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

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian optimisation for iterative learners commonly exploits partially observed learning curves by compressing them into scalar utilities for a surrogate model. In the original BOIL framework this compression ignores wall-clock cost, so runs that attain identical accuracy receive identical utility even when one is an order of magnitude slower Nguyen et al. (2019). That bias wastes compute under fixed time budgets. We introduce BOIL-C, a one-line modification that subtracts a logarithmic penalty from the BOIL score: $u(x,t) = s(r(x,t); m_0, g_0) - \beta \cdot \log(1 + C(x,t))$, where $s(\cdot)$ is BOIL's sigmoid score, $C(x,t)$ the cumulative observed training cost, and $\beta \in$ a small weight. The surrogate, acquisition function and optimisation loop remain untouched, so BOIL-C is a drop-in change that favours hyper-parameters which reach good accuracy quickly. On CIFAR-10 with a 1.2 M-parameter CNN and on CartPole-v0 with DQN we compare BOIL-C to BOIL and (for RL) to Hyperband. With an 8 GPU-hour budget and five seeds BOIL-C matches or slightly improves final accuracy yet raises the area-under-best-so-far-versus-time by $\approx 25\%$ on CIFAR-10, cuts median time-to-90% accuracy by 31%, and reaches the RL success threshold with 28% fewer interaction steps than BOIL, on par with Hyperband while evaluating fewer configurations. An ablation over β reveals a broad, easy-to-tune regime. These findings show that a principled, cost-aware scalar yields meaningful time savings without sacrificing model quality.

1 INTRODUCTION

Hyper-parameter optimisation (HPO) for deep learning remains expensive because each evaluation entails running an iterative algorithm such as SGD or Q-learning to near convergence. Bayesian optimisation (BO) techniques that exploit partial learning curves relieve this burden by extracting signal early and steering the search accordingly. BOIL compresses every curve prefix into a single scalar reflecting performance and stability, feeds those scalars to a Gaussian-process surrogate, and selects the next configuration via an acquisition function Nguyen et al. (2019).

A blind spot of BOIL is cost agnosticism: two runs that ultimately reach the same accuracy receive the same utility regardless of how long they took. In practice this biases BO toward slow but eventually strong configurations, delaying progress under wall-clock constraints. We therefore ask: can BOIL's compression be made cost-aware with minimal disruption to the rest of the pipeline?

The challenge is twofold. First, any penalty must be generic applicable across tasks and metrics and must preserve the desirable monotonicity of BOIL's score with respect to performance. Second, the modification should neither destabilise the surrogate nor require re-engineering downstream components.

We propose BOIL-C, which augments BOIL's score with a logarithmic cost penalty. For configuration x after t steps we define $u(x,t) = s(r(x,t); m_0, g_0) - \beta \cdot \log(1 + C(x,t))$, where $C(x,t)$ is cumulative wall-clock seconds and β controls the trade-off. The log ensures diminishing penalties, encouraging rapid progress early without discarding promising but slower curves. Setting $\beta = 0$ recovers BOIL exactly, hence BOIL-C is a strict generalisation.

We evaluate BOIL-C on two contrasting domains: image classification on CIFAR-10 with a modest CNN and reinforcement learning on CartPole-v0 with DQN. All methods BOIL-C, BOIL and (for RL) Hyperband receive identical 8 GPU-hour budgets and are run with five independent seeds. Efficiency

is measured by integrating the best-so-far validation metric over wall-clock time (AUC-Time). BOIL-C preserves or slightly betters final performance while substantially improving AUC-Time and reducing time-to-threshold.

1.1 CONTRIBUTIONS

- **One-line cost-aware compression**: A cost-aware learning-curve compression that adds a single logarithmic term to BOIL's scalar, requiring only one line of code.
- **Guidance for β**: A theoretical justification and practical guidance for choosing β.
- **Empirical gains**: Comprehensive experiments on CIFAR-10 and CartPole-v0 showing $\approx$ 25-30% faster time-to-result without loss of accuracy or return.
- **Positioning among HPO methods**: A discussion placing BOIL-C among cost-aware HPO strategies and contrasting it with partition-based objectives Mlodozeniec et al. (2023).

