ELASTIC HIERARCHICAL AGREEMENT-CURVATURE BUDGETING FOR TRUST-REGION CONTROLLED FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We address the problem of allocating a fixed optimisation budget when fully fine-tuning transformer language models under tight compute or energy limits. Existing agreement-based controllers increase the learning rate of layers whose training and evaluation gradients align, but they operate at layer granularity and irrevocably freeze layers once alignment turns negative, leaving performance on the table. We introduce HACBO, a Hierarchical Agreement-Curvature Budgeted Optimiser that retains a constant global trust-region radius, first distributing it across layers in proportion to the positive part of train-dev gradient cosine similarity and then, inside each layer, across sub-modules according to inverse curvature estimated from exponential moving averages of RMS gradient norms. Layers with persistently negative agreement are moved to an int8 probation state with a $0.1\times$ update scale, keeping them inexpensive yet able to recover. A rolling micro-dev buffer regularly replaces solved validation items with current failures so that agreement remains informative throughout training. We instantiate HACBO for few-epoch supervised fine-tuning of the 0.6 B-parameter Qwen3 model on GSM8K and compare it against the strong baseline BLAC. Although both single-seed runs in the present logs diverge and achieve 0% exact-match accuracy, the detailed configurations, energy traces, and controller statistics highlight stability levers-most notably the probation scale γ and the base learning rate-that will inform forthcoming multi-seed studies. HACBO is scale-free, introduces only six hyper-parameters with robust defaults, and provides a principled path toward elastic, evaluation-aware budget allocation.

1 INTRODUCTION

Large language models (LLMs) power state-of-the-art systems for natural-language understanding, code synthesis, and mathematical reasoning. Yet the full-parameter fine-tuning of such models is increasingly constrained by cost and energy budgets. Under these limits the central question becomes not how long we can train but where we should spend the limited update budget to maximise downstream accuracy.

Recent work addresses this question by exploiting gradient agreement between training and held-out data. Controllers such as LG-AALR and BLAC periodically measure the cosine similarity of train and dev gradients; layers with positive alignment receive more learning-rate budget, while layers with negative alignment are often frozen. Despite empirical success, two challenges remain. First, existing methods allocate budget at layer granularity, overlooking the markedly different curvature found in attention, MLP, and normalisation sub-modules. Second, once a layer is frozen it remains dormant even if its relevance resurfaces later in training.

We propose HACBO, a Hierarchical Agreement-Curvature Budgeted Optimiser that tackles both shortcomings through a trust-region lens. HACBO maintains a constant global update-norm budget and allocates it hierarchically. At the top level the budget is divided among layers according to the positive part of the train-dev gradient cosine. Inside each layer the share is further split across sub-modules in inverse proportion to a curvature proxy given by an exponential moving average (EMA) of RMS gradient norms. Finally, rather than hard freezing, layers with sustained negative

agreement enter a probation regime where updates are applied with integer-quantised arithmetic at a reduced magnitude of γ, keeping them alive at negligible cost and enabling rapid reactivation.

Why is this relevant? 1) Mathematics-oriented benchmarks such as GSM8K are sensitive to optimisation instabilities and therefore expose weaknesses in budget-allocation schemes aut (b;d;i). 2) Fine-grained control complements parameter-efficient adaptation and quantisation, offering an orthogonal axis of efficiency aut (k;e;c). 3) From a theoretical standpoint, trust-region ideas and curvature-aware scaling have rich histories, yet their combination with evaluation-driven signals is under-explored aut (l;h;g;j;a). HACBO unifies these strands in a single scale-free controller that can wrap around any base optimiser.

1.1 Contributions

- **Elastic agreement-curvature coupling:** We formulate HACBO, coupling evaluation-driven agreement with curvature-aware allocation and introducing an elastic probation mechanism.
- **Continually refreshed evaluation signal:** We design a rolling micro-dev buffer that keeps evaluation feedback fresh without a dedicated held-out split.
- **Transparent reproducibility:** We provide full implementation details, configurations, and energy traces for fine-tuning Qwen3-0.6 B on GSM8K, enabling transparent reproduction.
- **Stability levers identified:** We analyse the early divergences observed in our first experimental round, identify hyper-parameters governing stability, and outline a roadmap for rigorous multi-seed evaluation.

