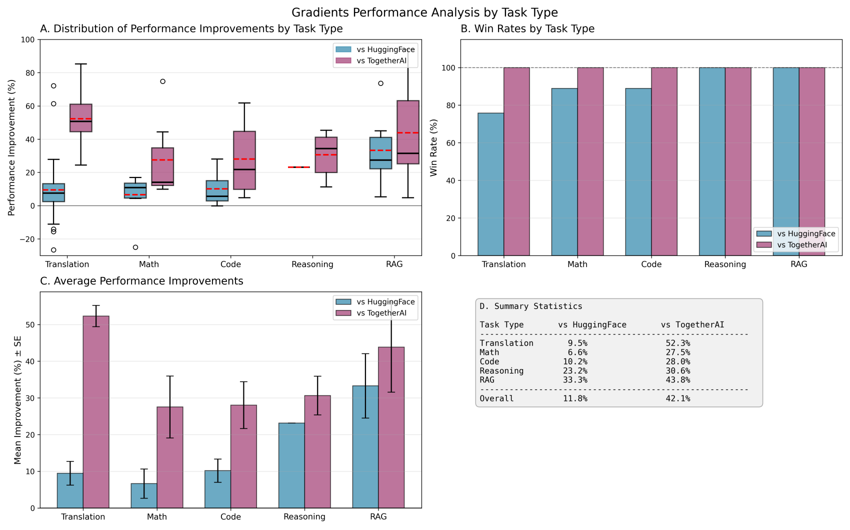
Gradients: When Markets Meet Fine-tuning - A Distributed Approach to Model Optimisation

Rayonlabs Team
Rayonlabs.ai
`contact@rayonlabs.ai`

Abstract

Foundation model fine-tuning faces a fundamental challenge: existing AutoML platforms rely on single optimisation strategies that explore only a fraction of viable hyperparameter configurations. In this white paper, We introduce Gradients, a decentralised AutoML platform that transforms hyperparameter optimisation into a competitive marketplace where independent miners compete to discover optimal configurations. Economic incentives align individual exploration with collective optimisation goals, driving systematic investigation of hyperparameter regions that centralised methods miss. We evaluate our approach across 180 controlled experiments spanning diverse model architectures (70M to 70B parameters) and task types. Gradients achieves an 82.8% win rate against HuggingFace AutoTrain and 100% against TogetherAI, Databricks, and Google Cloud, with mean improvements of 11.8% and 42.1% respectively. Complex reasoning and retrieval tasks show particularly strong gains of 30-40%, whilst diffusion models achieve 23.4% improvements for person-specific generation. These results demonstrate that competitive, economically-driven approaches can systematically discover superior configurations that centralised AutoML consistently miss.



Task Type	vs HuggingFace	vs TogetherAI
Translation	9.5%	52.3%
Math	6.6%	27.5%
Code	10.2%	28.0%
Reasoning	23.2%	30.6%
RAG	33.3%	43.8%
Overall	11.8%	42.1%

Figure 1: Statistical analysis by task type. Four-panel analysis: box plots show improvement distributions, bar charts display win rates and mean improvements with standard errors, summary table provides exact values.

1 The open challenges in model fine-tuning.

Foundation model fine-tuning involves optimisation across multiple interdependent dimensions: task type, model architecture, dataset size, and time constraints [41]. A 70M parameter model fine-tuned for mathematical reasoning with limited data requires different strategies than a 70B model trained for translation tasks with extensive datasets. Time constraints further complicate the picture—optimal configurations for 2-hour training windows differ from those for 8-hour allocations. Current AutoML platforms [37, 13] produce results across these diverse scenarios, yet our empirical evaluation across 180 experiments reveals systematic performance gaps, suggesting that task-specific optimisation opportunities remain undiscovered.

The challenge lies in creating systems that can adapt optimisation strategies to these multifaceted problem characteristics. Economic incentives offer a solution [11, 8]: when optimisation success directly determines rewards, agents develop approaches tailored to the specific combination of task type, model scale, dataset properties, and resource constraints.

Such incentive alignment naturally leads to decentralised approaches [24, 17]. By distributing the exploration task across multiple independent agents, each with unique strategies and computational resources, decentralised systems can simultaneously investigate diverse approaches that would otherwise remain unexplored. This parallel search capability scales with the number of participants, creating an explorative capacity that exceeds what centralised systems can achieve with equivalent resources. However, mere distribution of computation is insufficient; the key innovation lies in establishing economic incentives that align individual agent behaviour with collective optimisation goals. When agent rewards are directly tied to discovering superior configurations, the system naturally evolves toward increasingly sophisticated optimisation strategies [20]. This evolutionary pressure creates a form of collective intelligence that continuously adapts to the specific characteristics of each fine-tuning task, rather than applying standardised approaches. The theoretical foundation of this approach combines distributed optimisation [23], empirical risk minimisation, and tournament theory [26], establishing a framework where properly structured competition can efficiently navigate complex optimisation landscapes that resist traditional methods.

