7th ICML Workshop on Automated Machine Learning (2020)

Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits

Jack Parker-Holder JACKPH@ROBOTS.OX.AC.UK
University of Oxford, United Kingdom

Vu Nguyen VU@ROBOTS.OX.AC.UK
University of Oxford, United Kingdom

Stephen Roberts SJROB@ROBOTS.OX.AC.UK
University of Oxford, United Kingdom

Abstract

Selecting optimal hyperparameters is a key challenge in machine learning. A recent approach to this problem (PBT) showed it is possible to achieve impressive performance by updating both weights and hyperparameters in a single training run of a population of agents. Despite it's success, PBT relies on heuristics to explore the hyperparameter space, thus lacks theoretical guarantees, requires vast computational resources and often suffers from mode collapse when this is not available. In this work we introduce Population-Based Bandits (PB2), the first provably efficient PBT-style algorithm. PB2 uses a probabilistic model to balance exploration and exploitation, thus it is able to discover high performing hyperparameter configurations with far fewer agents than typically required by PBT.

1. Introduction

Neural networks have achieved remarkable success in a variety of fields (Silver et al., 2016; Hochreiter and Schmidhuber, 1997; Krizhevsky et al., 2012), leading to widespread adoption in both industry and academia. However, as with many machine learning algorithms, it is often only after extensive trial-and-error that impressive, yet hard to reproduce (Bergstra et al., 2013), headline results can be achieved.

This has led to a surge in popularity for Automated Machine Learning (AutoML, Hutter et al. (2018)), which seeks to automate the training of machine learning models. A key component in AutoML is automatic hyperparameter selection (Bergstra et al., 2011; Melis et al., 2018). Popular solutions include Bayesian Optimization (BO, Brochu et al. (2010); Hennig and Schuler (2012); Snoek et al. (2012)) and Evolutionary Algorithms (EAs).

A recent approach, Population Based Training (PBT, Jaderberg et al. (2017); Li et al. (2019)), showed it is possible to achieve impressive performance by updating both weights and hyperparameters during a *single* training run of a population of agents. PBT also benefits from being highly scalable and is shown to be particularly effective in reinforcement learning (Schmitt et al., 2018; Liu et al., 2019; Espeholt et al., 2018).

However, a key drawback of PBT is the reliance on heuristics for selecting new hyperparameter configurations. This leads to two shortcomings: 1) it requires vast computational resources to outperform a random baseline, since small populations will often collapse to a suboptimal mode, 2) it lacks theoretical grounding, given that greedy exploration methods suffer from unbounded regret.

©2020 Provably Efficient Online Hyperparameter Optimization with Population-Based Bandits.

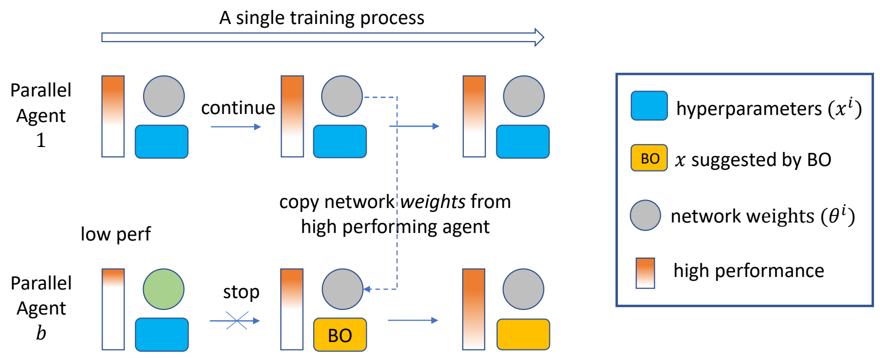

Figure 1: Population-Based Bandit Optimization: a population of agents is trained in parallel. Each agent has weights (grey) and hyperparameters (blue). The agents are evaluated periodically (orange bar), and if an agent is underperforming, it's weights are replaced by randomly copying one of the better performing agents, and its hyperparameters are selected using Bayesian Optimization.

Our key contribution is the first provably efficient PBT-style algorithm, Population-Based Bandit Optimization, or PB2 (Fig. 1). We draw the connection of maximizing the reward in PBT-style to the regret in bandit literature (Srinivas et al., 2010; Desautels et al., 2014). This allows us to formalize the online hyperparameter selection problem as batch Gaussian process bandit of a time-varying function. We derive a convergence analysis for the proposed PB2, the first such result for a PBT-style algorithm. Furthermore, we show this translates to impressive performance gain in tuning deep reinforcement learning agents.

