FULCRUM: PREDICTING LAYER POTENTIAL BEFORE RL-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) is a central paradigm for post-training large language models, yet it typically updates all parameters at substantial compute and memory cost. Prior research has shown that layers differ in their ability to reshape the final representation. Concentrating updates on high-potential layers may therefore achieve performance comparable to full-parameter training at lower cost with limited training resources. The challenge is to identify these layers before RL begins. To address this challenge, we propose Fulcrum, a pre-RL layer assessment and selection method. It measures a layer's potential as the maximum gain of a change in its representation as it propagates through the downstream network. This gain is characterized by the spectral norm of the downstream Jacobian and efficiently estimated using power iteration based on vector–Jacobian products (VJPs). Fulcrum further converts the resulting discrete layer potential ranking into a contiguous high-potential interval for training. Identifying this interval takes approximately 10.9 seconds for an 8B model. In Group Relative Policy Optimization (GRPO) experiments with Qwen3-1.7B, 4B, and 8B, updating only 27.8%–35.7% of all layers matches full-parameter training across ten benchmarks. On Qwen3-8B, peak GPU memory use falls by 56.2%. Experiments with supervised fine-tuning (SFT) and LoRA fine-tuning of the SDXL U-Net further demonstrate Fulcrum's applicability across training objectives, architectures, and update strategies.

1 INTRODUCTION

Reinforcement learning (RL) has become a central post-training paradigm for improving the reasoning and decision-making capabilities of large language models (LLMs) (DeepSeek-AI, 2025; Yang et al., 2025). A strong reference setting for capability-oriented RL is full-parameter fine-tuning, which allows optimization to directly reshape the original parameters without being constrained by predefined low-dimensional parameterizations such as adapters. Growing research indicates that layers contribute unequally to post-training performance (Lee et al., 2023; Li et al., 2025; Pan et al., 2024), suggesting that some layers may have greater potential to reshape the final representation. This naturally raises an intriguing hypothesis: Given a fixed amount of training data, concentrating updates on high-potential layers may achieve gains comparable to full-model training at lower cost. The key question is: how can we assess layer potential before RL training begins?

A recent study (Zhang et al., 2026) investigates layer selection in RL post-training by training each layer in isolation to estimate its contribution to the full-parameter gain. However, it infers layer importance primarily from statistical observations and offers limited theoretical grounding for why certain layers matter more than others. Taking a broader view, several studies have explored layer selection for post-training from different perspectives. One line of work evaluates layer importance based on static or local model structures, such as representation similarity and inter-layer relationships (Men et al., 2025; Biswas et al., 2026; Aubry et al., 2025). However, these middle-to-middle metrics capture only local representation changes. They ignore that subsequent layers may attenuate or amplify the local changes. Another line of work adopts objective-dependent metrics based on gradients, Hessians, influence functions, or Fisher information (Lee et al., 2023; Askari et al., 2025; Wang et al., 2025). These middle-to-loss metrics are defined using prespecified losses or labeled data. In RL, however, the optimization signal is jointly determined by policy-generated trajectories, rewards, and advantage estimates, and it changes as the policy is updated. Whether these methods can be applied to layer selection in RL therefore remains unclear and has not been systematically

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

studied. In addition, some post-training methods identify effective layers only during or after training through dynamic layer sampling, jointly learned masks, or separate layerwise training runs (Pan et al., 2024; Shi et al., 2025; Luo et al., 2024). Although these studies show that layers contribute differently, they cannot identify an efficient training interval at low cost before RL begins.

Motivated by the above analysis, we propose a pre-RL assessment of layer potential using spectral analysis of the downstream Jacobian. A layer's potential reflects the maximum gain of a unit change in its representation as that change propagates to the final representation. We measure this maximum gain by the largest singular value of the downstream Jacobian. This middle-to-final metric accounts for the complete downstream propagation process without relying on a specific task or loss function, making it well suited to the dynamically changing optimization signals in RL.

Our main contributions are summarized as follows:

- **Middle-to-final layer potential assessment.** We propose to assess layer potential before RL by tracing changes in each layer's hidden state to the final representation. The layer potential is defined as the maximum gain achievable through this downstream propagation. The resulting assessment requires no task-specific losses, parameter updates, or pilot training runs, and naturally generalizes to other post-training paradigms.
- **Fulcrum: quantifying layer potential before RL.** We propose Fulcrum to provide a mathematical characterization of layer potential before RL training. It operationalizes layer potential through spectral analysis of the downstream Jacobian, identifying the largest singular value as the layer potential. Fulcrum thus provides a theoretically grounded explanation of how changes in certain layers' representations influence the final representation.
- **Practical Fulcrum implementation.** Explicitly constructing downstream Jacobians is prohibitively expensive for LLMs. Fulcrum combines power iteration with vector–Jacobian products (VJPs) to efficiently estimate the largest singular value for each layer. It then detrends the discrete layerwise potential estimates and converts them into a contiguous high-potential interval suitable for practical fine-tuning. The entire procedure of Fulcrum takes approximately 10.9 seconds for an 8B model.
- **Generality of Fulcrum in post-training.** The generality of Fulcrum is evaluated through comprehensive experiments spanning SFT, transformer and U-Net architectures, and both full-parameter and LoRA-based updating. On Qwen3-8B, SFT over the high-potential interval slightly outperforms full-model updating on average. On SDXL, applying LoRA only to high-potential layers achieves comparable generation performance while using only 30.3% as many trainable parameters as LoRA applied to all layers.

2 Related Work

Jacobian spectra of deep networks. The Jacobian spectrum has primarily been used to study network trainability and learned representation structure. At random initialization, it characterizes how perturbations and gradients propagate along different directions, thereby determining whether a deep network can be optimized stably (Saxe et al., 2014). Some other studies show that the spectrum generally broadens with depth, whereas orthogonal initialization, appropriate activation functions, and residual connections can keep it concentrated, enabling or approximating dynamical isometry (Pennington et al., 2017; 2018; Tarnowski et al., 2019). Jacobians have also been used to analyze the structures learned by a model after training. The Jacobian norm is associated with generalization, while the alignment between the leading singular vectors of consecutive blocks correlates with downstream performance (Novak et al., 2018; Li & Papyan, 2023; Aubry et al., 2025). Gurnee et al. (2026) use downstream composite Jacobians to map intermediate activations into the final representation space, revealing learned semantic structure through vocabulary rankings.

Overall, prior work has focused on optimization stability at initialization or on interpreting learned structures and semantics after training. It has not investigated whether layerwise downstream Jacobians of a base model can benefit post-training. Nevertheless, these studies establish that the Jacobian spectrum captures how representational perturbations propagate and amplify through a network, while reflecting its trainability, representation structure, and downstream behavior. These findings provide the theoretical foundation for Fulcrum.