In this paper, we present Gradients, a decentralised AutoML platform built on the Bittensor network [32] that implements these theoretical principles to address the fine-tuning challenges of foundation models. We make several contributions: First, we formulate a mathematical scoring framework that effectively ranks model submissions based on their performance across both test and synthetic evaluation datasets, with appropriate mechanisms to prevent overfitting and ensure fair competition. Second, we develop an economic incentive structure that aligns individual miner rewards with overall optimisation quality, creating a self-reinforcing cycle of innovation in hyperparameter exploration. Third, we present empirical results from extensive comparative testing against leading centralised AutoML platforms, demonstrating consistent and significant performance improvements across diverse model types and application domains. Fourth, we analyse the dynamics of the system, showing how the competitive marketplace accelerates the adoption of novel optimisation techniques and enables more efficient allocation of computational resources compared to conventional approaches. Through these contributions, we demonstrate the practical effectiveness of decentralised, economically-driven approaches to hyperparameter optimisation and provide empirical evidence for their advantages, with particular application to the fine-tuning of Large Language Models and diffusion models.

2 Related Work

Hyperparameter Optimisation and AutoML. The field of automated hyperparameter optimisation has progressed from simple grid and random search methods to sophisticated multi-fidelity approaches. Bayesian optimisation frameworks like Hyperband [23] and BOHB [12] combine principled uncertainty estimation with early stopping mechanisms to efficiently navigate large hyperparameter spaces. These methods achieve computational efficiency through bandit-based resource allocation, but remain fundamentally centralised in their coordination mechanisms. Recent AutoML platforms including AutoKeras [21] and Auto-Sklearn [13] have democratised access to these techniques, yet they still rely on single optimisation strategies that may miss task-specific optima. HuggingFace Auto-Train [37] represents the current state-of-the-art in no-code model fine-tuning, providing automated hyperparameter optimisation across multiple modalities but using centralised search strategies.

Distributed and Competitive Optimisation. Population-based training (PBT) [20] introduced competitive dynamics to neural network optimisation, where populations of models compete and evolve hyperparameters during training. This approach demonstrates how competition discovers superior configurations through exploitation and exploration mechanisms reminiscent of evolutionary algorithms [10]. Parallel to this, federated learning [24] established principles for coordinating distributed model training without centralising data, achieving communication efficiency through local computation and periodic aggregation. However, these approaches lack economic incentive mechanisms to align individual participant behaviour with collective optimisation goals.

Parameter-Efficient Fine-tuning. The emergence of foundation models has necessitated new approaches to efficient adaptation. LoRA [19] and its variants inject trainable low-rank matrices into pre-trained transformers, reducing trainable parameters by orders of magnitude whilst preserving performance. AdaLoRA [42] extends this approach with adaptive rank allocation, whilst QLoRA [7] combines quantisation with low-rank adaptation for memory efficiency. These parameter-efficient methods are widely used by AutoML LLM and diffusion training platforms [9].

Economic Mechanisms in Machine Learning. Recent work explores how economic incentives coordinate machine learning systems. Differentiable economics [11] employs neural networks to automatically design optimal auction mechanisms, demonstrating how learning systems embed economic principles. Blockchain-based coordination mechanisms [17, 34] show promise for incentivising distributed computation, whilst tournament theory provides mathematical foundations for competition-based optimisation [26]. Incentive mechanism design for distributed coded machine learning [8] establishes optimal reward structures for heterogeneous computational contributors. However, researchers have largely left unexplored the application of these economic principles to automated hyperparameter optimisation.

Decentralised Machine Learning Networks. Emerging platforms like Bittensor [32] demonstrate how cryptoeconomic incentives coordinate distributed machine learning tasks. These networks employ token-based rewards to align individual computational contributions with network-wide objectives, creating sustainable ecosystems for collaborative AI development. The Yuma Consensus mechanism [27] provides robust peer-ranking systems resistant to collusion, whilst maintaining decentralised coordination. Our work builds on these foundations by specifically applying economic coordination mechanisms to the hyperparameter optimisation problem, demonstrating how competitive markets discover superior model configurations.

Foundation Model Fine-tuning and Evaluation. Recent advances in fine-tuning large language models focus on instruction following [29], preference optimisation [31], and synthetic data generation [2]. For diffusion models, techniques like DreamBooth [33] enable subject-driven generation through personalised fine-tuning. Community-driven platforms like CivitAI [38] have democratised access to fine-tuned diffusion models, though require an understanding of the diffusion training process or the use of default parameters in order to use. Our work addresses this gap by providing economic incentives for discovering optimal fine-tuning configurations across diverse model types and tasks without the need for users of the platform to have a deep understanding of the fine-tuning process.

3 Gradients

The primary purpose of Gradients is to serve real users with high-quality fine-tuning solutions for practical applications. Unlike traditional AutoML systems that rely on a single optimisation strategy or limited search methods, our approach transforms fine-tuning into a competitive tournament where participants are rewarded based on the quality of their solutions relative to others. This approach enables exploration of a much broader solution space to better meet genuine user needs, while supplementary synthetic benchmark tasks help maintain system quality and miner skill development.

3.1 System Architecture

The Gradients system consists of three principal actors and multiple computational processes:

Validators are network nodes responsible for task creation, dataset preparation, and submission evaluation. They maintain the integrity of the system through fair scoring and reward distribution. Crucially, validators also channel organic tasks from real users into the network, ensuring the system solves practical fine-tuning challenges alongside synthetic benchmarks.

Miners are independent computational participants who compete to find optimal fine-tuning configurations. They receive task specifications and submit fine-tuned models for evaluation.

Tasks represent specific fine-tuning challenges, each with defined datasets, model architectures, and evaluation criteria. Tasks may be organic (derived from real user needs) or synthetic (created for systematic benchmarking).