2. Population-Based Bandits

2.1 Problem Statement

In this paper we consider the problem of selecting optimal hyperparameters, x_t^i, from a compact, convex subset $\mathcal{D} \in \mathbb{R}^d$. Here the index i refers to the ith agent in a population/batch, and we use the subscript t to represent the number of timesteps/epochs/iterations elapsed during the training of a neural network. The Population-Based Training (PBT, Jaderberg et al. (2017)) algorithm trains a population (or batch) of B agents in parallel. Each agent $i \in B$ has both hyperparameters $x_t^i \in \mathbb{R}^d$ and weights θ_t^i. At every t_{ready} step interval (i.e. if $t \mod t_{ready} = 0$), the agents are ranked and the worst performing agents are replaced with members of the best performing agents ($A \subset B$) as follows:

- **Weights** (θ_t^i): copied from one of the best performing agents, i.e. $\theta_t^j \sim \text{Unif}\{\theta_t^j\}_{j \in A}$.
- **Hyperparameters** (x_t^i): with probability ϵ it uses **random exploration**, and re-samples from the original distribution, otherwise, it uses **greedy exploration**, and perturbs one of the best performing agents, i.e $\{x^j * \lambda\}_{j \in A}, \lambda \sim [0.8, 1.2]$.

We consider a reward for a deep RL agent under a given set of hyperparameters at timestep t, $F_t(x_t)$. When training for a total of T steps, our goal is to maximize the final reward $F_T(x_T)$. We formulate this problem as optimizing the black-box time-varying reward function f_t, over $\mathcal{D}$. Every t_{ready} steps, we observe noisy observations, $y_t = f_t(x_t) + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ for some fixed σ^2. The function f_t represents the change in F_t after training for t_{ready} steps, i.e. $F_t - F_{t-t_{\text{ready}}}$. We define the best choice at each timestep as $x_t^* = \arg\max_{x_t \in \mathcal{D}} f_t(x_t)$, and so the **regret** of each decision as $r_t = f_t(x_t^*) - f_t(x_t)$.

Lemma 1. *Maximizing the reward F_T for a deep RL agent for a given hyperparameter schedule $\{x_t\}_{t=1}^T$ is equivalent to maximizing the time-varying black-box function $f_t(x_t)$ and minimizing the corresponding cumulative regret $r_t(x_t)$,*

$$\max F_T(x_T) = \max \sum_{t=1}^{T} f_t(x_t) = \min \sum_{t=1}^{T} r_t(x_t). \tag{1}$$

In subsequent sections, we present a time-varying bandit approach which is used to minimize the cumulative regret $R_T = \sum_{t=1}^T r_t$. Lemma 1 shows this is equivalent to maximizing the final performance/reward of a neural network model (see: Section 7 for the proof).

2.2 Gaussian Process Bandits for a Time-Varying Function

We model f_t using a time-varying Gaussian Process (GP, Rasmussen and Williams (2005)) which is specified by a mean function $\mu_t : \mathcal{X} \rightarrow \mathbb{R}$ and a kernel (covariance function) $k : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$. If $f_t \sim GP(\mu_t, k)$, then $f_t(x_t)$ is distributed normally $\mathcal{N}(\mu_t(x_t), k(x_t, x_t))$ for all $x_t \in \mathcal{D}$. After we have observed T data points $\{(x_t, f(x_t))\}_{t=1}^T$, the GP posterior belief at new point $x'_t \in \mathcal{D}$, $f_t(x'_t)$ follows a Gaussian distribution with mean $\mu_t(x')$ and variance $\sigma_t^2(x')$ as:

$$\mu_t(x') := \mathbf{k}_t(x')^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_t \tag{2}$$

$$\sigma_t^2(x') := k(x', x') - \mathbf{k}_t(x')^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_t(x'), \tag{3}$$

where $\mathbf{K}_t := \{k(x_i, x_j)\}_{i,j=1}^t$ and $\mathbf{k}_t := \{k(x_i, x'_t)\}_{i=1}^t$. Equipped with (2) and (3), we can select new samples by maximizing an *acquisition function.*

