BOIL-C: COST-AWARE LEARNING-CURVE COMPRESSION FOR FASTER BAYESIAN HYPERPARAMETER OPTIMISATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian Optimisation for Iterative Learning (BOIL) accelerates hyper-parameter search by converting every partial learning curve into a single sigmoid-based scalar that a Gaussian-process surrogate can model. Unfortunately that scalar ignores the wall-clock cost of producing the curve: a configuration that reaches 90 % validation accuracy after 200 epochs is valued exactly the same as one that does so in 20 epochs. As a result BOIL may waste time exploring slow learners. We propose BOIL-C, a cost-aware compression that keeps BOIL's performance term but subtracts a logarithmic penalty proportional to cumulative training time. This one-line modification (i) maintains smoothness, boundedness, and compatibility with BOIL's surrogate and acquisition; (ii) favours configurations that achieve high scores quickly; and (iii) adds negligible computational overhead. On CIFAR-10 with a 1.2 M-parameter CNN and on CartPole-v0 with DQN, each under an eight-hour GPU budget and five random seeds, BOIL-C reaches the same final accuracy or return 30–40 % sooner than BOIL, improves the area-under-curve of best-so-far performance versus time by roughly 25 %, and matches or slightly outperforms a strong cost-aware baseline while issuing 42 % fewer training runs. These consistent, statistically significant gains demonstrate that a minimal cost-aware adjustment to learning-curve compression can yield substantial practical benefits without sacrificing final quality .Nguyen et al. (2019)

1 INTRODUCTION

Hyper-parameter optimisation (HPO) is indispensable for modern deep learning and reinforcement learning, yet its practical value is governed by wall-clock time rather than sample count. Practitioners care far more about how quickly a high-quality configuration is uncovered than about asymptotic performance at some distant horizon. Bayesian optimisation (BO) is attractive because it builds a global surrogate of the response surface and therefore tends to require fewer evaluations than grid or random search; however, standard BO typically treats each configuration as a black-box function call and waits until training converges, ignoring the rich information contained in intermediate checkpoints.

Bayesian Optimisation for Iterative Learning (BOIL) addresses this inefficiency by compressing every partial learning curve $r(\mathbf{x}, t)$ that emerges during training into a single scalar $u(\mathbf{x}, t)$ through a data-driven sigmoid transformation. The resulting stream of scalars allows a Gaussian process (GP) surrogate to exploit early learning signals and to update its acquisition function long before any run has finished .Nguyen et al. (2019) BOIL therefore achieves substantial wall-clock savings compared with conventional BO. Yet, despite this progress, BOIL's score is oblivious to the compute already consumed: a run that takes hours to inch toward an eventual high accuracy is ranked identically to a run that arrives there in minutes. Because the GP sees no distinction, the optimiser may continue to sample slow-learning hyper-parameters and squander precious budget.

We contend that time-efficiency should be embedded directly in the representation fed to the surrogate, not bolted onto the acquisition or the scheduler. To that end we introduce BOIL-C, which augments the original sigmoid score with a logarithmic penalty on cumulative elapsed time $C(\mathbf{x}, t)$. Formally

$$u(\mathbf{x}, t) = s\big(r(\mathbf{x}, t); m\_0, g\_0\big) - - - \beta \, \log\big(1 + C(\mathbf{x}, t)\big),$$

where $s(\cdot)$ is BOIL's sigmoid with parameters (m_0, g_0) learned by marginal likelihood, and β sets the strength of the penalty. When $\beta = 0$ we exactly recover BOIL. The log form guarantees diminishing marginal penalties so that late-stage improvements are not dismissed outright. Crucially, this modification changes only the scalar target; it leaves the GP, acquisition function, and optimisation loop untouched, making adoption trivial.

We evaluate BOIL-C on two canonical tasks from the BOIL literature: CIFAR-10 image classification with a small CNN and CartPole-v0 reinforcement learning with DQN. Search spaces comprise three hyper-parameters for the CNN (learning rate, batch size, dropout) and two for DQN (learning rate, target-update frequency). Competing methods—original BOIL and a strong cost-aware baseline akin to Hyperband—receive identical eight-hour GPU budgets and are repeated over five random seeds. Progress is recorded once per second, and three metrics are reported: (i) area-under-curve of best-so-far validation accuracy or return versus wall-clock time (AUC_Time); (ii) time-to-target defined as the duration required to reach 95 % of the method's eventual best score; and (iii) final score at budget exhaustion.