The system operates through a continuous cycle of task creation, miner assignment, model training, and evaluation, with economic rewards distributed according to performance.

The core idea of Gradients lies in its transformation of hyperparameter optimisation into a competitive marketplace, creating three key advantages. First, independent miners employ diverse approaches to the same optimisation problem, ensuring strategy diversity across the network. Second, multiple regions of hyperparameter space are explored simultaneously through parallel exploration, maximising coverage of potential solutions. Third, competitive rewards drive continuous refinement of optimisation techniques through evolutionary pressure, naturally selecting for the most effective approaches. Miners operate as independent agents with complete freedom in determining their fine-tuning approach, including choices in hyperparameter selection such as learning rates and batch sizes, optimisation techniques like LoRA, QLoRA, or full fine-tuning, hardware allocation and parallelisation strategies, and custom augmentation or preprocessing methods. This autonomy creates natural strategy diversity that evolves through competitive pressure, driving innovation without requiring centralised coordination and ensuring that the most effective optimisation strategies emerge organically through market dynamics.

3.2 Decentralised Task Distribution

The task distribution mechanism ensures fair allocation of fine-tuning opportunities while maintaining incentives for high-quality submissions. Validators create tasks by specifying $\mathcal{T} = \langle M, D, \Delta t, \mathcal{E} \rangle$ where M represents the base model identifier, D the dataset specifications, Δt the allocated time for completion, and $\mathcal{E}$ the evaluation criteria.

The system prioritises organic tasks—those created from real user needs—as these represent the actual problems people need solved. These organic tasks are the primary purpose of the Gradients system and provide real-world validation of fine-tuning techniques. To supplement the organic workload and maintain consistent network activity, the system also creates synthetic tasks through a controlled distribution mechanism. Synthetic tasks provide consistent benchmarks for miner skill development, ensure comprehensive coverage of model types and domains, and maintain network activity during periods of lower organic demand.

Tasks fall into four technical categories:

Instruction Tuning Tasks for training language models on instruction-following datasets.
Direct Preference Optimisation (DPO) Tasks for preference-based optimisation of language models.
Group Relative Policy Optimisation (GRPO) Tasks for advanced preference-based training.

Image Generation Tasks for fine-tuning diffusion models.

Tasks are created with a controlled distribution across categories to ensure balanced coverage:

$$P(\text{TaskType} = t) = \begin{cases} \rho_{\text{instruct}} & \text{if } t = \text{InstructTextTask} \\ \rho_{\text{dpo}} & \text{if } t = \text{DPOTask} \\ \rho_{\text{grpo}} & \text{if } t = \text{GRPOTask} \\ \rho_{\text{image}} & \text{if } t = \text{ImageTask} \end{cases} \tag{1}$$

with parameters $\rho_{\text{instruct}} = 0.25$, $\rho_{\text{dpo}} = 0.1$, $\rho_{\text{grpo}} = 0.3$, and $\rho_{\text{image}} = 0.35$, at the time of writing this with these weightings being adjusted to complement the organic job proportions.

For each task, the system selects a pool of miners through a weighted probability mechanism designed to balance exploration and exploitation. Given a set of available miners $\mathcal{M} = \{m_1, m_2, ..., m_n\}$, selection weights are assigned as $w_i = \alpha$ if miner m_i has not participated today, or $w_i = \max(s_i, \gamma)$ otherwise, where α is the default score for first daily participation (set to 2.0), s_i is miner m_i's average quality score from previous tasks, and γ is the minimum score threshold (0.01). Miners are then sorted by weight and assigned position-based selection probabilities:

$$p_i = \lambda - (i - 1) \cdot \frac{\lambda - 1}{|\mathcal{M}|} \tag{2}$$

where p_i is the relative probability of selecting the i-th ranked miner, λ is the top miner chance multiplier (3.0), and $|\mathcal{M}|$ is the total number of available miners. This approach preferentially selects high-performing miners while ensuring all miners maintain participation opportunities, preventing monopolisation by established participants.

3.3 Dataset Preparation Pipeline

Effective dataset preparation ensures fair evaluation and meaningful comparison of fine-tuning approaches. For each task, the system partitions the available data into three distinct sets: $D = D_{\text{train}} \cup D_{\text{test}} \cup D_{\text{synth}}$ where D_{train} is provided to miners for model fine-tuning, D_{test} is used for primary performance evaluation with size $|D_{\text{test}}| = \min(|D| \cdot \rho_{\text{test}}, \kappa_{\text{test}})$, and D_{synth} is algorithmically generated to test generalization with size $|D_{\text{synth}}| = \min(|D| \cdot \rho_{\text{synth}}, \kappa_{\text{synth}})$, using parameters $\rho_{\text{test}} = 0.1$, $\rho_{\text{synth}} = 1.0$, $\kappa_{\text{test}} = 1{,}000$, and $\kappa_{\text{synth}} = 300$. Synthetic data generation serves multiple purposes: it provides an additional evaluation benchmark, tests for generalisation capabilities, and mitigates overfitting to the test set. For language models, using a controlled process with temperature $\tau = 0.6$, we generate synthetic data that preserves schema requirements while introducing semantic diversity through the process $x'_i = G(x_i, \tau)$ for all $x_i \in D_{\text{sample}}$, where G is a high-quality foundation model generating function. For DPO tasks, this includes generating both chosen and rejected responses using models of different capabilities. For diffusion models, using temperature $\tau = 0.4$, the system generates new image-text pairs $(I_i, P_i) = G_{\text{image}}(S_i, R_i)$ with controlled style variations and prompt diversity, where S_i is a style specification and R_i is a target resolution. Image resolutions are standardised to ensure fair comparison, with dimensions constrained to be divisible by 64.