We cast the problem of optimizing our neural network parameters as time-varying bandit optimization. We follow Bogunovic et al. (2016) to formulate this problem by modelling the reward function under the time-varying setting as follows:

$$f_1(x) = g_1(x), \quad f_{t+1}(x) = \sqrt{1-\omega} f_t(x) + \sqrt{\omega} f g_{t+1}(x) \quad \forall t \geq 2, \tag{4}$$

where $g_1, g_2, ...$ are independent random functions with $g \sim GP(0, k)$ and $\omega \in [0, 1]$ models how the function varies with time, such that if $\omega = 0$ we return to GP-UCB and if $\omega = 1$ then each evaluation is independent. This model introduces a new hyperparameter (ω), however, we note it can be optimized by maximizing the marginal likelihood for a trivial additional cost compared to the expensive function evaluations (we include additional details on this in the Appendix). This leads to the extensions of Eqs. (2) and (3) using the new covariance matrix $\tilde{\mathbf{K}}_t = \mathbf{K}_t \circ \mathbf{K}_t^{\text{time}}$ where $\mathbf{K}_t^{\text{time}} = [(1-\omega)^{|i-j|/2}]_{i,j=1}^T$ and $\tilde{\mathbf{k}}_t(x) = \mathbf{k}_t \circ \mathbf{k}_t^{time}$ with $\mathbf{k}_t^{time} = [(1-\omega)^{(T+1-i)/2}]_{i=1}^T$. Here $\circ$ refers to the Hadamard product.

2.3 Moving to a Population

In the PBT setting we need to consider an entire population of agents. This changes the problem from a sequential to a *batch* blackbox optimization problem. This poses an additional challenge, since we must select x_t without full knowledge of all $\{(x_i, y_i)\}_{i=1}^{t-1}$. A key observation in Desautels et al. (2014) is that since a GP's variance (Eqn. 3) does not depend on y_t, the acquisition function can account for incomplete trials by updating the uncertainty at the selected points. Concretely, we define $x_{t,b}$ to be the b-th point selected in a batch, after t timesteps. This point may draw on information from $t + (b - 1)$ previously selected points. In the single agent, sequential case, we set $B = 1$ and recover $t, b = t - 1$. Thus, we can select the next sample by maximizing the following acquisition function:

$$x_{t,b} = \arg\max_{x \in \mathcal{D}} \mu_{t,1}(x) + \sqrt{\beta_t}\sigma_{t,b}(x) \tag{5}$$

for $\beta_t > 0$. In Equation 5 we have the mean from the previous batch $(\mu_{t,1}(x))$, but can update the uncertainty using our knowledge of the agents currently training $(\sigma_{t,b}(x))$.

2.4 Population-Based Bandit Optimization (PB2) Algorithm

We now introduce Population-Based Bandit Optimization (PB2). In a similar fashion to PBT, PB2 trains a population of agents in parallel, however, while we still copy *weights* from superior models, the key difference comes in the selection of x_t^i. We use the existing data (D_t) to model the function f_t with a GP. For the GP kernel we use $= k_{\text{SE}} \circ k^{\text{time}}$, and we use the acquisition function in Eqn. 5. This allows us to efficiently make use of data from previous trials when selecting new configurations.

```
Algorithm 1: Population-Based Bandit Optimization (PB2)
1 Initialize: Network weights $\{\theta_0^i\}_{i=1}^B$, hyperparameters $\{x_0^i\}_{i=1}^B$, dataset $D_0 = \emptyset$
2 (in parallel) for $t = 1, \dots, T-1$ do
3     1. Update Models: $\theta_t^i \leftarrow \text{step}(\theta_{t-1}^i | x_{t-1}^i)$ ;
4     2. Evaluate Models: $y_t^i = F_t(x_t^i) - F_{t-1}(x_{t-1}^i) + \epsilon_t$ for all $i$;
5     3. Record Data: $D_t = D_{t-1} \cup \{(y_t^i, t, x_t^i)\}_{i=1}^B$ ;
6     4. If $t \mod t_{\text{ready}} = 0$:
          • Copy weights: Rank agents, if $\theta^i$ is in the bottom $\lambda\%$ then copy weights $\theta^j$ from
            the top $\lambda\%$.
          • Select hyperparameters: Fit a GP model to $D_t$ and select hyper-parameters $x_t^i$ by
            maximizing equation (5).
7 Return best model.
```

Next we present our main theoretical result, showing that PB2 is able to achieve sublinear regret. This is the first such result for a PBT-style algorithm.

