance distribution alignment is measured through KL divergence:

$$D = 1 - \mathrm{KL}(p_\tau \| q_\tau), \tag{8}$$

where p_τ represents the target task distribution and q_τ the achieved distribution.

2.2 Integrated Performance Score

The system combines all evaluation metrics through a weighted performance score:

$$S = \alpha_1 P + \alpha_2 L + \alpha_3 W + \alpha_4 D, \tag{9}$$

subject to the constraints:

$$\begin{aligned} &\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1, \\ &\alpha_i \in [0,1] \quad \forall i \in \{1,2,3,4\}, \\ &S \in [0,1]. \end{aligned}$$

2.3 Reward Distribution System

The reward mechanism implements time-weighted scaling to incentivize both rapid improvements and sustained performance gains:

$$R = \begin{cases} R_b \cdot (e^{\beta t S} - 1), & \text{if } S > \text{threshold}, \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

where:

- R_b: Base reward,
- β: Reward scaling coefficient,
- t: Temporal weighting factor,
- threshold: Minimum improvement requirement.

The distribution maintains fairness through weight normalization:

$$\sum_i w_i = 1. \tag{11}$$

3 Evolutionary Strategy Optimization

The system implements continuous optimization over merge strategy selection through gradient-based heuristic evolution. This process adapts merge strategies based on empirical performance data:

$$\mathcal{H}(s, t) = \nabla_s \sum_{i=1}^{n} \text{Perf}_i(M(s, t)), \tag{12}$$

where $M(s, t)$ represents the merge outcome with strategy s at iteration t.

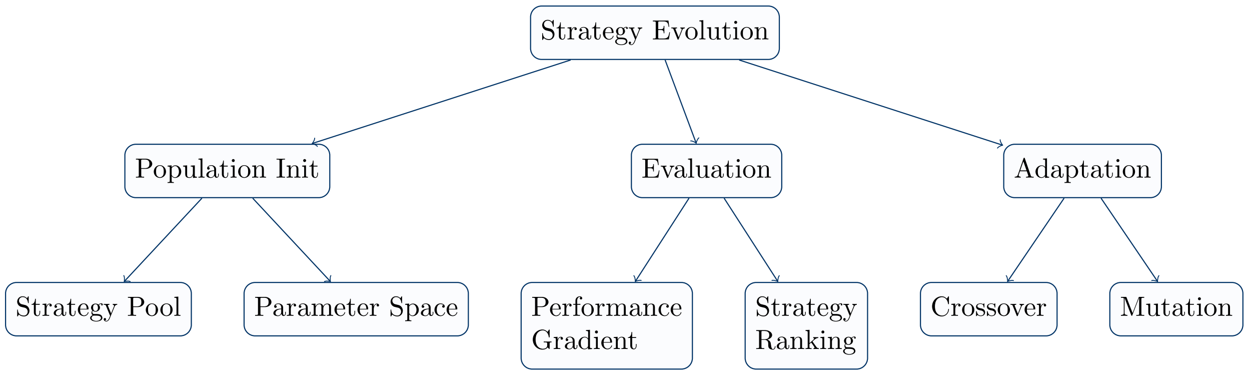

Figure 3: Strategy Evolution Process

The evolution process follows the algorithm:

Algorithm 2 Continuous Merge Strategy Optimization

```
1: Initialize strategy population $S = \{s_1, \dots, s_k\}$
2: while not converged do
3:     Evaluate $\mathcal{H}(s, t)$ for each $s \in S$
4:     Rank strategies by performance gradient
5:     $S_{\text{next}} \leftarrow \text{TopK}(S)$
6:     Generate new strategies via:
           $s_{\text{new}} = \alpha s_i + (1 - \alpha) s_j + \epsilon$    (13)
7:     where $s_i, s_j \in S_{\text{next}}$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$
8:     Update population: $S \leftarrow S_{\text{next}} \cup \{s_{\text{new}}\}$
9: end while
```

4 System Integration and Deployment

The system architecture implements comprehensive integration of all components through bidirectional optimization pathways that enable continuous improvement while maintaining system stability.