3.3.1 Task Time Allocation

The allocated time for task completion scales with both dataset size and model complexity:

$$\Delta t \in [f_{\min}(|D|, |M|), f_{\max}(|D|, |M|)] \tag{3}$$

For text tasks, time allocation follows a binned approach:

$$\Delta t \in \begin{cases} [3, 6] \text{ hours} & \text{if } |D| \in [10000, 25000] \\ [4, 8] \text{ hours} & \text{if } |D| \in [25000, 50000] \\ [5, 9] \text{ hours} & \text{if } |D| \in [50000, 100000] \\ [7, 10] \text{ hours} & \text{if } |D| \in [100000, 500000] \end{cases} \tag{4}$$

For image tasks, time allocation is generally shorter due to the efficiency of LoRA-based fine-tuning for diffusion models, typically ranging from 1-2 hours regardless of dataset size.

3.4 Multi-Level Evaluation Framework

To ensure fair comparison and prevent gaming, all models are evaluated in isolated Docker environments with standardised GPU configurations. This creates a controlled evaluation sandbox where performance differences reflect genuine optimisation quality rather than environmental variations. For language models, the evaluation process computes both test and synthetic losses:

$$L_{\text{test}}(m_i) = \frac{1}{|D_{\text{test}}|} \sum_{x \in D_{\text{test}}} \ell(M_{m_i}, x) \tag{5}$$

$$L_{\text{synth}}(m_i) = \frac{1}{|D_{\text{synth}}|} \sum_{x \in D_{\text{synth}}} \ell(M_{m_i}, x) \tag{6}$$

where M_{m_i} is the model submitted by miner m_i and ℓ is an appropriate loss function. For diffusion models, evaluation encompasses text-guided and non-text-guided image reconstruction quality:

$$L_{\text{text}}(m_i) = \frac{1}{|D_{\text{test}}|} \sum_{(I,p) \in D_{\text{test}}} \|I - M_{m_i}(I_{\text{noisy}}, p)\|_2^2 \tag{7}$$

$$L_{\text{no-text}}(m_i) = \frac{1}{|D_{\text{test}}|} \sum_{I \in D_{\text{test}}} \|I - M_{m_i}(I_{\text{noisy}}, \emptyset)\|_2^2 \tag{8}$$

where I is the original target image, I_{noisy} is the noise-corrupted input, and p is the text prompt. To prevent overfitting to either evaluation set, performance is measured as a weighted combination $L_{\text{weighted}}(m_i) = \omega \cdot L_{\text{test}}(m_i) + (1 - \omega) \cdot L_{\text{synth}}(m_i)$ where ω is the test score weighting parameter (0.7 for text tasks). For image models, the weighting combines text-guided and non-text-guided metrics: $L_{\text{weighted}}(m_i) = \delta \cdot L_{\text{text}}(m_i) + (1 - \delta) \cdot L_{\text{no-text}}(m_i)$ with $\delta = 0.25$ emphasising coherence over prompt adherence.

3.4.1 Anti-Gaming Mechanisms

To maintain system integrity, several mechanisms detect and penalise attempts to game the evaluation:

Duplicate Detection: Submissions with identical loss patterns are identified, and only the earliest submission is credited:

$$\text{isDuplicate}(m_i, m_j) = \begin{cases} \text{true} & \text{if } |L_{\text{test}}(m_i) - L_{\text{test}}(m_j)| < \epsilon \\ & \text{and } |L_{\text{synth}}(m_i) - L_{\text{synth}}(m_j)| < \epsilon \\ \text{false} & \text{otherwise} \end{cases} \tag{9}$$

Suspicious Performance Detection: Models showing anomalous performance disparities between test and synthetic datasets are flagged:

$$\text{isSuspicious}(m_i) = \begin{cases} \text{true} & \text{if } L_{\text{synth}}(m_i) > L_{\text{test}}(m_i) + \alpha \cdot \sigma(L_{\text{test}}) \\ \text{false} & \text{otherwise} \end{cases} \tag{10}$$

where α is a sensitivity parameter (0.5) and $\sigma(L_{\text{test}})$ is the standard deviation of test losses.

3.5 Miner Scoring

The scoring system translates raw performance metrics into reward signals that drive system behaviour, using a multi-level approach that balances immediate task performance with long-term incentive alignment. Within each task, miners are ranked by their weighted loss, with scores assigned according to:

$$S_{\text{task}}(m_i) = \begin{cases} S_{\text{first}} & \text{if rank}(m_i) = 1 \\ S_{\text{penalty}} & \text{if rank}(m_i) > |\mathcal{M}_{\text{valid}}| \cdot (1 - \rho_{\text{penalty}}) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

with parameters $S_{\text{first}} = 3.0$ (first-place score) and $S_{\text{penalty}} = -1.0$ (penalty score). The bottom 25% of miners by performance receive the penalty score. This creates strong incentives for top performance while discouraging poor-quality submissions that waste network resources.