Theorem 2. *Let the domain $\mathcal{D} \subset [0, r]^d$ be compact and convex where d is the dimension and suppose that $f \sim GP(0, k)$ where the kernel k is almost surely continuously differentiable and satisfies Lipschitz assumptions for some a, b. Let fix $\delta \in (0, 1)$ and set $\beta_T = 2 \log \frac{\pi^2 T^2}{2\delta} + 2d \log rdbT^2 \sqrt{\log \frac{da\pi^2 T^2}{2\delta}}$. By defining $C_1 = 32/\log(1 + \sigma_f^2)$, the PB2 algorithm satisfies the following regret bound after T time steps with probability at least $1 - \delta$:*

$$R_{TB} = \sum_{t=1}^{T} f_t(\mathbf{x}_t^*) - f_t(\mathbf{x}_t) \leq \sqrt{C_1 T \beta_T \left(\frac{T}{\tilde{N}B} + 1\right) \left(\gamma_{\tilde{N}B} + \left[\tilde{N}B\right]^{\frac{5}{2}} \omega\right)} + 2$$

the bound holds for any block length $\tilde{N} \in \{1, ..., T\}$ and $B \ll T$.

3. Experiments

We focus our experiments on the RL setting, since it is notoriously sensitive to hyperparameters (Henderson et al., 2017). We focus on population sizes of $B \in \{4, 8\}$, which means the algorithm can be run locally on most modern computers. We benchmark PB2 vs. PBT, with identical configurations aside from the selection of x_t^i. We also compare against a random search (RS) baseline (Bergstra and Bengio, 2012). Random search is a challenging baseline because it does not make assumptions about the underlying problem, and typically achieves close to optimal performance asymptotically (Hutter et al., 2018). We also compare our results against a recent state-of-the-art distributed algorithm (ASHA, Li et al. (2018)). ASHA was shown to outperform PBT for supervised learning but remains untested for RL. All experiments were conducted using the tune library (Liaw et al., 2018; Liang et al., 2018)[1].

3.1 On Policy Reinforcement Learning

We consider optimizing a policy for continuous control problems from the OpenAI Gym (Brockman et al., 2016). In particular, we seek to optimize the hyperparameters for Proximal Policy Optimization (PPO, Schulman et al. (2017)), for the following tasks: BipedalWalker, LunarLanderContinuous, Hopper and InvertedDoublePendulum.

Table 1: Median best performing agent across 10 seeds. The best performing algorithms are bolded.

	B	RS	ASHA	PBT	PB2	vs. PBT
BipedalWalker	4	234	236	223	**276**	+24%
LunarLanderContinuous	4	161	213	159	**235**	+48%
Hopper	4	1638	1819	1492	**2346**	+57%
InvertedDoublePendulum	4	8094	7899	**8893**	8179	-8%
BipedalWalker	8	240	255	277	**291**	+5%
LunarLanderContinuous	8	175	231	247	**275**	+11%

For all experiments we use a neural network policy with two 32-unit hidden layers and tanh activations. During training, we optimize the following hyperparameters: batch size, learning rate, GAE parameter (λ) and PPO clip parameter (ϵ). We use the same fixed ranges across all four environments (included in the Appendix Section 5). All experiments are conducted for 10^6 environment timesteps, with the t_{ready} command triggered every 5×10^4 timesteps. For ASHA, we initialize a population of 18 agents to compare against $B = 4$ and 48 agents for $B = 8$. These were chosen to achieve the same total budget with the grace period equal to the t_{ready} criteria for PBT and PB2. We repeat each experiment with ten seeds, and show the median best reward achieved from each run in Table 1.

In almost all cases we see performance gains from PB2 vs. PBT. In fact, 3/4 of cases PBT actually underperforms random search with the smaller population size ($B = 4$), demonstrating its reliance on large computational resources. This is confirmed in the original PBT paper, where the smallest population size fails to outperform (see: Table 1, Jaderberg et al. (2017)). One possible explanation for this is the greediness of PBT leads to prematurely abandoning promising regions of the search space. Another is that the small changes in

1. An implementation of PB2 will be made public upon publication.

parameters (multiple of 0.8 or 1.2) is mis-specified for discovering the optimal regions, thus requiring more initial samples to sufficiently span the space. This may also be the case if there is a shift later in the optimization process. Interestingly, PBT does perform well for InvertedDoublePendulum, this may be explained by the relative simplicity of the problem. For the larger population size PB2 outperforms all other methods, while PBT comes second. This shows that PB2 is able to scale to larger computational budgets. Interestingly, the state-of-the-art supervised learning performance of ASHA fails to translate to RL, where it clearly performs worse than both PBT and PB2 for the larger setting, and performs worse than PB2 for the smaller one.