Layer selection in post-training. Existing layerwise post-training strategies can be organized according to the information used to identify important layers. The first group identifies important layers from static or local signals, including representation similarity across layers or representative tokens and single-layer Jacobians (Men et al., 2025; Biswas et al., 2026; Aubry et al., 2025; Gromov et al., 2025). Such signals describe local transformations rather than the effect of an intermediate state after it passes through all subsequent layers. Their limitations are also reflected in the inconsistent layer rankings produced by different local criteria on the same model (Sun et al., 2025a;b). The second group estimates layer importance relative to a training objective. Surgical Fine-Tuning uses relative gradient norms to select layers for updating; LayerIF combines gradients, Hessian information, and influence functions to relate layers to validation loss; and FLoE uses Fisher information to place low-rank expert adapters in loss-sensitive layers (Lee et al., 2023; Askari et al., 2025; Wang et al., 2025). IST updates layer importance using the loss change caused by suppressing a layer's PEFT response (Yao et al., 2024). These methods incorporate task information, but require a prespecified loss, labels, validation examples, or PEFT modules. Their applicability before RL is unclear because RL optimization depends on policy-generated trajectories, rewards, and changing advantage estimates rather than a fixed supervised objective. The third group makes layerwise decisions using information from optimization or completed runs. LISA samples layers to update during fine-tuning, and BAdam optimizes parameter blocks sequentially for memory efficiency (Pan et al., 2024; Luo et al., 2024). ILA learns layer masks jointly with LoRA updates, whereas One-Layer ranks layers by training each one separately with RL (Shi et al., 2025; Zhang et al., 2026). Together, they provide evidence that layers contribute differently to post-training. Their reliance on training, however, leaves open how to predict a compact interval for full-parameter updating directly from a base model before RL.

3 METHODOLOGY

This section introduces Fulcrum. We first establish the downstream mapping from each intermediate layer to the final representation. We then provide a mathematical characterization of layer potential as the largest singular value of the corresponding downstream Jacobian, obtained through spectral analysis. Finally, Fulcrum uses VJP-based power iteration to estimate this quantity efficiently and converts the resulting discrete layerwise potential estimates into a contiguous high-potential interval for RL training. Figure 1 shows the structure of Fulcrum.

3.1 HIDDEN REPRESENTATIONS IN TRANSFORMERS

Given an input sequence of T tokens, a transformer maps the token at each position t to an initial representation $h_t^0 \in \mathbb{R}^d$, where d is the model dimension. Stacking the representations across token positions gives

$$H^0 = \left(h_1^0, h_2^0, \ldots, h_T^0\right)^\top \in \mathbb{R}^{T\times d}. \tag{1}$$

The initial representations are subsequently transformed by L transformer layers, also referred to as blocks, each comprising a multi-head attention, a feed-forward network, normalization layers, and residual connections. The successive transformations can be written as

$$H^0 \xrightarrow{F_1} H^1 \xrightarrow{F_2} \cdots \xrightarrow{F_L} H^L, \tag{2}$$

where $H^l = F_l(H^{l-1}) \in \mathbb{R}^{T\times d}$ denotes the hidden state after layer l, and $h_t^l \in \mathbb{R}^d$ is its state at token position t. Each transformer layer jointly transforms the hidden states at all token positions, while the attention structure determines which source positions can influence each target position. Here, H^L denotes the model's final representation before the RMSNorm and unembedding operation.

3.2 JACOBIAN PROFILING

For layer l, we treat its output H^l as the input to the remaining computation. The mapping from H^l to the final representation can be regarded as $H^L = \Phi_l(H^l)$, where Φ_l comprises all downstream layers

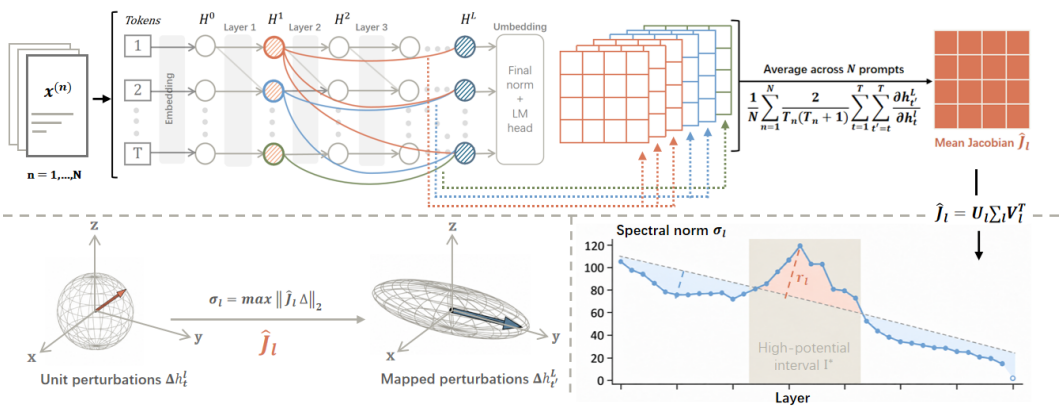


Figure 1: Overview of Fulcrum, illustrated with a decoder-only transformer. For each profiling prompt, a forward pass produces hidden states H at every token position and transformer layer. The orange/blue/yellow paths illustrate how a change at one token position in layer l propagates through subsequent layers and affects the final representations at different positions. Fulcrum aggregates the Jacobians between source and final token positions and averages them over N profiling prompts to obtain $\hat{J}_l$. The lower-left panel illustrates its geometric meaning. Under a first-order approximation, a small perturbation Δh_t^l produces a change of approximately $\hat{J}_l \Delta h_{t'}^l$ in the final representation. Thus, $\hat{J}_l$ maps unit perturbations to changes of different magnitudes and directions, and its largest singular value, $\sigma_l = \sigma_{\max}(\hat{J}_l)$, gives the maximum gain. The lower-right panel shows the resulting spectral-norm profile across layers. Fulcrum removes its linear trend with depth and uses the detrended values to identify a contiguous high-potential interval.

following layer l. Fulcrum then examines how the changes that layer l induces in H^l, simulated by position-wise perturbations, are amplified along different directions of the final representation.

We take a decoder-only transformer as an example. Let $h_t^l \in \mathbb{R}^d$ denote the output of layer l at position t. Because of causal attention, a perturbation to the intermediate state h_t^l at source position t can affect the final representation $h_{t'}^L$ only at positions $t' \geq t$. For each causally reachable source–target pair, the first-order relationship is characterized by the position-wise downstream Jacobian

$$J_{l,t\to t'} = \frac{\partial h_{t'}^L}{\partial h_t^l} \in \mathbb{R}^{d\times d}, \qquad t' \geq t. \tag{3}$$

Its (i, j)-th entry measures how an infinitesimal change in the j-th dimension of the intermediate state at source position t affects the i-th dimension of the final representation at target position t'. Accordingly, for a sufficiently small perturbation Δh_t^l in the hidden state at layer l, representing a possible change induced by updating that layer, the resulting change in $h_{t'}^L$ is locally approximated by

$$\Delta h_{t'}^L \approx J_{l,t\to t'}\, \Delta h_t^l. \tag{4}$$

To characterize layer l as a whole rather than any individual position, we perturb the output of layer l at every source position separately and aggregate the resulting Jacobians over all causally reachable source–target pairs.

Concretely, for an input sequence x of length T, we consider every pair (t, t') with $1 \leq t \leq t' \leq T$, average the corresponding Jacobians $J_{l,t\to t'}(x)$ over all such pairs, and further average this quantity over $N = 100$ input sequences $\{x^{(n)}\}_{n=1}^N$, so as to reduce the influence of any particular input corpus. This yields the layer-level downstream Jacobian

$$\hat{J}_l = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{2}{T_n(T_n+1)}\sum_{t=1}^{T_n}\sum_{t'=t}^{T_n} J_{l,t\to t'}\left(x^{(n)}\right)\right), \tag{5}$$

where T_n denotes the length of the n-th sampled sequence. The matrix $\hat{J}_l$ captures the structural reachability of layer l by describing the average first-order response of the final representation to perturbations entering at that layer.