4.1 Integration Pipeline

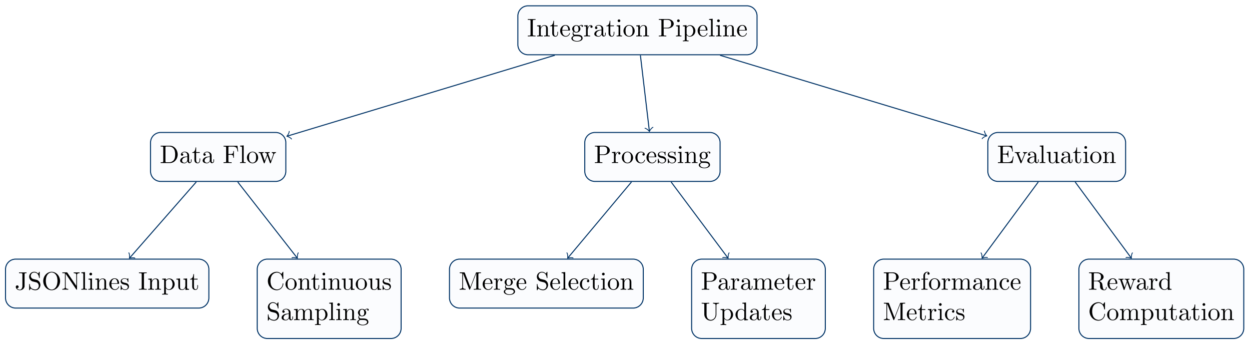

Figure 4: Integration Pipeline Components

The contribution pipeline evaluates task impact through standardized metrics:

$$\text{Task}_{\text{effect}} = \frac{\Delta\text{Performance}}{\Delta\text{Parameters}} \cdot \text{Coverage}_{\text{factor}} \tag{14}$$

where $\text{Coverage}_{\text{factor}}$ represents task diversity contribution.

4.2 Distributed Evaluation Framework

The Petals-based distributed inference system Borzunov et al. [2023] enables scalable evaluation across heterogeneous compute resources. This decentralized approach allows us to leverage a network of devices for continuous benchmarking and performance tracking, providing a transparent and dynamic measure of the ecosystem's progress. By continuously evaluating the core model and merged contributions on the Dharma benchmark, we generate a live feed of performance improvements, visualized as a publicly accessible line graph. This graph transparently displays the impact of individual contributions and provides a compelling visualization of the collective progress driven by the ecosystem.

4.2.1 Real-time Performance Tracking and Visualization

The distributed evaluation system continuously benchmarks the model on randomly sampled rows from Hugging Face datasets, ensuring a broad and unbiased assessment of its capabilities. This ongoing evaluation provides a real-time performance feed that is publicly accessible, allowing for transparent monitoring of the model's evolution. The performance data is visualized as a line graph, plotting performance metrics (e.g., average Dharma score) over time. This visualization clearly demonstrates the value of contributions and provides a compelling representation of the ecosystem's collective progress.

4.2.2 Reward Calculation and Distribution

The reward system is directly tied to the performance improvements measured by the distributed evaluation framework. When a contributor submits a merged model, it undergoes rigorous evaluation on the Dharma benchmark using the Petals network. The resulting performance score (S), calculated as described below, determines the reward allocated to the contributor.

Automatic General Improvement Score for Reward Distribution Let τ be the set of tasks from the Dharma benchmark distribution Pharaouk [2023], where each task τ_i has an associated weight w_i such that $\sum_i w_i = 1$. The performance improvement is quantified using the following metrics:

1. **Perplexity Improvement (P):** Measured across n datapoints from the Dharma distribution:

$$P = \sigma\left(\sum_i w_i \cdot \frac{\text{PPL}_{\text{base},\tau_i} - \text{PPL}_{\text{merged},\tau_i}}{\text{PPL}_{\text{base},\tau_i}}\right)$$

   where PPL_{x,τ_i} represents the perplexity of model x on task τ_i.

2. **Loss Delta (L):** Calculated across n^2 benchmark datapoints:

$$L = \sigma\left(\sum_i w_i \cdot \frac{L_{\text{base},\tau_i} - L_{\text{merged},\tau_i}}{L_{\text{base},\tau_i}}\right)$$

   where L_{x,τ_i} represents the loss of model x on task τ_i. The quadratic sampling (n^2) enhances robustness.