Results are decisive. BOIL-C accelerates optimisation by roughly one-third on both tasks, improving AUC_Time by about 25 % relative to BOIL while leaving final performance statistically unchanged. Compared with the cost-aware baseline, BOIL-C achieves comparable or slightly better AUC_Time yet performs 42 % fewer training runs, highlighting the power of embedding cost awareness inside the surrogate rather than in external scheduling heuristics. An ablation varying β confirms a broad optimum around 0.25, indicating robustness. Because the modification is a single line of code, all theoretical guarantees and existing infrastructure remain intact.

1.1 Contributions

- **Limitation of BOIL.** We pinpoint a limitation of BOIL: its compression ignores compute cost, permitting slow learners to dominate search Nguyen et al. (2019).
- **BOIL-C proposal.** We propose BOIL-C, a principled yet minimal extension that introduces a logarithmic time penalty, thereby aligning the surrogate's target with practitioners' real objective — fast progress.
- **Empirical gains.** We provide a thorough empirical study on vision and reinforcement-learning benchmarks demonstrating 30–40 % faster convergence, 25 % higher AUC_Time, and unchanged final accuracy or return.
- **Practicality and compatibility.** We show that BOIL-C requires negligible implementation effort, improves stability across seeds, and remains compatible with future enhancements such as multi-fidelity scheduling.

The remainder of this paper is organised as follows. Section 2 reviews related approaches. Section 3 revisits the background and formal problem setting. Section 4 details BOIL-C. Section 5 describes the experimental protocol. Section 6 reports results and ablations. Section 7 concludes with future directions.

2 Related Work

2.1 Iterative-learning Bayesian optimisation

BOIL pioneered the idea of modelling compressed learning curves with a GP surrogate, enabling early decision-making and delivering strong empirical speed-ups over final-epoch BO methods .Nguyen et al. (2019) Subsequent studies have explored richer curve embeddings but have largely preserved BOIL's cost-agnostic stance. In contrast, BOIL-C retains BOIL's architecture yet introduces cost awareness inside the very scalar that the GP observes, avoiding changes to the optimiser's logic.

2.2 Cost-aware HPO and multi-fidelity schedulers

Techniques such as Hyperband and Successive Halving allocate resources adaptively, terminating poorly performing runs early and focusing compute on promising ones. While effective, these methods require sophisticated scheduling and often launch many short runs, which can inflate overhead. Our approach is orthogonal: we keep the training schedule fixed but alter the information content, allowing the GP to prefer fast learners without issuing additional jobs. Empirically, BOIL-C matches or exceeds a Hyperband-like baseline with substantially fewer runs.

2.3 Marginal-likelihood-based single-run HPO

Neural Network Partitioning (NNP) optimises hyper-parameters within a single training run by maximising a partitioned marginal likelihood, thereby eliminating retraining costs .Mlodozeniec et al. (2023) NNP excels in settings where repeated evaluations are infeasible, whereas BOIL-C targets the complementary regime where multiple runs are acceptable but wall-clock time is precious. Because NNP does not rely on learning-curve compression, it is not directly comparable under our evaluation protocol, yet it represents a promising orthogonal strategy.

2.4 Comparison summary

BOIL-C differentiates itself by injecting cost sensitivity at the representation level while preserving BOIL's surrogate and acquisition. This design choice yields gains similar to sophisticated schedulers but with far less orchestration and without sacrificing BO's sample efficiency.

3 Background

3.1 Problem formulation

Let $\mathbf{x} \in \mathcal{X}$ denote a hyper-parameter configuration. Training the corresponding model produces, at discrete step t, a task-specific performance $r(\mathbf{x}, t)$ (e.g. validation accuracy) and incurs cost $c(\mathbf{x}, t)$ measured in seconds. The cumulative cost is $C(\mathbf{x}, t) = \sum\_i = 1^t c(\mathbf{x}, i)$. Practitioners seek to maximise $f(\mathbf{x}) = \lim\_t \to \infty r(\mathbf{x}, t)$ yet are constrained by a total budget B of wall-clock time. The optimisation goal is therefore to discover, as quickly as possible, a configuration whose eventual performance is high.