Future work includes adaptive β schedules, integration with multi-fidelity schedulers and evaluation on larger-scale tasks.

2 RELATED WORK

2.1 ITERATIVE-LEARNING BO

BOIL is a seminal example of using partial curves within BO Nguyen et al. (2019). Follow-up work has explored hierarchical surrogates and curve extrapolation yet still compresses curves independently of cost. BOIL-C differs by embedding cost directly in the scalar target, leaving the surrogate unchanged.

2.2 COST-AWARE SCHEDULERS

Hyperband and successive-halving allocate resources adaptively based on intermediate results, implicitly trading accuracy against compute. Those methods act at the scheduler level, whereas BOIL-C embeds cost inside the surrogate target, making it orthogonal and in principle combinable with schedulers.

2.3 ALTERNATIVE OBJECTIVES

Neural network partitioning optimises hyper-parameters via an out-of-sample loss computed from parameter/data shards, bypassing validation data entirely Mlodozeniec et al. (2023). While elegant, that objective assumes a single training run and different statistical properties; it is therefore not directly comparable to BO's iterative evaluation paradigm adopted here.

3 BACKGROUND

3.1 PROBLEM SETTING

Let x denote a hyper-parameter configuration and $r(x, t)$ the validation metric after t training steps (epochs for supervised learning, environment interactions for RL). Each step incurs cost $c(x, t)$ seconds, and cumulative cost is $C(x, t) = \sum_{i \le t} c(x, i)$. We wish to maximise r subject to a fixed wall-clock budget.

3.2 BOIL'S COMPRESSION

BOIL maps each partial curve to $s(r(x, t); m_0, g_0) = \frac{1}{1+\exp(-(r-m_0)/g_0)}$, where m_0 and g_0 are learnt hyper-parameters, and feeds that scalar to a Gaussian-process surrogate. The acquisition function (expected improvement, EI) balances exploration and exploitation when proposing either to extend an existing run or start a new one.

3.3 LIMITATIONS

Because s depends only on r, the surrogate undervalues fast learners and overvalues slow ones that will eventually perform well, delaying progress when the budget is tight. Addressing this deficiency motivates BOIL-C.

4 METHOD

4.1 COST-AWARE COMPRESSION

BOIL-C defines the utility

$$u(x,t) = s\big(r(x,t); m_0, g_0\big) - \beta \cdot \log\big(1 + C(x,t)\big).$$

The logarithm ensures u remains bounded and yields diminishing penalties: a second of cost early is penalised more than a second late. β may be fixed (0.25 in our experiments) or optimised jointly with m_0, g_0 by marginal likelihood.

4.2 ALGORITHMIC INTEGRATION

Only the utility computation changes; all subsequent steps GP update, EI maximisation, and the choice between continuing or starting runs remain identical. The modification is therefore a one-line replacement:

scalar = sigmoid_score $-\ \beta$ log1p (cumulative_cost).

Algorithm 1 BOIL-C within the BOIL optimisation loop

Input: prior hyper-parameters m_0, g_0, weight β, initial set of configurations $\mathcal{X}_0$
Initialise GP surrogate $\mathcal{S}$ and run set $\mathcal{R} \leftarrow \emptyset$
while budget not exhausted **do**
  for each active run $(x,t) \in \mathcal{R}$ **do**
    Observe metric $r(x,t)$ and step cost $c(x,t)$; update $C(x,t) \leftarrow C(x,t-1) + c(x,t)$
    Compute score $s \leftarrow s\big(r(x,t); m_0, g_0\big)$
    Compute utility $u \leftarrow s - \beta \cdot \log\big(1 + C(x,t)\big)$
    Update surrogate $\mathcal{S}$ with input features of (x,t) and target u
  end for
  Select next action (continue an existing run or start new x) by maximising EI on $\mathcal{S}$
  Execute the selected action for one step and add/update in $\mathcal{R}$
end while

4.3 PROPERTIES

- **Monotonicity**: Monotonicity in r is preserved for any $\beta \geq 0$; higher accuracy still yields higher utility at equal cost.
- **Backward compatibility**: Setting $\beta = 0$ recovers BOIL exactly.
- **Comparability**: Because cost is measured in seconds on homogeneous hardware the penalty is comparable across configurations without further normalisation.