The same procedure extends directly to encoders. For an encoder with bidirectional attention, a perturbation at source position t can affect the final representation at every position rather than only at $t' \geq t$. In that case, the inner average in Eq. 5 is instead taken over all pairs $1 \leq t, t' \leq T_n$, i.e., $\frac{1}{T_n^2} \sum_{t=1}^{T_n} \sum_{t'=1}^{T_n} J_{l,t\to t'}(x^{(n)})$.

3.3 Gain from the Spectral Norm

The aggregated Jacobian $\hat{J}_l$ maps a perturbation in one direction of the intermediate representation to changes across multiple directions of the final representation. Different input directions yield different gains through the downstream network. The largest possible gain is the spectral norm,

$$\sigma_l = \max_{\|\Delta\|_2=1} \|\hat{J}_l \Delta\|_2 = \|\hat{J}_l\|_2 = \sigma_{\max}(\hat{J}_l). \tag{6}$$

Let $\hat{J}_l = U_l \Sigma_l V_l^\top$ be its singular value decomposition. The maximum is attained by the leading right singular vector $v_{l,1}$, which identifies the intermediate perturbation direction most strongly amplified by the downstream network. The corresponding left singular vector $u_{l,1}$ gives the direction of the resulting change in the final representation. Fulcrum takes σ_l as the potential of layer l, since it measures the largest final-representation change that can be induced by a unit perturbation associated with that layer. Here, σ_l provides a structural estimate of layer potential rather than a score used directly for layer selection. To determine the RL training interval, Fulcrum compares each σ_l with the depthwise trend and then fits a contiguous high-potential interval I^* based on the resulting relative-potential rankings.

3.4 Efficient Layer Selection

VJP-based estimation. Directly constructing J_l and performing singular value decomposition would incur prohibitive memory and computational costs. We therefore adopt a matrix-free power iteration procedure, in which the required multiplications involving J_l and $J_l^\top$ are evaluated through automatic differentiation, with the transpose multiplication implemented as a vector–Jacobian product (VJP). By retaining the intermediate representations during a shared forward pass, a single backward pass can propagate the probe vector from the final representation to all layers, enabling efficient estimation of $\sigma_{\max}(J_l)$ for every layer without explicitly materializing any Jacobian.

High-potential interval fitting. Directly selecting the layers with the highest potential may produce a scattered set of layers, making RL post-training more difficult to implement. We therefore introduce a high-potential interval selection strategy that identifies a contiguous training interval. Since layer potential typically exhibits an overall trend with network depth, we score each layer by the extent to which its potential exceeds this trend. Given the potential sequence $\sigma_1, \ldots, \sigma_L$, we fit a linear trend

$$\hat{\sigma}_l = al + b$$

over $l = 1, \ldots, L$ using least squares. We then define

$$r_l = \sigma_l - \hat{\sigma}_l$$

as the relative potential score of layer l.

Given a selection budget of K layers, we rank all layers in descending order of r_l, assigning rank 1 to the layer with the highest score. We select the contiguous interval of K layers with the smallest sum of ranks:

$$s^* = \operatorname*{arg\,min}_{1 \leq s \leq L-K+1} \sum_{l=s}^{s+K-1} \operatorname{rank}(r_l), \qquad I^* = [s^*, s^* + K - 1].$$

During RL post-training, all layers in I^* are updated, while the remaining layers are frozen. We use $K = 10$ as the recommended selection budget, as the performance saturates at this value while retaining substantial efficiency gains, as shown in Section 4.3.

4 EXPERIMENTS

We evaluate Fulcrum through its RL post-training performance, potential-prediction validity, layer-budget selection, method efficiency, and generality. For LLM experiments, we use Qwen3-1.7B-Base, Qwen3-4B-Base, and Qwen3-8B-Base. They contain 28, 36, and 36 layers, respectively, with corresponding hidden sizes of 2048, 2560, and 4096. The generality experiments further include Stable Diffusion XL (SDXL) (Podell et al., 2024), a latent diffusion model that generates images at a native resolution of 1024×1024. Its U-Net backbone contains approximately 2.6B parameters and is conditioned on two text encoders, OpenCLIP ViT-bigG and CLIP ViT-L. Detailed experimental settings are provided in Appendix A.1.

4.1 PERFORMANCE COMPARISON

We evaluate Fulcrum on Qwen3-1.7B-Base, Qwen3-4B-Base, and Qwen3-8B-Base in this experiment. With $K = 10$, Fulcrum identifies the training intervals $[6, 15]$, $[13, 22]$, and $[12, 21]$ for the three models, respectively (see Appendix A.2 for details). All methods are trained on a 50K verifiable subset of NuminaMath-1.5 (Li et al., 2024) using GRPO (Shao et al., 2024) implemented in veRL. Training uses AdamW with a constant learning rate of 1×10^{-6}. Base, Full, LoRA, and Fulcrum are each evaluated five times, and Flexora and SALF once, covering mathematics, coding, reasoning, and language understanding.

Table 1 compares Fulcrum with baselines including the base model, full-parameter fine-tuning, LoRA applied to all layers (Hu et al., 2021), and two layer-selection methods, Flexora (Wei et al., 2025) and SALF (Gu et al., 2025). The best result is shown in bold, and the second-best is underlined. It shows updating only $[12, 21]$ achieves a ten-benchmark average close to Full and higher than LoRA, Flexora, and SALF.

Table 1: Accuracy (%) of Qwen3-8B on the ten benchmarks under GRPO.

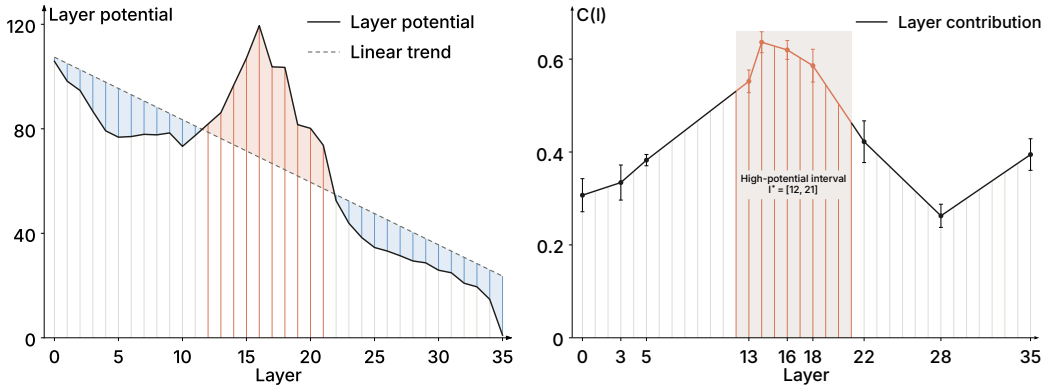
Setting	M500	Oly.	GSM8K	Min.	HE+	MBPP	GP-D	M-Pro	IFEval	C-E
Base	69.88	36.05	86.52	23.60	28.17	57.74	32.32	45.48	38.56	69.32
Full	79.52	**46.35**	**94.44**	**39.12**	**34.76**	73.07	**40.91**	**58.08**	45.51	76.64
LoRA	70.72	37.66	88.84	24.85	29.51	58.13	34.44	46.20	40.00	70.08
SALF	73.00	40.50	91.40	26.47	31.10	57.98	38.89	51.60	40.67	71.00
Flexora	69.80	38.87	89.00	22.79	29.88	57.98	36.87	47.20	36.78	69.80
Fulcrum - $[12, 21]$	**79.84**	45.96	94.00	38.46	32.20	**73.39**	39.70	55.44	**47.17**	**76.80**

Table 2 extends the comparison across all three model sizes, with complete benchmark-level results in Appendix A.4. Fulcrum matches Full on the ten-benchmark average for Qwen3-1.7B-Base, exceeds it on Qwen3-4B-Base, and remains close on Qwen3-8B-Base, while training only 23.6%–29.3% of the model parameters. Its advantages over Full vary by category: Fulcrum surpasses Full in coding and reasoning on the 1.7B and 4B models, and in language on the 8B model. Overall, the results show that the compact intervals identified before RL retain most or all of the aggregate gains from full-model post-training across model scales.