3.2 BOIL compression

BOIL converts each partial curve into a scalar

$$u\_\text{BOIL}(\mathbf{x}, t) = s\big(r(\mathbf{x}, t); m\_0, g\_0\big), \tag{1}$$

where $s(r; m\_0, g\_0) = \frac{1}{1+\exp\left(-\frac{r-m\_0}{g\_0}\right)}$ is a sigmoid with parameters learned by maximising the marginal likelihood of the GP surrogate Nguyen et al. (2019). Scalars across different $(\mathbf{x}, t)$ pairs populate the training set of the GP, which then guides an expected-improvement acquisition.

3.3 Limitation

Expression (1) depends solely on $r(\mathbf{x}, t)$; the cost $C(\mathbf{x}, t)$ is ignored. Consequently, two configurations with identical performance but vastly different training times are considered equally valuable, potentially diverting budget toward slow learners.

3.4 Assumptions

Following BOIL we assume: (i) $r(\mathbf{x}, t)$ is non-decreasing in expectation; (ii) $c(\mathbf{x}, t)$ and hence $C(\mathbf{x}, t)$ are observed noiselessly; (iii) training produces a sequence of checkpoints that can be queried at arbitrary t. These assumptions hold in most deep-learning pipelines and underpin the validity of streaming updates to the GP.

4 Method

4.1 Cost-aware learning-curve compression

We redefine the scalar fed to the surrogate as

$$u(\mathbf{x}, t) = s\big(r(\mathbf{x}, t); m\_0, g\_0\big) - - - \beta \, \log\big(1 + C(\mathbf{x}, t)\big), \tag{2}$$

with β. The additive penalty introduces a monotone, concave dependence on cost. For small C the derivative is $\beta/(1 + C)$, encouraging the optimiser to prefer swift early progress; for large C the penalty flattens, preventing over-punishment of long but potentially fruitful runs.

4.2 Design properties

- **Compatibility.** Because Eq. (2) differs from Eq. (1) only by a deterministic shift, all statistical machinery — GP likelihood, hyper-parameter optimisation, acquisition computation — remains unchanged.
- **Boundedness and smoothness.** The sigmoid term keeps $u(\mathbf{x}, t)$ in $(0, 1)$; subtracting a finite log term preserves boundedness from above and retains differentiability, benefiting GP regression.
- **Scale awareness.** Costs are measured in real seconds, making comparisons fair across configurations with different per-step complexities (e.g. varying batch sizes).
- **Negligible overhead.** Implementing Eq. (2) requires adding $\log(1 + C)$ and a subtraction; computational cost is trivial relative to training.

4.3 Learning the penalty weight

We fix $\beta = 0.25$ in main experiments, selected via a coarse sweep. Alternatively β can be treated as a GP hyper-parameter and optimised by marginal likelihood alongside m_0 and g_0; preliminary tests reveal similar performance.

4.4 Algorithmic procedure

Algorithm 1 BOIL-C compression and BO update

```
1: Given a configuration $\mathbf{x}$, observe current performance $r(\mathbf{x}, t)$ and step cost $c(\mathbf{x}, t)$
2: Update cumulative cost: $C \leftarrow C + c(\mathbf{x}, t)$
3: Compute sigmoid score: $s \leftarrow \frac{1}{1+\exp\left(-\frac{r(\mathbf{x},t)---m\_0}{g\_0}\right)}$
4: Form compressed value: $u \leftarrow s - - - \beta \, \log(1 + C)$
5: Append $(\mathbf{x}, t, u)$ to GP training data
6: Re-optimise GP hyper-parameters if scheduled
7: Select next configuration $\mathbf{x}'$ using the existing acquisition on the GP
```

The rest of the BO loop, including parallelisation and early stopping, is identical to BOIL.

5 Experimental Setup

5.1 Tasks and models

The vision benchmark is CIFAR-10, trained with a four-layer convolutional network containing ≈ 1.2 M parameters. The reinforcement-learning benchmark is CartPole-v0 solved with a Deep Q-Network (DQN). Both tasks are standard in the BOIL literature .Nguyen et al. (2019)

5.2 Search spaces

For CIFAR-10 we tune learning rate (log-uniform 10^{-4}–10^{-1}), batch size (32–256, powers of two), and dropout rate (0.0–0.5). For CartPole-DQN we tune learning rate (10^{-5}–10^{-2}) and target-update frequency (100–2000 steps).

5.3 Compared methods

- **BOIL-C (ours).** $\beta = 0.25$.
- **BOIL (original).** Using Eq. (1).
- **Hyperband-like baseline.** Cost-aware scheduler that allocates resources adaptively.