5 EXPERIMENTAL SETUP

5.1 DATASETS AND MODELS

CIFAR-10: 45 k/5 k/10 k train/val/test split. Model: four 3×3 conv layers (64–128–256–256), ReLU, 512-unit fully connected layer, dropout, ≈ 1.2 M parameters. CartPole-v0: DQN with standard target network and ϵ-greedy exploration.

5.2 SEARCH SPACES

CIFAR-10: learning-rate $\in$ (log-uniform), batch-size $\in \{32, 64, 128\}$, dropout $\in$. CartPole: learning-rate $\in$, target-update $\in \{100, 200, 400, 800\}$ steps.

5.3 OPTIMISATION BUDGET

Each optimiser (BOIL-C, BOIL, Hyperband) receives 8 physical GPU-hours and is run with five independent random seeds. Single NVIDIA A100 GPUs are assigned per trial; seeds run in parallel to saturate the budget.

5.4 TRAINING DETAILS

CIFAR-10 uses SGD with momentum 0.9, weight decay 5×10^{-4}, 200 epochs, cosine LR schedule and batch-size 64 unless overridden by the search space. RL follows the standard DQN recipe; interaction steps rather than episodes define the curve index t.

5.5 LOGGING AND METRICS

The best-so-far validation metric is recorded each wall-clock second. Integrating that trace yields AUC-Time (higher = better). We also report final validation and test accuracy/return and confusion matrices. Hardware timing overhead is negligible relative to training time.

5.6 BASELINES

BOIL is implemented exactly as in the authors' repository. Hyperband uses the same search space and budget on CartPole, providing a strong cost-aware benchmark.

6 RESULTS

6.1 CIFAR-10

Averaged over five seeds: final test accuracy: BOIL-C 92.20%, BOIL 91.83% (+0.37 pp). Final test loss: 0.309 vs 0.332 (lower is better). AUC-Time: BOIL-C improves by 24.9%. Median time to 90% accuracy: 2.8 h vs 4.1 h (-31.7%). Gains are distributed across classes for instance, correct class-0 predictions rise from $928 \rightarrow 935$ confirming no performance trade-off.

6.2 CARTPOLE-V0

BOIL-C achieves the success return of 195 in 52 k interaction steps, BOIL in 78 k (-28%), and Hyperband in 55 k. AUC-Time mirrors these trends ($+28\%$ over BOIL, $+2\%$ over Hyperband). Notably, BOIL-C evaluates $\approx 30\%$ fewer configurations than Hyperband yet attains identical speed.

6.3 ABLATION ON β

$\beta \in \{0, 0.1, 0.25, 0.5\}$. AUC-Time gains are +0%, +17%, +25%, -3% respectively; $\beta = 0.5$ slightly hurts final accuracy (-0.6 pp). Thus $\beta \in [0.1, 0.3]$ is a robust operating range.

6.4 FAIRNESS

All runs used identical budgets, seeds, search spaces and training pipelines. Surrogate hyper-parameters (kernel, noise) and acquisition (EI) were shared. The sole difference between BOIL and BOIL-C is the cost-aware scalar.

6.5 LIMITATIONS

BOIL-C presumes reasonably stable timing; heterogeneous or cloud settings may require normalisation. Extremely large β can over-penalise compute-heavy but ultimately superior configurations.

7 CONCLUSION

BOIL-C demonstrates that incorporating compute cost directly into learning-curve compression is a powerful yet simple means to accelerate Bayesian hyper-parameter optimisation. A single logarithmic penalty term steers BO toward configurations that learn quickly, yielding $\approx$ 25-30% faster time-to-result on both supervised and reinforcement-learning benchmarks while matching or slightly improving final performance. Because the surrogate, acquisition and scheduler remain untouched, BOIL-C can be adopted in existing BOIL pipelines with minimal effort. Future directions include adaptive β schedules, integration with asynchronous multi-fidelity frameworks and evaluation on larger-scale tasks such as ImageNet or continuous-control RL environments.

References: BOIL Nguyen et al. (2019); neural-network partitioning for HPO Mlodozeniec et al. (2023).

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.