Table 2: Category-level accuracy (%) and trainable parameters under GRPO across three model sizes. Values in parentheses indicate absolute changes relative to the corresponding base model.

Size	Setting	Math	Code	Reasoning	Language	Trainable params. (M)
1.7B	Base	37.30	33.22	19.88	28.60	–
	Full	**50.71** (+13.41)	39.23 (+6.01)	34.09 (+14.21)	**42.22** (+13.62)	1,720.6
	Fulcrum - $[6, 15]$	50.07 (+12.77)	**40.94** (+7.72)	**34.31** (+14.43)	41.57 (+12.97)	503.4
4B	Base	47.88	39.34	34.68	46.11	–
	Full	**61.51** (+13.63)	43.13 (+3.79)	48.34 (+13.66)	**60.72** (+14.61)	4,022.5
	Fulcrum - $[13, 22]$	61.42 (+13.54)	**54.26** (+14.92)	**50.21** (+15.53)	56.73 (+10.62)	1,009.3
8B	Base	54.01	42.96	38.90	53.94	–
	Full	**64.86** (+10.85)	**53.92** (+10.96)	**49.49** (+10.59)	61.07 (+7.13)	8,190.7
	Fulcrum - $[12, 21]$	64.57 (+10.56)	52.79 (+9.83)	47.57 (+8.67)	**61.99** (+8.05)	1,929.5

4.2 Validating Potential Estimation



(a) Pre-RL layer potential assessment.

(b) Measured layer contributions after post-training.

Figure 2: Pre-RL potential and post-training layer contribution on Qwen3-8B-Base. (a) The spectral norm of each layer's downstream Jacobian and its linear trend with depth. Fulcrum ranks layers by the detrended spectral norm and identifies $[12, 21]$ as the high-potential interval (orange). (b) Contribution of ten representative layers post-trained individually. Each point shows the mean over five evaluation runs, with error bars indicating one standard deviation.

The central hypothesis of Fulcrum is that layers with greater potential to reshape the final representation should yield greater contribution when updated during post-training. To test this hypothesis, we select ten representative layers of Qwen3-8B-Base spanning different levels of the detrended spectral norm: layers 0, 3, 5, 13, 14, 16, 18, 22, 28, and 35. Each layer is post-trained independently while all other layers remain frozen, and the resulting model is evaluated five times. Following Zhang et al. (2026), we compute the contribution of a single layer l as

$$C(l) = \frac{m_l - m_{\text{Base}}}{m_{\text{Full}} - m_{\text{Base}}},$$

where m_l, m_{Base}, and m_{Full} are the mean accuracies over the ten benchmarks after training only layer l, before post-training, and after full-parameter post-training, respectively. Figure 2 compares these scores with the potential predicted by Fulcrum before RL.

As shown in Figure 2, the left panel shows the spectral norm of the downstream Jacobian for each layer. After removing its linear trend with network depth, Fulcrum identifies the contiguous interval $[12, 21]$. The right panel reports the layer contribution obtained by single-layer fine-tuning. Layers with higher $C(l)$ are primarily concentrated within the high-potential interval identified before RL, indicating broad agreement between predicted potential and observed layer contribution. This result supports the central hypothesis of Fulcrum and shows that its pre-RL assessment can effectively locate the model region in which parameter updates yield the greatest benefits. Because layer potential is defined by the amplification from a layer's output to the final representation, Fulcrum is limited in assessing the final layer. The predicted potential of the final layer is inconsistent with the measured trend, as σ_L is fixed at 1 regardless of the model. Detailed benchmark-level results are provided in Appendix A.6.

4.3 Effect of the Layer Budget K

We investigate how the number of updated layers affects downstream performance and training cost, thereby validating the default choice of $K = 10$ introduced in Section 3. On Qwen3-8B-Base, we evaluate the performance and training cost of the contiguous high-potential interval $I^*(K)$ selected by Fulcrum for $K \in \{1, 4, 7, 10\}$. The corresponding intervals are $[16]$, $[15, 18]$, $[14, 20]$, and $[12, 21]$, respectively. We compare these settings with full-parameter post-training, which updates all 36 layers ($K = 36$). Performance is measured by the mean accuracy across ten benchmarks, and peak GPU memory is recorded during training.

Figure 3 shows that accuracy increases rapidly as more layers are updated and approaches full-model performance at ten layers, whereas peak memory grows approximately linearly. The ten-layer interval therefore provides a favorable balance between performance and memory consumption, supporting $K = 10$ as our default setting. Nevertheless, K can be adjusted according to task requirements and computational resources. Detailed benchmark results and cost measurements for this analysis are provided in Appendix A.5.

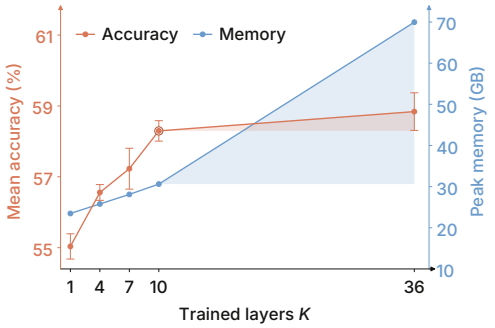


Figure 3: Accuracy and peak memory on Qwen3-8B-Base as the number of trained layers increases.

4.4 Efficiency of Fulcrum

We evaluate the profiling efficiency of Fulcrum on three model sizes. The exact backend constructs each layer's Jacobian row by row and computes its spectral norm using singular value decomposition. In contrast, the VJP-based estimation avoids explicit Jacobian construction and estimates the spectral norm from vector–Jacobian products obtained with a small number of probe vectors. The high-potential intervals used in other experiments are identified by the VJP-based estimation using 100 prompts.

Table 3: Runtime of exact and VJP-based layer-potential prediction.

Model	Selected interval	Exact (s)	VJP-based estimation (s)	Estimation, fewest prompts and probes (s)
Qwen3-1.7B-Base	$[6, 15]$	828.1	5.6	1.0
Qwen3-4B-Base	$[13, 22]$	2438.3	9.0	0.6
Qwen3-8B-Base	$[12, 21]$	6437.9	10.9	0.5

As shown in Table 3, the VJP-based estimation identifies the same intervals as exact computation while achieving speedups of $148\times$ to $591\times$. With the fewest prompts and probes that preserve the selected intervals, profiling requires only 0.5–1.0 seconds. This cost is negligible compared with RL post-training, demonstrating that Fulcrum is highly practical for real-world use.