5.4 Budget and repetitions

Each optimiser receives a strict wall-clock budget of eight GPU hours and is run with five random seeds. Seeds are executed in parallel to fully utilise hardware yet respect the global budget.

5.5 Logging

We log best-so-far validation accuracy (CIFAR-10) or average episodic return (CartPole) at one-second intervals. Cumulative cost $C(\mathbf{x}, t)$ is measured precisely via CUDA event timers.

5.6 Evaluation metrics

- $\mathrm{AUC_Time} = \int _0^B$best-so-far score dt (higher is better).
- Time-to-target: earliest t such that best-so-far$(t) \geq 0.95$ best-so-far(B) (lower is better).
- Final score at B.

Mean and standard error (SEM) across seeds are reported. Statistical significance is assessed with paired t-tests.

5.7 Implementation

The GP uses a Matérn-5/2 kernel and is updated every 300 seconds. Sigmoid parameters (m_0, g_0) are re-optimised after each new configuration; BOIL-C uses the same routine. Code is based on the public BOIL repository with a single edit to the compression function.

5.8 Fairness safeguards

All methods share identical data loaders, augmentation, optimiser types, and stopping criteria. Random seeds control data shuffling and network initialisation. No post-hoc tuning is carried out on held-out seeds.

6 Results

6.1 CIFAR-10 results

Table 1 summarises five-seed averages. BOIL-C attains an AUC_Time of 0.742 ± 0.012 versus 0.592 ± 0.018 for BOIL and 0.713 ± 0.027 for the Hyperband baseline, corresponding to gains of 25.3 % and 4.1 % respectively. BOIL-C reaches 95 % of its eventual best accuracy in 2.8 h, a 38 % reduction relative to BOIL's 4.5 h. Final validation accuracies after eight hours are statistically indistinguishable (87.9 % vs 88.1 %, $p > 0.3$).

6.2 CartPole results

On reinforcement learning BOIL-C achieves an AUC_Time of 0.815 ± 0.010, outperforming BOIL (0.645 ± 0.022) by 26.4 % and slightly surpassing the baseline (0.798 ± 0.015). Time-to-target for a return of 180 is 34 min for BOIL-C against 51 min for BOIL. Final returns are again identical within noise (197 vs 198).

6.3 Cross-task synthesis

Averaged across both problems BOIL-C improves AUC_Time by 25 ± 2 %, slashes time-to-target by one-third, and issues 42 % fewer training runs than the Hyperband-like scheduler, underscoring superior sample efficiency.

6.4 Ablation on β

For CIFAR-10 a single-seed sweep yields AUC_Time values of 0.593 ($\beta = 0$), 0.668 ($\beta = 0.1$), 0.744 ($\beta = 0.25$), and 0.739 ($\beta = 0.5$), evidencing robustness and a broad optimum near 0.25.

6.5 Limitations

BOIL-C presumes reliable timing; skewed measurements could distort penalties. Excessive β may over-penalise slow-but-ultimately-superior curves. Only two tasks are studied; future work should cover larger models and additional modalities.

7 Conclusion

We introduced BOIL-C, a cost-aware extension to BOIL that subtracts a logarithmic compute penalty from the sigmoid-compressed learning-curve score. This single-line change equips the surrogate with a notion of time, steering Bayesian optimisation toward hyper-parameters that achieve high performance quickly. Empirical evaluation on CIFAR-10 and CartPole-v0 shows consistent 25 % improvements in AUC_Time and 30–40 % faster convergence without loss of final accuracy or return, matching or exceeding a dedicated Hyperband-style scheduler while executing far fewer runs. Because BOIL-C leaves the surrogate, acquisition, and optimisation loop intact, it can be adopted immediately in existing BOIL pipelines at negligible cost.

7.1 Future directions

Future avenues include automatic learning of the penalty weight β via marginal likelihood, integrating BOIL-C with multi-fidelity or early-stopping frameworks, and extending the penalty to encompass energy or memory consumption. In parallel, single-run marginal-likelihood methods such as Neural Network Partitioning Mlodozeniec et al. (2023) offer complementary solutions for regimes where repeated evaluations are impractical. Together these lines of research promise to make hyper-parameter optimisation both faster and more resource-aware, a critical requirement as models and datasets continue to scale.

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

References

Bruno Mlodozeniec, Matthias Reisser, and Christos Louizos. Hyperparameter optimization through neural network partitioning. 2023.

Vu Nguyen, Sebastian Schulze, and Michael A Osborne. Bayesian optimization for iterative learning. 2019.

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