Raw scores are adjusted based on task complexity to ensure fair comparison across different fine-tuning challenges:

$$S_{\text{adjusted}}(m_i, \mathcal{T}) = S_{\text{task}}(m_i) \cdot W_{\text{task}}(\mathcal{T}) \tag{12}$$

where the task weight function $W_{\text{task}}(\mathcal{T}) = \max(1, 2 \cdot \sqrt{\phi(M) \cdot t})$ incorporates model size ($\phi(M)$ in billions of parameters) and training time allocation (t in hours).

Performance is aggregated across multiple time windows to balance recent achievements with established track records:

$$S_{\text{temporal}}(m_i) = \sum_{w \in W} \omega_w \cdot \text{Normalize}\left(\sum_{\mathcal{T} \in \mathcal{T}w} S\text{adjusted}(m_i, \mathcal{T})\right) \tag{13}$$

where W is the set of time windows 1-day, 3-day, 7-day, ω_w is the weight for window w (0.3, 0.3, and 0.4 respectively), and $\mathcal{T}_w$ is the set of tasks in window w. The final score undergoes a non-linear transformation to create appropriate reward differentials:

$$S_{\text{final}}(m_i) = \beta \cdot S_{\text{sigmoid}}(m_i) + \omega \cdot x_i \tag{14}$$

where:

$$S_{\text{sigmoid}}(m_i) = \left(\frac{1}{1 + \exp(-\gamma \cdot (x_i - \mu))}\right)^{\nu} \tag{15}$$

and $x_i = \frac{\text{quality\_score}_i}{\text{max\_score}}$ is the normalised input score, with parameters $\beta = 0.7$ (sigmoid weight), $\omega = 0.05$ (linear weight), $\gamma = 9$ (sigmoid steepness), $\mu = 0.5$ (sigmoid shift), and $\nu = 0.75$ (sigmoid power). This transformation creates pronounced rewards for top performers while maintaining reasonable incentives for all participants.

3.6 Blockchain Integration and Economic Incentives

The final scores are translated into on-chain weights that determine token emissions from the underlying Bittensor network: $W_{\text{chain}}(m_i) = S_{\text{final}}(m_i) \cdot \text{Vtrust}$.

3.7 System Fault Tolerance

Gradients additionally provides inherent fault tolerance through several complementary mechanisms. When insufficient miners accept a task, the system enters a delay state and reattempts the task later with extended time allocation following $\Delta t_{\text{delay}} = \Delta t_{\text{initial}} + c \cdot$ attempts. The assignment of multiple miners to each task (8-15 for text tasks, 15-25 for image tasks) creates redundancy that ensures successful completion even when individual miners encounter hardware failures or connectivity issues. Additionally, the weighted selection process naturally evolves to favour more reliable participants, as miners with higher success rates receive increased selection probabilities over time. These mechanisms work together to maintain system operability despite the inherent unpredictability of distributed computational environments, including fluctuations in miner participation, hardware failures, and network connectivity problems.

Having described the system design and competitive mechanism, we now evaluate whether this approach actually produces superior results in practice. Our experimental methodology tests this across diverse foundation models and task types.

4 Methodology

4.1 Experimental Design

We conducted 180 controlled experiments to evaluate the effectiveness of decentralised hyperparameter optimisation across diverse models, datasets, and task types. To ensure fair comparison, all experiments used identical conditions across platforms: the same base models, datasets, training time allocations, and evaluation metrics. We trained all models using the Gradients platform's distributed fine-tuning approach, then evaluated them using task-specific metrics against baseline implementations from leading AutoML services using these identical experimental conditions. We're focussing the experimental space on instruct and diffusion tasks - since there are no other platforms to reliably compare DPO and GRPO tasks.

4.2 Model Selection

We tested five model families spanning three orders of magnitude in size. EleutherAI Pythia-70M [1] represents the smallest scale, testing extreme parameter efficiency. The Qwen2 family [39] (1.5B, 7B) provides popular open models with strong multilingual capabilities. Llama-3.1-8B and Llama-3.3-70B [16, 25] are Meta's latest instruction-tuned models. DeepSeek-R1-Distill-70B [6] represents state-of-the-art reasoning-optimised architecture.

For diffusion models, we evaluated two state-of-the-art text-to-image architectures: SDXL 1.0 [30] (Stable Diffusion XL), the workhorse of open image generation, and Flux.1 [22], a next-generation architecture with improved coherence.

4.3 Training Protocol

We trained all models using the Gradients decentralised platform as described in Section 2. The training process used competitive hyperparameter optimisation where validators prepared train/test/synthetic splits (typically 80/10/synth), the system selected 8-15 miners for text tasks and 15-25 for image tasks via weighted probability based on historical performance, the platform allocated time based on dataset size (3-10 hours for text, 1-2 hours for images), and miners exercised complete autonomy in selecting optimisation strategies, learning rates, batch sizes, and training techniques.

4.4 Evaluation Methodology

4.4.1 Language Model Evaluation

For text models, we computed direct loss on held-out test sets. We calculated the cross-entropy loss on the test split using the fine-tuned model, applied no additional post-processing or ensemble methods, and results represent the raw predictive performance of each fine-tuning approach.