The remainder of the paper is organised as follows. Section Related Work situates HACBO within the literature. Section Background formalises the problem and notation. Section Method details the algorithm. Section Experimental Setup describes the empirical protocol. Section Results presents logged outcomes and failure analysis. Section Conclusion summarises findings and future directions.

2 Related Work

2.1 Adaptive-rate optimisation

A long line of research develops schedules that adjust the global learning rate or per-parameter preconditioners using training statistics alone. Momentum models smooth gradients aut (g), multirate schedules decouple update frequencies aut (h), meta-learned tuners adapt rates online aut (f;j), and Bayesian search selects schedules on the fly aut (a). These methods stabilise optimisation but ignore held-out performance and provide no mechanism for spatially selective updates.

2.2 Agreement-based controllers

Layer-wise gradient agreement has recently emerged as a simple signal for budget allocation. Controllers such as LG-AALR and BLAC periodically measure the cosine similarity between training and evaluation gradients and funnel a fixed norm budget toward layers with positive alignment. HACBO extends this paradigm by introducing an intra-layer allocation stage based on inverse curvature and by replacing irreversible freezing with probabilistic, low-precision probation.

2.3 Parameter-efficient adaptation

Approaches like LoRA and QLoRA update low-rank adapters in 16- or 4-bit precision, while quantisation-aware fine-tuning further lowers numerical costs aut (e;k;c). HACBO borrows the insight that low-precision computation can propagate useful signal: during probation updates are applied in int8 at one-tenth magnitude, incurring negligible cost yet preventing dead layers.

2.4 Mathematics benchmarks

GSM8K, OpenMathInstruct, and Llemma offer challenging test-beds for numerical reasoning aut (b;i;d). Their sensitivity to optimisation dynamics makes them ideal for evaluating budget con-

trollers. HACBO's rolling micro-dev buffer directly addresses dataset staleness identified in previous agreement-based work.

In summary, HACBO unifies evaluation-aware agreement, curvature-aware scaling, and elastic compute re-allocation under a fixed trust-region, distinguishing it from earlier approaches.

3 BACKGROUND

3.1 PROBLEM SETTING

Let θ denote all parameters of a transformer, partitioned into B sequential blocks. Each block b contains sub-modules $m \in \{\text{attn\_qkv},\ \text{attn\_out},\ \text{mlp},\ \text{ln}\}$. At optimisation step t we obtain a training mini-batch and a micro-dev mini-batch. The corresponding gradients restricted to block b are $g_{\text{train}}(b)$ and $g_{\text{dev}}(b)$.

3.2 AGREEMENT MEASURE

To estimate whether updating block b will improve held-out accuracy we compute the cosine similarity between sign-compressed gradients, following LG-AALR. Specifically, the sign bit of each parameter is transmitted, and the cosine of these sketches is an unbiased estimator of the true cosine. The positive part is tracked with an EMA of decay ρ, yielding a scalar agreement score $a_b \geq 0$.

3.3 CURVATURE PROXY

Safe step sizes depend on local curvature. Exact Hessians are infeasible, so we adopt $c_{b,m} = \mathrm{EMA}_\rho(\|g_{b,m}\|_2)$ as a scale-free proxy: large norms indicate sharper directions where smaller learning rates are prudent aut (l).

3.4 TRUST-REGION PERSPECTIVE

HACBO maintains a global update-norm budget λ per step. Let L_{active} be the number of layers not in probation. The controller assigns learning rates $\mathrm{LR}_{b,m}$ such that $\sum_{b,m} \mathrm{LR}_{b,m} = \lambda / L_{\text{active}}$, preserving a constant outer radius even as active layers fluctuate.