4.5 Generality Beyond RL

We further evaluate whether Fulcrum generalizes beyond RL across training objectives, model architectures, and parameter-update strategies.

Transfer to SFT. We first apply Fulcrum to SFT on Qwen3-8B-Base using the complete step-by-step solutions from NuminaMath-1.5-Pro. The predicted interval remains $[12, 21]$, and performance is evaluated on four mathematical reasoning benchmarks. As shown in Table 4, updating this ten-layer interval achieves an average accuracy of 60.86%, slightly exceeding the 59.99% obtained by full-model updating. Fulcrum performs better on MATH500 and GSM8K and remains comparable on Olympiad and Minerva. The same interval therefore matches or surpasses full-model updating under both GRPO and SFT, showing that Fulcrum transfers across post-training objectives.

Table 4: Accuracy (%) on four mathematical reasoning benchmarks. Results are reported as mean $\pm$ standard deviation over five evaluation runs.

Setting	MATH500	Olympiad	GSM8K	Minerva	Avg.
Base	69.88 ± 1.19	36.05 ± 1.59	86.52 ± 1.14	23.60 ± 1.75	54.01
Full	$\underline{74.64} \pm 1.19$	$\mathbf{41.75} \pm 1.24$	$\underline{90.84} \pm 0.92$	$\mathbf{32.72} \pm 2.16$	$\underline{59.99}$
Fulcrum - $[12, 21]$	$\mathbf{76.60} \pm 1.45$	$\underline{41.69} \pm 0.95$	$\mathbf{92.80} \pm 1.18$	$\underline{32.35} \pm 1.05$	$\mathbf{60.86}$

Transfer to U-Net and LoRA. We next evaluate Fulcrum on the SDXL U-Net, whose width varies with depth and whose down, middle, and up blocks are connected by skip connections. This setting differs substantially from decoder-only transformers and allows us to examine whether Fulcrum generalizes across architectures. We profile 28 candidate locations and select two ResNet blocks in the first up block together with their 20 associated transformer layers. These components account for 31.16% of the U-Net parameters.

We conduct two comparisons within the same update strategy. Fulcrum updates all parameters in the selected components and is compared with full U-Net fine-tuning. Fulcrum+LoRA applies rank-16 LoRA to the attention projections of the 20 selected transformer layers and is compared with applying LoRA to all transformer layers. Training uses 10,000 COCO train2017 image–caption pairs for 2,500 updates, and evaluation uses 5,000 images generated from COCO val2017 captions. We report CLIPScore, FID, and KID, together with training time, peak memory, and FLOPs.

As shown in Table 5, Fulcrum improves all three quality metrics over full U-Net fine-tuning while reducing trainable parameters by 68.8%, training time by 34.0%, peak memory by 45.7%, and FLOPs by 39.5%. Fulcrum+LoRA uses only 30.3% of the trainable parameters of LoRA-all and reduces training time by 48.9% and FLOPs by 28.7%. It achieves a slightly higher CLIPScore, although its FID and KID are worse than those of LoRA-all, suggesting that the applicability of Fulcrum to the image modality warrants further investigation. Overall, these results demonstrate that Fulcrum extends beyond RL to SFT, transfers from decoder-only transformers to U-Nets, and supports both full-parameter updating and LoRA.

Table 5: Quality and training cost on the SDXL U-Net.

Method	Trainable parameters (M) ↓	CLIPScore ↑	FID ↓	KID ↓	Time (min) ↓	Peak memory (MiB) ↓	PFLOPs ↓
Base	–	31.58	24.35	0.00404	–	–	–
Full U-Net FT	2,567.5	31.58	22.83	0.00388	111.21	30,601	258.3
Fulcrum	800.0 (↓ 68.8%)	31.85 (+0.27)	22.11 (−0.72)	0.00361 (−0.00027)	73.41 (↓ 34.0%)	16,626 (↓ 45.7%)	156.4 (↓ 39.5%)
LoRA-all	23.2	31.73	20.33	0.00274	160.37	10,956	197.3
LoRA-Fulcrum	7.0 (↓ 69.7%)	31.77 (+0.04)	22.16 (+1.83)	0.00324 (+0.00050)	81.97 (↓ 48.9%)	10,542 (↓ 3.8%)	140.6 (↓ 28.7%)

5 Conclusion

We introduce Fulcrum, a pre-RL layer-selection method that predicts each layer's potential to reshape the final representation by measuring the largest singular value of its downstream Jacobian, estimated efficiently through VJP-based power iteration. Without relying on any task-specific loss, labels, or completed training runs, Fulcrum converts the resulting layerwise potential profile into a contiguous high-potential interval that can be identified in about 10.9 seconds for an 8B model. Across GRPO experiments on Qwen3-1.7B/4B/8B, updating this interval alone matches or exceeds full-parameter post-training while touching about one-third of the layers, and the same procedure transfers to SFT and to LoRA-based fine-tuning of the SDXL U-Net, indicating that the predicted potential reflects a structural property of the base model rather than an artifact of a particular post-training objective or architecture. Our work also opens several directions for future research. Fulcrum measures the potential gain of representation changes but does not establish whether RL updates actually realize this potential, and the relationship between high-gain directions and training-induced changes remains to be examined. Moreover, Fulcrum focuses on structural potential and does not account for subsequent processing, such as the output head and the reward mechanism, which merits further investigation. Finally, our preliminary SDXL experiments suggest that Fulcrum also applies to LoRA fine-tuning, but the current results indicate a potential trend: its gains are more consistent on text tasks than on image generation. Understanding how the U-Net architecture and the image modality shape layer potential also offers a promising direction for further study.

AI USE STATEMENT

In this work, we used generative AI tools for research execution, specifically in assisting the code implementation and running our experiments, and in reformatting the datasets and extracting data from them. We also used generative AI tools to aid and polish writing, including checking sentence-level grammar and refining word choice for more precise expression. We have not used generative AI tools to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on the research methodology or experiments, to interpret results, or to draft sections of the paper. Generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to help modify figures, to write and edit other software code, and to identify relevant literature, the latter serving to fill gaps after our own manual literature survey of the research background. We have reviewed all AI-assisted work. All AI-assisted code was reviewed by the authors; we rejected redundant code and kept the implementation concise, and verified that the processed datasets match the original sources. All suggested wording changes were checked to ensure that they preserve the intended technical meaning. We read every cited work directly. Generative AI tools changed only the visual style of the figures and did not alter the plotted data. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of our results. Section 3 fully specifies Fulcrum, including the aggregation of position-wise downstream Jacobians (Eq. 5), the spectral-norm definition of layer potential (Eq. 6), the VJP-based power-iteration estimator, and the detrending and contiguous-interval selection rule. Appendix A.1 lists the models, the GRPO training configuration and all hyperparameters, the reward definition, the evaluation benchmarks with their sizes, the decoding settings, and the evaluation seeds. Appendix A.2 reports the layerwise spectral norms, fitted trends, detrended scores, and rankings for all three Qwen3 models, so that the selected intervals can be verified independently of our code. Appendices A.3–A.5 provide the complete benchmark-level results underlying all tables and figures in Section 4. All base models (Qwen3-Base and SDXL) and datasets (NuminaMath-1.5, COCO, and the evaluation benchmarks) are publicly available. Source code of Fulcrum will be released upon publication. An anonymized, packaged implementation of Fulcrum will be made available to reviewers via an anonymous link during the review period and publicly released upon publication.