3.2 Off Policy Reinforcement Learning

We also evaluate PB2 in a larger setting, optimizing three hyperparameters for IMPALA (Espeholt et al., 2018) in the space invaders environment from the Arcade Learning Environment (Bellemare et al., 2012). In the original paper, the best results for IMPALA come with the use of PBT with a population size of $B = 24$. Here we optimize the same three hyperparameters as in the original paper, but with a much smaller population ($B = 4$), making it essential to efficiently explore the hyperparameter space.

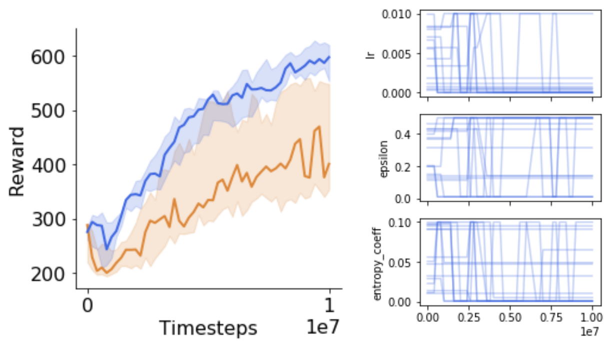

Figure 2: Left, median curves for seven seeds, with inter-quartile range shaded. Right, all agent configurations found by PB2.

We train for 10 million timesteps, equivalent to 40 million frames, and set t_{ready} to 5×10^5 timesteps. PB2 achieves a median best reward of 614, only slightly less than the hand-tuned performance reported in the rllib implementation implementation[2]. Meanwhile, PBT is only able to achieve a median best reward of 479.

As we see on the right hand side in Fig 2, PB2 effectively explores the entire range of parameters, which enables it to find optimal configurations even with a small number of trials. More details are in the Appendix, Section 5.

4. Conclusion and Future Work

We introduced Population-Based Bandits (PB2), the first PBT-style algorithm with sublinear regret guarantees. PB2 replaces the heuristics from the original PBT algorithm with theoretically guided GP-bandit optimization. This allows it to balance exploration and exploitation in a principled manner, preventing mode collapse to suboptimal regions of the hyperparameter space and making it possible to find high performing configurations with a small computational budget. Our algorithm complements the existing approaches for optimizing hyperparameters for deep learning frameworks.

2. See "RLlib IMPALA 32-workers", here: `https://github.com/ray-project/rl-experiments`

References

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *CoRR*, abs/1207.4708, 2012.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281â305, February 2012. ISSN 1532-4435.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*. 2011.

Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. Time-Varying Gaussian Process Bandit Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian Process bandit optimization. *Journal of Machine Learning Research*, 15 (119):4053–4103, 2014.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *CoRR*, abs/1709.06560, 2017.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *J. Mach. Learn. Res.*, 13(null):1809â1837, June 2012. ISSN 1532-4435.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Roger A. Horn and Charles R. Johnson. *Frontmatter*, pages i–vi. Cambridge University Press, 2 edition, 2012.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

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

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2012.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*. 2012.

Ang Li, Ola Spyra, Sagi Perel, Valentin Dalibard, Max Jaderberg, Chenjie Gu, David Budden, Tim Harley, and Pramod Gupta. A generalized framework for population based training. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, 2019.

Liam Li, Kevin G. Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. Massively parallel hyperparameter tuning. *CoRR*, 2018.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint*, 2018.

Siqi Liu, Guy Lever, Nicholas Heess, Josh Merel, Saran Tunyasuvunakool, and Thore Graepel. Emergent coordination through competition. In *International Conference on Learning Representations*, 2019.

Gabor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Simon Schmitt, Jonathan J. Hudson, Augustin Zídek, Simon Osindero, Carl Doersch, Wojciech M. Czarnecki, Joel Z. Leibo, Heinrich Küttler, Andrew Zisserman, Karen Simonyan, and S. M. Ali Eslami. Kickstarting deep reinforcement learning. *CoRR*, abs/1803.03835, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*. 2012.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, 2010.

Appendix

5. Experiment Details

For all experiments we set $\beta_t = c_1 + \log(c_2 t)$ with $c_1 = 0.2$ and $c_2 = 0.4$, as in the traffic speed data experiment from Bogunovic et al. (2016).