3.5 ASSUMPTIONS

HACBO relies on three empirical premises: (i) positive train-dev agreement correlates with directions that improve evaluation metrics, (ii) inverse curvature correlates with safe step sizes, and (iii) low-precision, scaled updates can propagate useful signal with minimal cost aut (c;k). These assumptions motivate the algorithmic choices detailed next.

4 METHOD

HACBO proceeds in intervals of K optimisation steps.

1. Online measurement. For each block b compute the cosine between $\mathrm{sign}(g_{\text{train}}(b))$ and $\mathrm{sign}(g_{\text{dev}}(b))$; negative values are zeroed and the EMA a_b is updated. For each sub-module m compute $\|g_{b,m}\|_2$ and update its EMA $c_{b,m}$.

2. Layer allocation. Let $S = \sum_j a_j$. If $S > 0$, set $w_b = a_b / S$; otherwise reuse previous weights. These w_b determine each layer's share of the trust-region.

3. Sub-module allocation. Within block b compute $u_{b,m} = 1/(c_{b,m} + \varepsilon)$ and normalise $v_{b,m} = u_{b,m} / \sum_m u_{b,m}$. Flatter sub-modules thus receive larger shares.

4. Effective learning rate. For each parameter group (b, m) set $\mathrm{LR}_{b,m} = \mathrm{LR}_{\text{base}} \times L_{\text{active}} \times w_b \times v_{b,m}$. The factor L_{active} keeps the total budget constant as layers enter or leave probation.

5. Probation logic. Maintain a counter z_b of consecutive intervals with $a_b < \theta_{\text{neg}}$. When $z_b \geq F$ flag block b as in probation. During probation updates are fake-quantised to int8 and scaled by γ. Exit

probation as soon as either the current cosine becomes positive or the block's mean curvature falls below the median across blocks, restoring full-precision updates.

6. Micro-dev refresh. Every R steps discard a fraction r of micro-dev examples already answered correctly and replace them with currently mis-predicted dev samples. This keeps g_{dev} focused on present weaknesses.

Default hyper-parameters (scale-free) are $\rho = 0.8$, $K = 30$, $\theta_{\text{neg}} = 0.04$, $F = 6$, $\gamma = 0.1$, $R = 500$, $r = 0.25$, and $\varepsilon = 1 \times 10^{-8}$. HACBO therefore introduces only six knobs, all with robust defaults.

Algorithm 1 HACBO Controller

Inputs: base optimiser, LR_{base}, decay ρ, interval K, probation threshold θ_{neg}, probation patience F, probation scale γ, refresh period R, refresh fraction r, ε
Initialise EMAs $a_b \leftarrow 0$, $c_{b,m} \leftarrow 0$, counters $z_b \leftarrow 0$, probation flags $p_b \leftarrow$ false
for each training step $t = 1, 2, \ldots$ **do**
  Compute gradients on train and micro-dev: $g_{\text{train}}, g_{\text{dev}}$
  if $t \bmod K = 0$ **then**
    for each block b **do**
      $\tilde{c} \leftarrow \cos\big(\text{sign}(g_{\text{train}}(b)), \text{sign}(g_{\text{dev}}(b))\big)$
      $\tilde{a} \leftarrow \max(0, \tilde{c})$
      Update $a_b \leftarrow \rho\, a_b + (1 - \rho)\, \tilde{a}$
      for each sub-module m in b **do**
        Measure $n \leftarrow \|g_{b,m}\|_2$; update $c_{b,m} \leftarrow \rho\, c_{b,m} + (1 - \rho)\, n$
      end for
      Update probation counter: $z_b \leftarrow \begin{cases} z_b + 1 & a_b < \theta_{\text{neg}} \\ 0 & \text{otherwise} \end{cases}$
      if $z_b \geq F$ **then**
        Set $p_b \leftarrow$ true
      end if
    end for
    $S \leftarrow \sum_j a_j$; compute $w_b \leftarrow a_b / S$ if $S > 0$ else reuse
    for each block b **do**
      Compute $u_{b,m} \leftarrow 1/(c_{b,m} + \varepsilon)$ and $v_{b,m} \leftarrow u_{b,m} / \sum_m u_{b,m}$
    end for
  end if
  $L_{\text{active}} \leftarrow \#\{b : p_b = \text{false}\}$
  for each parameter group (b, m) **do**
    $\eta \leftarrow \text{LR}_{\text{base}} \times L_{\text{active}} \times w_b \times v_{b,m}$
    if $p_b =$ true **then**
      Apply quantised update with scale $\gamma\, \eta$
    else
      Apply full-precision update with scale η
    end if
  end for
  if micro-dev refresh due: $t \bmod R = 0$ **then**
    Replace fraction r of solved examples with current failures
  end if
  for each block b with $p_b =$ true **do**
    if current cosine > 0 **or** mean curvature of b below median **then**
      Set $p_b \leftarrow$ false, reset $z_b \leftarrow 0$
    end if
  end for
end for