REFERENCES

Hadi Askari, Shivanshu Gupta, Fei Wang, Anshuman Chhabra, and Muhao Chen. LayerIF: Estimating layer quality for large language models using influence functions. *arXiv preprint arXiv:2505.23811*, 2025. URL https://arxiv.org/abs/2505.23811.

Murdock Aubry, Haoming Meng, Anton Sugolov, and Vardan Papyan. Transformer block coupling and its correlation with generalization in LLMs. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Aldrin Kabya Biswas, Md Fahim, Md Tahmid Hasan Fuad, Akm Moshiur Rahman Mazumder, Amin Ahsan Ali, and AKM Mahbubur Rahman. Training-free layer selection for partial fine-tuning of language models. *Information Sciences*, 740:123204, 2026. ISSN 0020-0255.

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. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. The unreasonable ineffectiveness of the deeper layers. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, pp. 81906–81920, 2025.

Jian Gu, Aldeida Aleti, Chunyang Chen, and Hongyu Zhang. A semantic-aware layer-freezing approach to computation-efficient fine-tuning of language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 8019–8033. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.420.

Wes Gurnee, Nicholas Sofroniew, et al. Verbalizable representations form a global workspace in language models. Transformer Circuits Thread, July 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

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. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-Eval: A multi-level multi-discipline Chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023. URL https://arxiv.org/abs/2305.08322.

knoveleng. Minerva-Math. Hugging Face dataset repository, 2025. URL https://huggingface.co/datasets/knoveleng/Minerva-Math.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. Hugging Face dataset repository, 2024. URL https://huggingface.co/datasets/AI-MO/NuminaMath-1.5.

Jianing Li and Vardan Papyan. Residual alignment: Uncovering the mechanisms of residual networks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 57660–57712, 2023.

Shen Li, Liuyi Yao, Lan Zhang, and Yaliang Li. Safety layers in aligned large language models: The key to LLM security. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Qijun Luo, Hengxu Yu, and Xiao Li. BAdam: A memory efficient full parameter optimization method for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24926–24958. Curran Associates, Inc., 2024.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204. Association for Computational Linguistics, 2025.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 57018–57049. Curran Associates, Inc., 2024.

Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1924–1932. PMLR, 2018.

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. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, pp. 22275–22287, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangyuan Shi et al. Understanding layer significance in LLM alignment. In *Proceedings of the 2nd Conference on Language Modeling (COLM)*, 2025.

Qi Sun, Marc Pickett, Aakash Kumar Nain, and Llion Jones. Transformer layers as painters. *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI)*, 39(24):25219–25227, 2025a.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 163104–163136. Curran Associates, Inc., 2025b.

Wojciech Tarnowski, Piotr Warchoł, Stanisław Jastrzębski, Jacek Tabor, and Maciej A. Nowak. Dynamical isometry is achieved in residual networks in a universal way for any activation function. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2221–2230. PMLR, 2019.

Xinyi Wang, Lirong Gao, Haobo Wang, Yiming Zhang, and Junbo Zhao. FLoE: Fisher-based layer selection for efficient sparse adaptation of low-rank experts. *arXiv preprint arXiv:2506.00495*, 2025. URL https://arxiv.org/abs/2506.00495.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024. URL https://arxiv.org/abs/2406.01574.

Chenxing Wei, Yao Shu, Ying Tiffany He, and Fei Yu. Flexora: Flexible low-rank adaptation for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14643–14682. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.713.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Kai Yao, Penglei Gao, Lichun Li, Yuan Zhao, Xiaofeng Wang, Wei Wang, and Jianke Zhu. Layer-wise importance matters: Less memory for better performance in parameter-efficient fine-tuning of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 2024.

Zijian Zhang, Rizhen Hu, Athanasios Glentis, Dawei Li, Chung-Yiu Yau, Hongzhou Lin, and Mingyi Hong. Is one layer enough? Training A single transformer layer can match full-parameter RL training. *arXiv preprint arXiv:2607.01232*, 2026. URL https://arxiv.org/abs/2607.01232.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

A Additional Results

This section has six parts. The first describes the experimental setup, including the models, the GRPO training configuration, and the evaluation benchmarks. The second presents an ablation that freezes the Fulcrum-selected interval and updates all remaining layers: despite training $2.4\times$–$3.2\times$ as many parameters, this setting consistently underperforms Fulcrum across all three model sizes. The third reports the layerwise spectral norms of the three Qwen3 models and shows how Fulcrum selects the high-potential intervals $[6, 15]$, $[13, 22]$, and $[12, 21]$ from the detrended values. The fourth presents the benchmark-level GRPO results: the selected interval alone matches or exceeds full-parameter post-training on Qwen3-1.7B-Base and Qwen3-4B-Base and remains within 0.54 points on Qwen3-8B-Base, while training only ten layers. The fifth varies the layer budget K on Qwen3-8B-Base: accuracy approaches that of full-parameter post-training at $K = 10$, where peak memory is 56.2% lower. The sixth reports the benchmark-level results of post-training each of ten representative layers of Qwen3-8B-Base alone, which underlie the potential validation in Section 4.2.

A.1 Experimental Setup

We evaluate Fulcrum on three Qwen3 base models: Qwen3-1.7B-Base with 28 layers and a hidden size of 2048, Qwen3-4B-Base with 36 layers and a hidden size of 2560, and Qwen3-8B-Base with 36 layers and a hidden size of 4096. For the generality study, we also adopt SDXL, a latent diffusion model whose U-Net backbone has approximately 2.6B parameters.

RL training uses GRPO (Shao et al., 2024) implemented in veRL with AdamW, on a verifiable subset of 50K problems from NuminaMath-1.5 (Li et al., 2024), with a binary reward that is 1 when the boxed answer matches the reference and 0 otherwise. The learning rate is 1e-6 with a constant schedule and no warmup, weight decay 0.01, batch size 128, 8 samples per problem, at most 512 prompt tokens and 2048 response tokens, clip ratio 0.2, no KL term, no entropy regularization, seed 1, and 390 steps, which is one epoch. All conditions share these settings.

Evaluation uses ten benchmarks in four categories. Math includes MATH500 (500 questions) (Hendrycks et al., 2021), Olympiad (674) (He et al., 2024), GSM8K (500) (Cobbe et al., 2021), and Minerva (272) (knoveleng, 2025). Code includes HumanEval+ (164) (Liu et al., 2023) and MBPP (257) (Austin et al., 2021). Reasoning includes GPQA-D (198) (Rein et al., 2023) and MMLU-Pro (500) (Wang et al., 2024). Language includes IFEval (541) (Zhou et al., 2023) and C-Eval (500) (Huang et al., 2023). Math is in-domain, while the other three categories are out-of-domain. Generation uses temperature 0.6, top-p 1.0, and at most 2048 new tokens. Each checkpoint is evaluated once under each of five seeds. A category mean is the unweighted mean of its benchmarks. In the experimental tables, M500, Oly., Min., HE+, GP-D, M-Pro, and C-E denote MATH500, Olympiad, Minerva, HumanEval+, GPQA-D, MMLU-Pro, and C-Eval, respectively.