3. **Weight Delta (W):** Measures the relative change in model parameters:

$$W = 1 - \sigma\left(\frac{\|\theta_{\text{merged}} - \theta_{\text{base}}\|_2}{\|\theta_{\text{base}}\|_2}\right)$$

   This metric incentivizes efficient parameter updates.

4. **Task Performance Distribution Alignment (D):** Assesses the alignment between the model's performance distribution and the target Dharma distribution:

$$D = 1 - \mathrm{KL}(p_\tau || q_\tau)$$

where p_τ is the target task distribution and q_τ is the achieved performance distribution.

These metrics are combined into a final performance score:

$$S = \alpha_1 P + \alpha_2 L + \alpha_3 W + \alpha_4 D$$

where $\sum_{i=1}^{4} \alpha_i = 1$ and $\alpha_i \in [0, 1]$. The weights α_i can be adjusted to prioritize specific aspects of performance. The sigmoid function $\sigma(x) = \frac{1}{1+e^{-kx}}$ is used for normalization, where k is a scaling factor.

The reward R is then calculated as:

$$R = \begin{cases} R_{\text{base}} \cdot (e^{\beta t S} - 1) & \text{if } S > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

where R_{base} is the base reward, β is a reward scaling factor, t is an optional time factor, and threshold is the minimum performance improvement required to receive a reward. This formula ensures that rewards are proportional to the measured performance improvement and can optionally incorporate a time-decay factor to incentivize rapid progress.

5 Empirical Results, Analysis, and Extrapolative Implications

The foundation of our proposed ecosystem rests upon the empirical success of collaborative model development, particularly exemplified by the ARCO project Appvoid Research Group [2023]. ARCO's sustained performance improvement through 500+ merges, surpassing typical limitations of traditional training paradigms, provides compelling evidence for the viability and efficacy of continuous, merge-based refinement. Achieving state-of-the-art results with a relatively small model (0.5B parameters) further underscores the efficiency gains achievable through this decentralized approach. This observation aligns with findings in Lee et al. [2023] demonstrating efficient performance scaling with minimal resource investment, challenging the prevailing notion that model scale is the sole determinant of performance.

5.1 Flexibility and Performance of Merging across Domains

Adaptability is crucial for building a robust and versatile ecosystem capable of addressing a wide range of real-world applications.

* **Multimodal Merging:** The work of Sung et al. [2023] demonstrates that merging models trained on different modalities (e.g., vision and language) can create a unified model with enhanced performance across all base modalities. This aligns with our vision of a synergistic ecosystem where specialized models contribute to a more capable and versatile core model.

* **Multitask Merging:** Techniques like TIES-Merging Yadav et al. [2023] and MaTS Tam et al. [2023] address the critical challenge of interference in multitask merging, enabling the successful combination of models specialized for different tasks. These advanced merging strategies are essential for preserving and enhancing performance across a diverse range of tasks within our ecosystem. Furthermore, algorithms based on task indices, as explored in Tam et al. [2023], provide a structured approach to matching and merging models based on their task specialization.

* **Multilingual Merging:** MergeDistill Khanuja et al. [2021] demonstrates the potential of merging in a multilingual context, enabling the creation of models that outperform their monolingual teachers while maintaining a fixed capacity. This suggests that merging can effectively combine linguistic knowledge from diverse sources, creating more robust and adaptable multilingual models.

5.2 GRIT Architecture and Ecosystem Compatibility

Our chosen base model, built upon the GRIT architecture Muennighoff [2024], is uniquely suited for this collaborative ecosystem. GRIT's ability to handle both generative and embedding tasks, through its unified encoder-decoder structure and integrated instruction following, provides a versatile foundation for diverse downstream applications. The incorporation of rope embeddings further enhances the model's capacity to process long sequences and handle multimodal data effectively. Moreover, the model's derivation from Mistral ensures compatibility with state-of-the-art optimizations for scaling and efficiency, enabling seamless integration with distributed training and inference frameworks like Petals Borzunov et al. [2023]. This compatibility is crucial for maximizing resource utilization and ensuring the scalability of our ecosystem. The Dharma benchmark Pharaouk [2023], with its focus on out-of-distribution task evaluation, further strengthens the robustness and generalizability of models within our ecosystem. ColD Fusion Don-Yehiya et al. [2022] provides a conceptual foundation for our collaborative, distributed training approach, highlighting the potential for synergistic improvement through the integration of diverse contributions.