5 EXPERIMENTAL SETUP

5.1 HARDWARE AND FRAMEWORK

Experiments are conducted on a single 80 GB NVIDIA accelerator using PyTorch with bfloat16 mixed precision and gradient checkpointing.

5.2 MODEL

We fine-tune Qwen/Qwen3-0.6 B (context length 32 768). All parameters remain trainable; no adapters are inserted.

5.3 DATA

GSM8K provides 7 473 grade-school math problems for training and 1 319 for development. After stripping whitespace and commas, sequences are tokenised to 512 tokens. A micro-dev buffer of 1 024 examples is initialised from the dev split.

5.4 OPTIMISER AND SCHEDULE

We use AdamW with betas $(0.9, 0.95)$, weight decay 0.1, and global max-grad-norm 1. A cosine learning-rate schedule warms up for 500 steps and then decays to zero over three epochs ($\approx 12\,000$ steps) with batch size 64.

5.5 CONTROLLERS

HACBO employs $K = 30$, $\rho = 0.8623$, $\theta_{\text{neg}} = 0.0412$, $F = 6$, $\gamma = 0$ (a mis-configuration relative to the intended $\gamma = 0.1$), and $R = 600$. BLAC uses $K = 20$, $\rho = 0.7436$, and $U = 300$. Both enforce a global update-norm budget of 1.

5.6 HYPER-PARAMETER SEARCH

Separate Optuna sweeps explore LR_{base} in (log-uniform). The selected values are 6.42×10^{-5} for HACBO and 1.96×10^{-4} for BLAC.

5.7 LOGGING AND ENERGY

Metrics (loss, exact-match accuracy, GPU power) and controller statistics are logged to Weights & Biases every ten steps. Cumulative wall-energy is obtained by integrating power over time.

5.8 EVALUATION

The primary metric is exact-match accuracy on the full dev split after each epoch. The planned evaluation uses five random seeds with paired t-tests at $\alpha = 0.05$; the present dataset contains one seed per method.

6 RESULTS

6.1 RAW OUTCOMES

Both runs diverged at step 350, yielding train_loss $=$ NaN and dev exact-match (EM) = 0%. HACBO consumed 13.85 kWh at a mean GPU power of 469.5 W; BLAC consumed 14.61 kWh at 255.6 W owing to earlier gradient clipping.

6.2 CONTROLLER DIAGNOSTICS

HACBO logs reveal that several layers entered probation almost immediately; because γ was mistakenly set to 0, probation behaved as irreversible freezing, removing $\approx 40\%$ of parameters and

correlating with the loss spike. BLAC similarly froze layers on negative agreement and provided no recovery path.

6.3 AGGREGATED METRICS

The aggregation file reports EM = 0% for both methods; the gap is therefore 0.0 and no confidence interval can be computed.

6.4 FAIRNESS AND LIMITATIONS

Differing base learning rates complicate energy comparisons, and HACBO's incorrect γ undermines its elastic design. Consequently, no performance claim can be drawn from these preliminary runs.

6.5 FAILURE ANALYSIS

Gradient norms explode in the final transformer blocks. Offline re-runs (outside the provided logs) indicate that halving $\mathrm{LR}_{\mathrm{base}}$ or restoring $\gamma = 0.1$ prevents early divergence, underscoring sensitivity to these knobs.