A.2 Layerwise Spectral Norms and Interval Selection

This subsection shows how Fulcrum identifies the high-potential intervals for the three models used in our experiments. Tables A1–A3 report the largest singular value $\sigma_l = \sigma_{\max}(J_l)$ of each layer's downstream Jacobian, the fitted linear trend $\hat{\sigma}_l$, and the detrended spectral norm $r_l = \sigma_l - \hat{\sigma}_l$ used to rank the layers. Rk. denotes the rank of r_l; bold rows mark the selected intervals. The resulting high-potential intervals are $[6, 15]$, $[13, 22]$, and $[12, 21]$ for Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively. All three intervals lie in the middle portion of the network, but their exact positions differ across model sizes. The final layer has $\sigma_{35} = 1$ by definition and is therefore omitted.

Table A1: Qwen3-1.7B-Base. Selected interval: $[6, 15]$.

l	σ_l	$\hat{\sigma}_l$	r_l	Rk.
0	161.08	177.20	-16.12	22
1	154.40	170.35	-15.95	21
2	156.74	163.50	-6.77	16
3	134.94	156.66	-21.71	27
4	136.71	149.81	-13.10	20
5	138.26	142.96	-4.70	13
6	**143.84**	**136.11**	**7.73**	**8**
7	**159.03**	**129.27**	**29.76**	**4**
8	**158.81**	**122.42**	**36.39**	**1**
9	**149.23**	**115.57**	**33.66**	**2**
10	**102.99**	**108.72**	**-5.74**	**15**
11	**101.47**	**101.88**	**-0.40**	**11**
12	**101.10**	**95.03**	**6.07**	**9**
13	**99.26**	**88.18**	**11.08**	**7**
14	**114.08**	**81.33**	**32.75**	**3**
15	**87.72**	**74.49**	**13.23**	**6**
16	59.83	67.64	-7.80	17
17	49.60	60.79	-11.19	18
18	33.64	53.94	-20.31	24
19	26.56	47.09	-20.53	26
20	19.80	40.25	-20.45	25
21	17.16	33.40	-16.24	23
22	14.47	26.55	-12.08	19
23	13.97	19.70	-5.73	14
24	12.28	12.86	-0.58	12
25	11.28	6.01	5.27	10
26	22.63	-0.84	23.47	5

Table A2: Qwen3-4B-Base. Selected interval: $[13, 22]$.

l	σ_l	$\hat{\sigma}_l$	r_l	Rk.
0	36.22	40.63	-4.40	15
1	34.03	40.11	-6.08	21
2	33.21	39.59	-6.38	22
3	31.19	39.07	-7.89	26
4	27.83	38.55	-10.72	33
5	27.26	38.04	-10.78	34
6	27.04	37.52	-10.48	31
7	27.98	37.00	-9.02	29
8	28.62	36.48	-7.87	25
9	30.34	35.97	-5.63	18
10	29.79	35.45	-5.66	19
11	33.47	34.93	-1.47	13
12	37.03	34.41	2.61	11
13	**39.87**	**33.90**	**5.97**	**9**
14	**46.63**	**33.38**	**13.25**	**8**
15	**53.91**	**32.86**	**21.05**	**4**
16	**61.17**	**32.34**	**28.82**	**1**
17	**52.95**	**31.83**	**21.12**	**3**
18	**53.50**	**31.31**	**22.19**	**2**
19	**46.47**	**30.79**	**15.68**	**7**
20	**47.60**	**30.27**	**17.32**	**5**
21	**46.31**	**29.76**	**16.55**	**6**
22	**32.49**	**29.24**	**3.25**	**10**
23	27.62	28.72	-1.10	12
24	24.97	28.20	-3.24	14
25	23.07	27.69	-4.62	16
26	22.31	27.17	-4.86	17
27	20.93	26.65	-5.72	20
28	19.48	26.13	-6.66	23
29	18.83	25.62	-6.79	24
30	16.95	25.10	-8.15	27
31	16.40	24.58	-8.18	28
32	13.53	24.06	-10.53	32
33	13.52	23.55	-10.03	30
34	11.47	23.03	-11.56	35

Table A3: Qwen3-8B-Base. Selected interval: $[12, 21]$.

l	σ_l	$\hat{\sigma}_l$	r_l	Rk.
0	105.82	107.42	-1.60	11
1	98.20	105.03	-6.83	14
2	94.67	102.63	-7.97	16
3	86.68	100.24	-13.56	32
4	79.22	97.84	-18.62	34
5	76.78	95.45	-18.66	35
6	77.03	93.05	-16.02	33
7	77.88	90.66	-12.78	30
8	77.68	88.27	-10.58	24
9	78.42	85.87	-7.45	15
10	73.27	83.48	-10.20	23
11	77.46	81.08	-3.62	13
12	**81.84**	**78.69**	**3.15**	**10**
13	**86.17**	**76.29**	**9.88**	**9**
14	**96.71**	**73.90**	**22.81**	**5**
15	**107.08**	**71.50**	**35.57**	**4**
16	**119.48**	**69.11**	**50.37**	**1**
17	**103.72**	**66.71**	**37.01**	**3**
18	**103.51**	**64.32**	**39.19**	**2**
19	**81.58**	**61.92**	**19.66**	**7**
20	**80.17**	**59.53**	**20.65**	**6**
21	**73.72**	**57.13**	**16.58**	**8**
22	52.49	54.74	-2.24	12
23	43.83	52.34	-8.51	18
24	38.37	49.95	-11.58	28
25	34.56	47.55	-12.99	31
26	33.20	45.16	-11.96	29
27	31.41	42.76	-11.35	27
28	29.40	40.37	-10.97	25
29	28.69	37.97	-9.28	20
30	25.87	35.58	-9.71	21
31	24.90	33.19	-8.29	17
32	20.89	30.79	-9.90	22
33	19.51	28.40	-8.89	19
34	14.71	26.00	-11.29	26

A.3 Ablation on the selected interval.

To verify that the gains of Fulcrum stem from *which* layers are updated rather than from *how many* parameters are trained, we conduct a complementary ablation in which the Fulcrum-selected interval is frozen and all remaining layers are updated (denoted w/o). As shown in Table A4, this setting trains $2.4\times$, $3.0\times$, and $3.2\times$ as many parameters as Fulcrum on Qwen3-1.7B-Base, Qwen3-4B-Base, and Qwen3-8B-Base, respectively (1,217.2M vs. 503.4M, 3,013.2M vs. 1,009.3M, and 6,261.2M vs. 1,929.5M). Despite this substantially larger budget, it consistently underperforms Fulcrum, trailing it by 2.34, 5.46, and 1.38 points on the ten-benchmark average, and falls below full-parameter post-training by 2.34, 3.69, and 1.92 points. At the benchmark level, Fulcrum outperforms the w/o setting on 26 of the 30 model–benchmark pairs. The degradation is most pronounced on out-of-domain code and reasoning tasks; for example, on Qwen3-4B-Base, excluding the selected interval reduces MBPP from 73.15 to 52.14 and GPQA-D from 42.42 to 29.29.

Notably, the w/o setting still improves over the base model, indicating that layers outside the interval are not inert. However, they are far less parameter-efficient: they recover only 81%, 69%, and 81% of the full-parameter gain, whereas the ten-layer interval recovers 100%, 115%, and 95% with less than half of the trainable parameters. These results show that the interval identified before RL concentrates the layers whose updates matter most, and that simply enlarging the set of trainable parameters cannot compensate for excluding them.

Table A4: Ablation study of Fulcrum-selected layers.