4.4.2 Diffusion Model Evaluation

For diffusion models, we implemented a dual evaluation protocol that tests the model's ability to reconstruct original images through a controlled denoising process:

Text-Guided Generation: Models received test images with added noise and accompanying text prompts, then attempted to reconstruct the original images. We measured reconstruction quality via L2 pixel loss:

$$L_{\text{text}} = \frac{1}{N}\sum_{i=1}^{N} \|I_{\text{original},i} - G(I_{\text{noisy},i}, p_i)\|_2^2 \tag{16}$$

where G is the fine-tuned generator, $I_{\text{noisy},i}$ is the noise-corrupted input image, p_i is the text prompt, and $I_{\text{original},i}$ is the target original image.

Non-Text-Guided Generation: Models performed the same image reconstruction task without text conditioning to evaluate learned style consistency:

$$L_{\text{no-text}} = \frac{1}{N}\sum_{i=1}^{N} \|I_{\text{original},i} - G(I_{\text{noisy},i}, \emptyset)\|_2^2 \tag{17}$$

We evaluated each model at 9 noise levels (0.1-0.9 denoise strength) with 10 seeds per level, testing robustness across denoising strengths.

4.5 Baseline Comparisons

We benchmarked Gradients against four leading AutoML services:

HuggingFace AutoTrain [37]: State-of-the-art AutoML with Bayesian hyperparameter optimisation

TogetherAI Fine-tuning: Cloud platform using standard fine-tuning configurations

Databricks: Enterprise MLOps platform (limited experiments due to the platform removing fine-tuning options part-way through our experiments)

Google Cloud Vertex AI: GCP's managed AutoML service (limited experiments due to prohibitive costs - \$10,000+ for a single 70B model training run)

For diffusion models, we compared against CivitAI [38], the only competitor available for this form of lora training.

4.6 Language Model Tasks

We evaluated language model fine-tuning across five categories representing diverse reasoning capabilities and linguistic complexity. These tasks span fundamental NLP challenges from mathematical reasoning to multilingual translation, testing the platform's ability to optimise hyperparameters across varied domains. Each task category presents distinct optimisation challenges due to differences in data structure, reasoning requirements, and evaluation metrics.

Mathematical Reasoning: GSM8K [4] (grade school maths), OpenThoughts-114k [36] (step-by-step solutions), and verified mathematical solutions with chain-of-thought reasoning.

Translation: Eight language pairs from Opus-100 [40] and WMT19 [28]: English$\leftrightarrow${French, Spanish, Japanese, Korean, Portuguese, Turkish, Russian, Chinese}, plus challenging low-resource pairs (Lithuanian$\rightarrow$English, Kazakh$\rightarrow$English, Gujarati$\rightarrow$English).

Code Generation: CodeAlpaca-20k [3] (instruction-following), self-instruct-starcoder [5] (compilation tasks), and CodeFeedback-Filtered (complex programming challenges).

Retrieval-Augmented Generation: RAG-Instruct [14] (document-grounded QA), Ruler [18] (long-context retrieval), and RAGBench [35, 15] (multi-hop reasoning).

Reasoning: Codeforces chain-of-thought, medical reasoning with explanations, and curated reasoning collections requiring multi-step inference.

4.7 Diffusion Model Tasks

For diffusion model evaluation, we focused on two primary challenge categories that test different aspects of fine-tuning effectiveness. These tasks assess the model's ability to learn specific visual concepts and artistic styles while maintaining generation quality and coherence. The task categories represent common real-world applications where users seek to customise diffusion models for specific aesthetic or identity requirements.

Person-Specific: 16 individuals with 10-50 images each, testing identity preservation across poses, lighting, and contexts.

Style Transfer: 18 artistic styles including cyberpunk, neoclassical, romanticism, constructivism, pointillism, pop art, and various cultural aesthetics (ukiyo-e, vintage anime).

5 Results

Our experimental evaluation tests whether economic incentives drive miners to discover task-specific optimisation strategies that consistently outperform standard approaches. The results support this hypothesis: competitive pressure leads to systematic performance improvements across diverse problem types.

5.1 Task-Specific Performance Analysis

The effectiveness of decentralised optimisation varies by task complexity. Figure 1 shows comprehensive statistical breakdown across task types. RAG and reasoning tasks—requiring complex contextual understanding—show the largest gains, while well-studied translation tasks show more modest but consistent improvements.

5.2 Overall Performance and Relative Analysis

Figure 2 demonstrates consistent superiority across all competitors, achieving 82.8% win rate against HuggingFace AutoTrain and 100% against TogetherAI, Databricks, and Google Cloud across 180 controlled experiments.

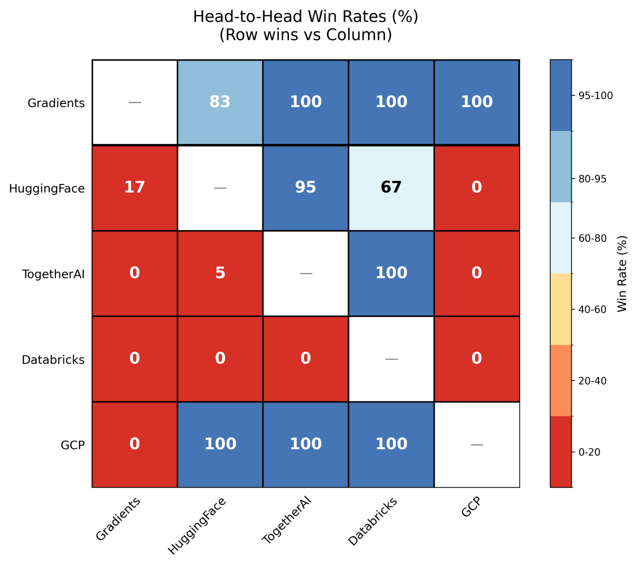


Figure 2: Win rates against leading AutoML platforms. Matrix shows percentage of experiments where Gradients achieved lower loss than competitors across 180 controlled experiments.