Table 2: IMPALA: Fixed

Parameter	Value
Num Workers	5
Num GPUs	0

Table 3: IMPALA: Learned

Parameter	Value
Epsilon	$\{0.01, 0.5\}$
Learning Rate	$\{10^{-3}, 10^{-5}\}$
Entropy Coeff	$\{0.001, 0.1\}$

Table 4: PPO: Fixed

Parameter	Value
Filter	$MeanStdFilter$
SGD Iterations	10
Architecture	32-32
ready	5×10^4

Table 5: PPO: Learned

Parameter	Value
Batch Size	$\{1000, 60000\}$
GAE λ	$\{0.9, 0.99\}$
PPO Clip ϵ	$\{0.1, 0.5\}$
Learning Rate η	$\{10^{-3}, 10^{-5}\}$

Table 6: CIFAR: Fixed

Parameter	Value
Optimizer	Adam
Iterations	50
Architecture	3 Conv Layers
ready	5

Table 7: CIFAR: Learned

Parameter	Value
Train Batch Size	$\{4, 128\}$
Dropout-1	$\{0.1, 0.5\}$
Dropout-2	$\{0.1, 0.5\}$
Learning Rate	$\{10^{-3}, 10^{-4}\}$
Weight Decay	$\{10^{-3}, 10^{-5}\}$
Momentum	$\{0.8, 0.99\}$

6. Additional Experiments

Supervised Learning We used PB2 to optimize six hyperparameters for a Convolutional Neural Network (CNN) (architecture from `https://zhenye-na.github.io/2018/09/28/pytorch-cnn-cifar10.html`) on the CIFAR-10 dataset (Krizhevsky, 2012). In each setting we randomly sample the initial hyperparameter configurations and train on half of the dataset for 50 epochs. We $B = 4$ agents for RS, PBT and PB2, with t_{ready} as 5 epochs. For ASHA we have the same maximum budget across all agents but begin with a population size of 16.

In Table 6 we show the median best performing agent from each training run. As we see, PB2 significantly outperforms both PBT and the Random baseline, while slightly outperforming ASHA. This result is non trivial since ASHA was designed for SL and focused on outperformance on this exact task in the original paper (see: Li et al. (2018)).

Table 8: Median best performing agent across 5 seeds. The best performing methods are bolded.

	RS	ASHA	PBT	PB2
Test Accuracy	84.43	88.85	87.20	**89.10**

Robustness to Hyperparameter Ranges One key weakness of PBT is its reliance on a large population size to explore the hyperparameter space. This can be magnified if the hyperparameter range is mis-specified or unknown (the bounds placed on the hyperparameters may require tuning). PB2 avoids this issue, since it is able to select a point anywhere in the range, so does not rely on random sampling or gradual movements to get to optimal regions. We evaluate this by re-running the BipedalWalker task with a batch size drawn from $\{5000, 200000\}$. This means many agents are initialized in a very inefficient way, since when an agent has a batch size of $200,000$ it is using 20% of total training samples for a single gradient step. In Fig. 3 we see the performance for PBT is significantly reduced, while PB2 is still able to learn good policies. While both methods perform worse than in Table 1, PB2 still achieves a median best of 203, vs. -12 for PBT.

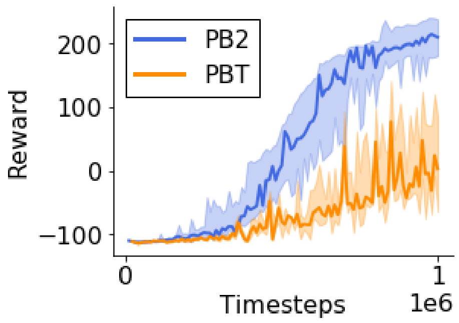

Figure 3: Median curves, IQR shaded.

Additional Off Policy Results Here we include an additional off policy experiment, with the same configuration as in Section 3.2. We test PB2 and PBT on the breakout environment, where we see in Fig. 4 that once again PB2 is more sample efficient than PBT.