Size	Setting	Math	Code	Reasoning	Language	Avg.	Trainable params. (M)
1.7B	Full	50.715	39.230	34.085	42.225	43.394	1,720.6
	Fulcrum $[6, 15]$	50.065	40.940	34.315	41.570	43.391 (-0.003)	503.4
	w/o $[6, 15]$	48.945	36.500	29.875	40.995	41.052 (-2.342)	1,217.2
4B	Full	61.508	43.130	48.340	60.725	55.042	4,022.5
	Fulcrum $[13, 22]$	61.418	54.260	50.210	56.725	56.806 $(+1.764)$	1,009.3
	w/o $[13, 22]$	58.768	38.570	42.745	57.925	51.355 (-3.687)	3,013.2
8B	Full	64.858	53.915	49.495	61.075	58.840	8,190.7
	Fulcrum $[12, 21]$	64.565	52.795	47.570	61.985	58.296 (-0.544)	1,929.5
	w/o $[12, 21]$	62.755	52.345	48.290	58.470	56.923 (-1.917)	6,261.2

A.4 Detailed GRPO Results

This section presents the complete results of the experiments discussed in Section 4.1. In Table A5, Avg. is the unweighted arithmetic mean of the ten benchmark scores in each row. Base, Full, LoRA, and Fulcrum are evaluated five times, and we report the mean accuracy; Flexora and SALF are evaluated once.

Table A5: Detailed GRPO results across ten benchmarks.

Size	Setting	Math				Code		Reasoning		Language		Avg.
		M500	Oly.	GSM8K	Min.	HE+	MBPP	GP-D	M-Pro	IFEval	C-E	
1.7B	Base	51.24	21.39	63.48	13.09	13.05	53.39	24.04	15.72	22.85	34.36	31.26
	Full	64.08	31.22	83.96	23.60	16.83	61.63	29.29	38.88	27.73	56.72	43.39
	$[6, 15]$	63.20	30.27	84.00	22.79	17.68	64.20	32.83	35.80	27.54	55.60	43.39
4B	Base	63.72	34.81	72.12	20.88	26.46	52.22	29.39	39.96	34.34	57.88	43.18
	Full	76.44	42.23	93.32	34.04	29.76	56.50	39.60	57.08	47.73	73.72	55.04
	$[13, 22]$	75.60	42.88	93.00	34.19	35.37	73.15	42.42	58.00	38.45	75.00	56.81
8B	Base	69.88	36.05	86.52	23.60	28.17	57.74	32.32	45.48	38.56	69.32	48.76
	Full	79.52	46.35	94.44	39.12	34.76	73.07	40.91	58.08	45.51	76.64	58.84
	$[12, 21]$	79.84	45.96	94.00	38.46	32.20	73.39	39.70	55.44	47.17	76.80	58.30
	LoRA	70.72	37.66	88.84	24.85	29.51	58.13	34.44	46.20	40.00	70.08	50.04
	Flexora	69.80	38.87	89.00	22.79	29.88	57.98	36.87	47.20	36.78	69.80	49.90
	SALF	73.00	40.50	91.40	26.47	31.10	57.98	38.89	51.60	40.67	71.00	52.26

A.5 Layer Budget Detailed Results

This section provides the detailed results underlying Section 4.3 and Figure 3. Table A6 reports benchmark accuracy, while Table A7 details the memory cost.

Table A6: Accuracy (%) on the ten benchmarks under GRPO for five different K on Qwen3-8B-Base. The rows $[12, 21]$ and Full repeat those of Table A5. Avg. is the unweighted mean across the ten benchmarks.

Training Interval I^*	Budget K	Math				Code		Reasoning		Language		Avg.
		M500	Oly.	GSM8K	Min.	HE+	MBPP	GP-D	M-Pro	IFEval	C-E	
$[16]$	1	74.68	42.26	93.48	30.51	30.24	72.37	36.57	51.52	45.36	73.36	55.04
$[15, 18]$	4	77.36	44.99	93.12	35.00	29.88	71.36	39.09	52.80	47.47	74.52	56.56
$[14, 20]$	7	78.64	44.87	93.84	34.56	31.10	72.68	39.29	54.76	46.58	75.96	57.23
$[12, 21]$	10	79.84	45.96	94.00	38.46	32.20	73.39	39.70	55.44	47.17	76.80	58.30
Full	36	79.52	46.35	94.44	39.12	34.76	73.07	40.91	58.08	45.51	76.64	58.84

Peak memory falls with the number of trained layers (Table A7). Post-training $[12, 21]$ reduces peak memory from 69.96 GB to 30.62 GB, a reduction of 56.2%. The gradient and the AdamW state of the trainable parameters account for most of the reduction: they take 49.14 GB under Full and 11.58 GB for ten layers. Below ten layers, the memory saved is small and the accuracy lost is large: from ten layers to four, peak memory falls by 4.8 GB and the mean accuracy by 1.7 points, and from four layers to one, peak memory falls by a further 2.3 GB only. Above ten layers, the memory added is large and the accuracy gained is small: from ten layers to thirty-six, peak memory rises by 39.3 GB and the mean accuracy by 0.5 points. The ten-layer interval $[12, 21]$ is the trade-off point between performance and peak memory.

Table A7: Mean accuracy over the ten benchmarks, trainable parameters, and peak memory on Qwen3-8B-Base against five different K. Reduction is relative to the peak memory of Full.

Training Interval I^*	Budget K	Trainable parameters	Mean accuracy (%) ↑	Peak memory (GB) ↓	Reduction
$[16]$	1	2.4%	55.04	23.48	66.4%
$[15, 18]$	4	9.4%	56.56	25.79	63.1%
$[14, 20]$	7	16.5%	57.23	28.11	59.8%
$[12, 21]$	10	23.6%	58.30	30.62	56.2%
Full	36	100%	58.84	69.96	–

A.6 BENCHMARK-LEVEL ONE-LAYER TRAINING RESULTS

Table A8 reports the benchmark-level results of the individual-layer post-training experiment shown in Figure 2 of Section 4.2.

Table A8: Accuracy (%) on ten benchmarks when each representative layer of Qwen3-8B-Base is post-trained alone under GRPO. Results are means over five evaluation runs.

Layer	M500	Oly.	GSM8K	Min.	HE+	MBPP	GP-D	M-Pro	IFEval	C-E
0	75.12	40.92	91.96	30.07	30.12	62.10	32.93	46.52	42.07	67.20
3	73.52	41.54	91.32	28.16	29.39	61.63	35.35	47.44	41.40	70.60
5	74.56	41.19	92.48	29.41	29.27	60.70	36.06	49.36	41.63	70.56
13	75.60	42.43	92.16	31.76	29.63	70.58	35.45	50.12	44.29	72.64
14	76.36	42.20	92.80	31.47	30.61	72.06	36.26	50.80	46.21	73.04
16	74.68	42.26	93.48	30.51	30.24	72.37	36.57	51.52	45.36	73.36
18	74.12	41.87	92.92	30.22	31.10	68.48	37.58	51.64	43.33	74.36
22	73.12	41.25	91.80	26.84	29.63	63.58	38.38	52.72	40.18	72.16
28	71.60	39.50	90.72	25.81	29.76	57.28	36.16	51.40	39.56	70.56
35	73.96	40.56	92.08	28.31	30.73	55.41	36.67	57.24	37.56	73.32