Figure 3 reveals the magnitude of these improvements, with distributions showing substantial gains—averaging 11.8% against HuggingFace and 42.1% against TogetherAI.

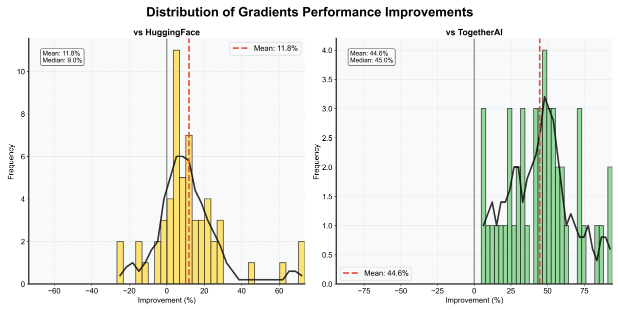
5.3 Model Scale Effects

A surprising finding emerges from model scale analysis (Figure 4): mid-scale models (7-8B parameters) benefit most from hyperparameter optimisation. This challenges conventional wisdom about model robustness and suggests an optimal complexity range for hyperparameter sensitivity, although we wish to note to the reader that there were fewer 70b model experiments due to prohibitive costs (over $10,000 per experiment for GCP 70b llama for example) - so experiment noise could play a role.

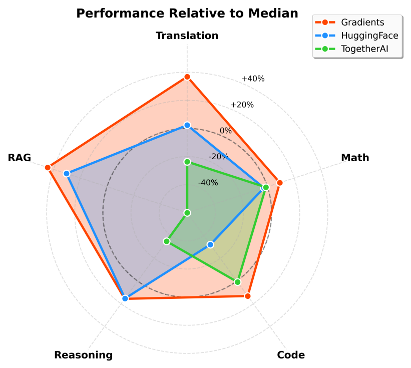
5.4 Implications and Insights

Our results reveal two potential insights about hyperparameter optimisation:

1. The optimisation landscape is more complex than assumed. The 100% win rate against TogetherAI and 82.8% against HuggingFace—across diverse tasks and models—indicates that standard configurations occupy local optima. The broad improvement

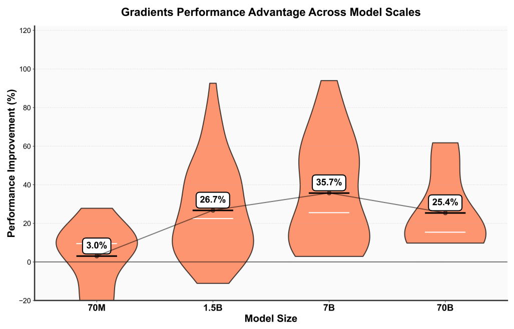


(a) Improvement distributions across experiments

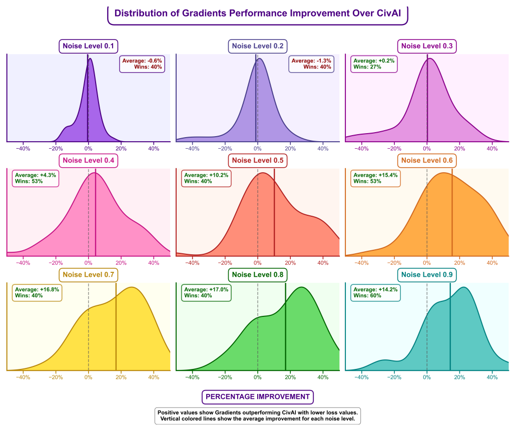


(b) Performance relative to median

Figure 3: Performance magnitude analysis. (a) Improvement distributions across all experiments demonstrate both consistency and magnitude of gains over competitors. (b) Spider chart visualises performance relative to cross-provider median, where positive values indicate above-median performance.



(a) Language model scale effects



(b) Diffusion model improvements

Figure 4: Scale and architecture effects. (a) Performance improvements peak at 7-8B parameters, suggesting optimal hyperparameter sensitivity. Very small (70M) and large (70B) models show reduced gains. (b) Diffusion model improvements increase with task difficulty - Gradients' advantage grows substantially as noise levels rise, demonstrating superior optimization for challenging denoising scenarios.

distributions suggest multiple superior configurations exist but remain undiscovered by conventional search.

2. Task complexity determines optimisation potential. Complex tasks (RAG, reasoning) show 30-40% improvements while mature tasks (translation) show 10-20%. This pattern suggests that as we tackle increasingly sophisticated AI challenges, the value of the decentralised mining pool seemingly grows proportionally.

These findings validate our core thesis: decentralised competitive search, by aligning economic incentives with exploration, systematically discovers configurations that centralised approaches miss. The consistency across 180 experiments provides strong evidence that current fine-tuning practices leave substantial performance unrealised.