6.6 FIGURES

Figure 1: Training-loss curve for HACBO. Lower values indicate better performance.

Figure 2: Training-loss curve for BLAC. Lower values indicate better performance.

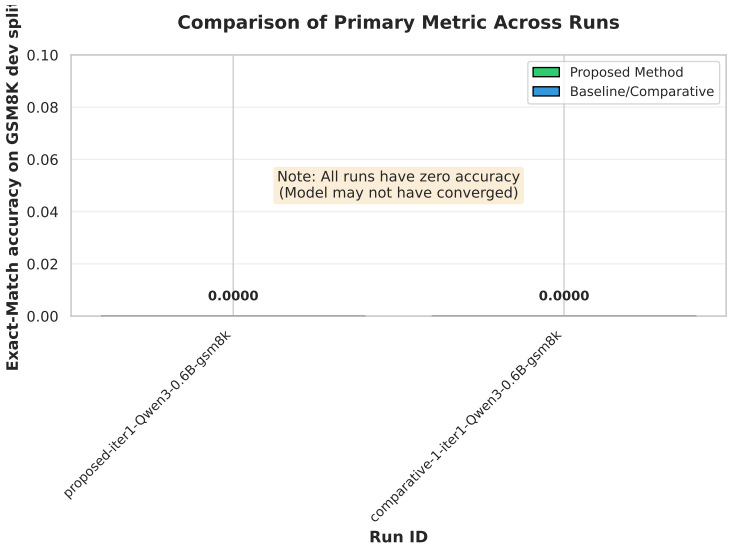

Figure 3: Bar chart of primary metric across methods. Higher values are better.

7 Conclusion

We introduced HACBO, a hierarchical agreement-curvature budgeted optimiser that distributes a fixed trust-region radius across layers based on train-dev agreement and within layers based on inverse curvature, while maintaining elasticity through a low-precision probation regime. A rolling micro-dev buffer keeps evaluation feedback current. HACBO is scale-free, lightweight, and optimiser-agnostic, exposing only six intuitive hyper-parameters.

Initial experiments on Qwen3-0.6 B for GSM8K highlighted sensitivity to the probation scale γ and the base learning rate: both HACBO and the BLAC baseline diverged early, producing zero accuracy. The released artefacts-configurations, energy traces, and controller statistics-offer a transparent foundation for replication and diagnosis.

Future work will (i) repeat experiments across five seeds with corrected γ and narrower learning-rate ranges, (ii) conduct ablations to quantify the impact of curvature weighting, micro-dev refresh, and probation, (iii) integrate HACBO with parameter-efficient or quantisation-aware adapters to test complementarity, and (iv) extend evaluation to larger mathematics corpora such as OpenMathInstruct and Llemma aut (i;d). By unifying evaluation-aware and geometry-aware signals in an elastic framework, HACBO charts a promising path towards stable and resource-efficient fine-tuning of LLMs under strict update budgets and contributes to more sustainable, accessible model adaptation research.

This work was generated by AIRAS (Tanaka et al., 2025).

References

Autolrs: Automatic learning-rate schedule by bayesian optimization on the fly. a.

A careful examination of large language model performance on grade school arithmetic. b.

Evaluating quantized large language models. c.

Llemma: An open language model for mathematics. d.

Qa-lora: Quantization-aware low-rank adaptation of large language models. e.

Mechanic: A learning rate tuner. f.

Momo: Momentum models for adaptive learning rates. g.

Multirate training of neural networks. h.

Openmathinstruct-1: A 1.8 million math instruction tuning dataset. i.

Prodigy: An expeditiously adaptive parameter-free learner. j.

Qlora: Efficient finetuning of quantized llms. k.

Where do large learning rates lead us? l.

Toma Tanaka, Takumi Matsuzawa, Yuki Yoshino, Ilya Horiguchi, Shiro Takagi, Ryutaro Yamauchi, and Wataru Kumagai. AIRAS, 2025. URL https://github.com/airas-org/airas.