5.3 Scaling Characteristics and Resource Efficiency

By distributing the computational burden across a network of contributors, we circumvent the escalating costs associated with training ever-larger models while sidestepping the tradeoffs of traiditional distributed training systems entirely, allowing for collaborative consensus based optimization of the model based purely on participation in the work of pursuing your own use cases in the open source The continuous merging process allows efficient integration of incremental improvements without requiring massive retraining. We hypothesize that performance gain scales logarithmically with merge complexity, augmented by a term representing the diversity of contributions Don-Yehiya et al. [2022]:

$$\text{Gain}(n) = \alpha \log(n) + \beta \sqrt{\text{Diversity}(n)}$$

where n is the number of merges. Resource efficiency, we posit, improves exponentially over time, modulated by the initial quality of the base model:

$$\text{Efficiency}(t) = \text{Efficiency}_0 \cdot (1 + \delta)^t \cdot \text{Quality}_{\text{base}}$$

where t represents time. These hypothesized relationships, while requiring further empirical validation, suggest a promising trajectory of sustainable and scalable growth for our ecosystem.

6 Conclusion: An ecosystem

By empowering a distributed network of contributors, we aim to transcend the limitations of the traditional monolithic training paradigm. Our system, grounded in the principles of open collaboration, distributed computation, and collective intelligence drivinh continuous improvement with no burden on the end users beyond their personal participation in building useful models for their own use cases. The integration of a robust merging pipeline, comprehensive evaluation metrics, and a transparent reward mechanism ensures that the ecosystem remains sustainable, scalable, and aligned with the public interest.

The empirical success of projects like ARCO Appvoid Research Group [2023], demonstrating sustained performance gains through hundreds of merges, provides strong evidence for the viability of our approach. Furthermore, the flexibility of merging, demonstrated across multimodal, multitask, and multilingual domains Sung et al. [2023], Yadav et al. [2023], Khanuja et al. [2021], underscores the adaptability and broad applicability of our framework. The strategic choice of a GRIT-based architecture Muennighoff [2024], with its inherent versatility and compatibility with existing ecosystems, further strengthens our foundation for collaborative development.

We envision a future where the development of artificial intelligence is no longer confined to the walls of isolated institutions, where the ecosystem passively produces communal intelligence at such a scale that the most powerful synthetic intelligences are publicly known, well understood, and accessible to all of humanity.

References

Appvoid Research Group. ARCO: Continuous model improvement through merging. `https://huggingface.co/appvoid/arco`, 2023.

Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin Raffel. Distributed inference and fine-tuning of large language models over the internet. *arXiv preprint arXiv:2312.08361*, 2023. URL `https://arxiv.org/abs/2312.08361`.

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. ColD fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint arXiv:2212.01378*, 2022. URL `https://arxiv.org/abs/2212.01378`.

Simran Khanuja, Melvin Johnson, and Partha Talukdar. Mergedistill: Merging pre-trained language models using distillation. *arXiv preprint arXiv:2106.02834*, 2021. URL `https://arxiv.org/abs/2106.02834`.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of LLMs. *arXiv preprint arXiv:2308.07317*, 2023. URL `https://arxiv.org/abs/2308.07317`.

Niklas Muennighoff. Generative representational instruction tuning (GRIT). *arXiv preprint arXiv:2402.09906*, 2024. URL `https://arxiv.org/abs/2402.09906`.

Pharaouk. Dharma: A benchmark suite for out-of-distribution task evaluation. `https://github.com/pharaouk/dharma`, 2023.

Yi-Lin Sung, Linjie Li, Kevin Lin, Zhe Gan, Mohit Bansal, and Lijuan Wang. An empirical study of multimodal model merging. *arXiv preprint arXiv:2304.14933*, 2023. URL `https://arxiv.org/abs/2304.14933`.

Derek Tam, Mohit Bansal, and Colin Raffel. Merging by matching models in task parameter subspaces (MaTS). *arXiv preprint arXiv:2312.04339*, 2023. URL `https://arxiv.org/abs/2312.04339`.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 2023. URL `https://arxiv.org/abs/2306.01708`.