6 Discussion

Why Competitive Approaches Work Our results suggest that competition alters how hyperparameter optimisation occurs. In our experiments, centralised systems followed predetermined search strategies, whilst independent miners developed varied approaches. The tournament structure in our system provided incentives for exploring configurations that centralised methods did not test, as evidenced by the consistent performance improvements across task types. The 82.8% win rate against HuggingFace and 100% against other providers indicates that competitive exploration can identify configurations in regions of hyperparameter space that standard AutoML approaches do not examine.

Task Complexity Correlation We observed a pattern where certain complex tasks showed larger performance improvements: RAG tasks improved by 23-48 and reasoning tasks by 24-31, whilst translation tasks improved by 9-57 and mathematical tasks by 1-26. This suggests that the type of complexity matters—tasks requiring multi-step reasoning and information synthesis showed the most substantial gains, while mathematical problem-solving showed more modest improvements despite its inherent complexity.

Computational Requirements One limitation with the Gradients approach is that it requires substantially more computational resources than centralised alternatives. Each task involves 8-15 parallel training runs instead of a single optimisation attempt, resulting in proportionally higher energy consumption and cost. Whilst this produces better-performing models, the environmental and economic costs may limit applicability in resource-constrained settings. The energy trade-off between optimisation quality and computational efficiency represents a fundamental limitation of the competitive approach.

7 Conclusion

Our findings reveal that hyperparameter optimisation contains unexploited potential that centralised search strategies often miss. Through competitive economic coordination, Gradients' discovers superior fine-tuning configurations across diverse foundation model architectures and task types, achieving 82.8% win rate against HuggingFace AutoTrain and perfect performance against other leading platforms. The 30-40% improvements on complex reasoning tasks and 23% gains for diffusion models indicate that current AutoML approaches explore only a fraction of viable hyperparameter space.

References

[1] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*, 2023.

[2] Yung-Chieh Chan, George Pu, Apoorv Shanker, Navdeep Jaitly, Joshua Susskind, and Mohammad Norouzi. Balancing cost and effectiveness of synthetic data generation strategies for llms. *arXiv preprint arXiv:2410.23940*, 2024.

[3] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

[4] 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 preprint arXiv:2110.14168*, 2021.

[5] CodeParrot. Self-instruct-starcoder: Instruction dataset generated from starcoder. `https://huggingface.co/datasets/codeparrot/self-instruct-starcoder`, 2023.

[6] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

[8] Ning Ding, Zhenghao Han, Sheng Zhou, and Zhisheng Niu. Incentive mechanism design for distributed coded machine learning. *arXiv preprint arXiv:2012.08715*, 2020.

[9] Zeyu Ding, Xipeng Qiu, Xuanjing Zhang, and Xuanjing Huang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[10] Arkadiy Dushatskiy, Hector Mendoza, Madalina M Drugan, and Bas van Stein. Multi-objective population based training. *arXiv preprint arXiv:2306.01436*, 2023.

[11] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David C. Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1706–1715. PMLR, 2019.

[12] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyper-parameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR, 2018.

[13] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28, pages 2962–2970, 2015.

[14] FreedomIntelligence. Rag-instruct: Knowledge-grounded instruction tuning dataset. `https://huggingface.co/datasets/FreedomIntelligence/RAG-Instruct`, 2024.

[15] Robert Friel, Masha Belyi, and Christophe Baik. RAGBench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024.

[16] Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] Justin D Harris and Bo Waggoner. Decentralized & collaborative ai on blockchain. *arXiv preprint arXiv:1907.07247*, 2019.

[18] Cheng-Ping Hsieh, Simengs Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[20] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[21] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.

[22] Black Forest Labs. FLUX.1: A 12 billion parameter rectified flow transformer for text-to-image synthesis. `https://huggingface.co/black-forest-labs/FLUX.1-dev`, 2024.

[23] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[25] Meta. Llama 3.3 70b instruct. `https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct`, 2024.

[26] Benny Moldovanu and Aner Sela. The optimal allocation of prizes in contests. *American Economic Review*, 91(3):542–558, 2001.

[27] Opentensor Foundation. Dynamic tao (dtao) proposal and implementation. Deployed February 13, 2025, 2025.

[28] WMT Organizers. ACL 2019 fourth conference on machine translation (WMT19), shared task: Machine translation of news, 2019.

[29] 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.

[30] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[31] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2024.

[32] Yuma Rao, Jacob Steeves, Ala Shaabana, Daniel Attevelt, and Matthew McAteer. Bittensor: A peer-to-peer intelligence market. *arXiv preprint arXiv:2003.03917*, 2021.

[33] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[34] Mahdi Shayan, Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Biscotti: A ledger for private and secure peer-to-peer machine learning. *arXiv preprint arXiv:1811.09904*, 2021.

[35] Yixuan Tang and Yi Yang. MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024.

[36] Open Thoughts Team. Openthoughts-114k: Open synthetic reasoning dataset with 114k high-quality examples. `https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k`, 2025.

[37] Abhishek Thakur. Autotrain: No-code training for state-of-the-art models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 419–423. Association for Computational Linguistics, 2024.

[38] Yimeng Wei, Yingjie Zhu, Yunzhuo Zheng, Jingwei Chai, and Jingjing Chen. Exploring the use of abusive generative ai models on civitai. *arXiv preprint arXiv:2407.12876*, 2024.

[39] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[40] Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. Improving massively multilingual neural machine translation and zero-shot translation. *arXiv preprint arXiv:2004.11867*, 2020.

[41] Binghai Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193*, 2024.

[42] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.