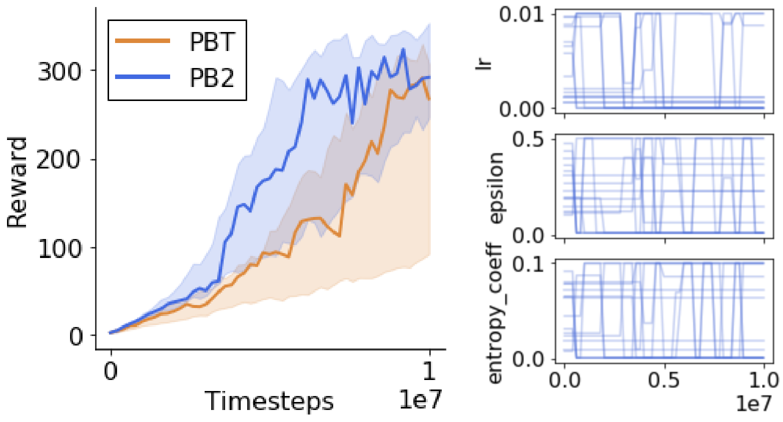

Figure 4: Left, median curves for seven seeds, with inter-quartile range shaded. Right, all agent configurations found by PB2.

7. Theoretical Results

We provide the proof for Lemma 1.

Proof. We have a reward at the starting iteration $F_1(x_1)$ as a constant that allows us to write the objective function as:

$$F_T(x_T) - F_1(x_1) = F_T(x_T) - F_{T-1}(x_{T-1}) + \cdots + F_3(x_3) - F_2(x_2) + F_2(x_2) - F_1(x_1) \quad (6)$$

Therefore, maximizing the left of Eqn (6) is equivalent to minimizing the cummulative regret as follows:

$$\max\left[F_T(x_T) - F_1(x_1)\right] = \max \sum_{t=1}^{T} F_t(x_t) - F_{t-1}(x_{t-1}) = \max \sum_{t=1}^{T} f_t(x_t) = \min \sum_{t=1}^{T} r_t(x_t)$$

where we define $f_t(x_t) = F_t(x_t) - F_{t-1}(x_{t-1})$, the regret $r_t = f_t(x_t^*) - f_t(x_t)$ and $f_t(x_t^*) := \max_{\forall x} f_t(x)$ is an unknown constant. □

7.1 Convergence Analysis

We minimize the cumulative regret R_T by sequentially suggesting an $\mathbf{x}_t$ to be used in each iteration t. We shall derive the upper bound in the cumulative regret and show that it asymptotically goes to zero as T increases, i.e., $\lim_{T\to\infty} \frac{R_T}{T} = 0$. We make the following smoothness assumption to derive the regret bound of the proposed algorithm.

Assumptions. We will assume that the kernel k is hold for some (a, b) and all $L \geq 0$. The joint kernel satisfies for $k = 1, ..., K$,

$$\forall L \geq 0, t \leq \mathcal{T}, p(\sup \left|\frac{\partial f_t(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^{(k)}}\right| \leq L) \leq a e^{-(L/b)^2}. \quad (7)$$

The assumption 1 is achieved by using a time-varying kernel $k_{time}(t, t') = (1-\omega)^{\frac{|t-t'|}{2}}$ (Bogunovic et al., 2016) while assumption 2 is from the smooth functions.

Lemma 3 (Srinivas et al. (2010)). *Let* $L_t = b\sqrt{\log 3da\frac{\pi_t}{\delta}})$ *where* $\sum_{t=1}^{T} \frac{1}{\pi_t} = 1$*, we have with probability* $1 - \frac{\delta}{3}$,

$$\left|f_t(\mathbf{x}) - f_t(\mathbf{x}')\right| \leq L_t ||\mathbf{x} - \mathbf{x}'||_1, \forall t, \mathbf{x}, \mathbf{x}' \in D \quad (8)$$

Lemma 4 (Srinivas et al. (2010)). *We define a discretization* $D_t \subset D \subseteq [0, r]^d$ *of size* $(\tau_t)^d$ *satisfying* $||\mathbf{x} - [\mathbf{x}]_t||_1 \leq \frac{d}{\tau_t}, \forall \mathbf{x} \in D$ *where* $[\mathbf{x}]_t$ *denotes the closest point in* D_t *to* $\mathbf{x}$*. By choosing* $\tau_t = \frac{t^2}{L_t d} = rdbt^2\sqrt{\log(3da\pi_t/\delta)}$*, we have*

$$|f_t(\mathbf{x}) - f_t([\mathbf{x}]_t)| \leq \frac{1}{t^2}.$$

Lemma 5 (Srinivas et al. (2010)). *Let* $\beta_t \geq 2\log\frac{3\pi_t}{\delta} + 2d\log\left(rdbt^2\sqrt{\log\frac{3da\pi_t}{2\delta}}\right)$ *where* $\sum_{t=1}^{T} \pi_t^{-1} = 1$*, then with probability at least* $1 - \frac{\delta}{3}$*, we have*

$$|f_t(\mathbf{x}_t) - \mu_t(\mathbf{x}_t)| \leq \sqrt{\beta_t}\sigma_t(\mathbf{x}_t), \forall t, \forall \mathbf{x} \in \mathcal{D}.$$

We use TB to denote the batch setting where we will run the algorithm over T iterations with a batch size B. The mutual information is defined as $\tilde{\mathbf{I}}(f_{TB}; y_{TB}) = \frac{1}{2}\log\det\left(\mathbf{I}_{TB} + \sigma_f^{-2}\tilde{K}_{TB}\right)$ and the maximum information gain is as $\tilde{\gamma}_T := \max \tilde{\mathbf{I}}(\mathbf{f}_{TB}; \mathbf{y}_{TB})$ where $\mathbf{f}_{TB} := f_{TB}(\mathbf{x}_{TB}) = \left(f_{t,b}(\mathbf{x}_{t,b}), ..., f_{T,B}(\mathbf{x}_{T,B})\right), \forall b = 1....B, \forall t = t...T$ for the time variant GP f.

We extend the result presented in (Bogunovic et al., 2016) into the parallel setting using a population size of B. We will show that $\tilde{\gamma}_{TB} \leq \left(\frac{T}{\tilde{N}B} + 1\right)\left(\gamma_{\tilde{N}B} + \sigma_f^{-2}\left[\tilde{N}B\right]^{5/2}\omega\right)$. To derive the bound over the maximum information gain, we shall split $T \times B$ observations in time steps $\{1, ..., T\}$ into $T/\tilde{N}$ blocks of length $\tilde{N} \times B$, such that within each block the function f_i does not vary significantly. We assume for the time being that $T/\tilde{N}$ is an integer, and then handle the general case.

Lemma 6. *(Mirsky's theorem [Horn and Johnson (2012), Cor. 7.4.9.3]). For any matrices U and V of size $\tilde{N} \times \tilde{N}$, and any unitarily invariant norm $|||\cdot|||$, we have*

$$|||diag(\lambda_1(U), ..., \lambda_{\tilde{N}}(U)) - diag(\lambda_1(V), ..., \lambda_{\tilde{N}}(V))||| \leq |||U - V|||$$

where $\lambda_i(\cdot)$ is the i-th largest eigenvalue.

Lemma 7. *(extended from Bogunovic et al. (2016)) Let ω be the forgetting-remembering trade-off parameter and consider the kernel for time $1 - K_{time}(t, t') \leq \omega\,|t - t'|$, we bound the maximum information gain that*

$$\tilde{\gamma}_{TB} \leq \left(\frac{T}{\tilde{N} \times B} + 1\right)\left(\gamma_{\tilde{N}\times B} + \sigma_f^{-2}\left[\tilde{N} \times B\right]^{5/2}\omega\right).$$

Proof. Following the strategy in Bogunovic et al. (2016), we bound the mutual information with time variations by connecting to the time-invariant case (Srinivas et al., 2010). We shall split $T \times B$ observations in time steps $\{1, ..., T\}$ into $T/\tilde{N}$ blocks of length $\tilde{N} \times B$, such that within each block the function f_i does not vary significantly.

We bound the time-varying maximum information gain for each i-th block as $\tilde{\gamma}^i := \max \tilde{\mathbf{I}}^i(\mathbf{f}^i; \mathbf{y}^i)$ where $\mathbf{y}^i = [y_1, ..., y_{\tilde{N}\times B}]$ contains the observations in the i-th blocks and analogous design for $\mathbf{f}^i$. Our dataset is different from Srinivas et al. (2010); Bogunovic et al. (2016) that we have $\tilde{N} \times B$ in each block i as each evaluation has been done using B simultaneous workers. For convenience in deriving the bound, we denote the covariance matrix as

$$\tilde{K}^i = K^i \circ K^i_{time} = K^i + A^i$$

where $\circ$ denotes the Hadamard product, $K^i_{time} = \left[(1 - \omega)^{\frac{|t-t'|}{2}}\right]$, $A^i = K^i \circ \left(K^i_{time} - \mathbf{1}_{\tilde{N}}\right)$ and $\mathbf{1}_{\tilde{N}}$ is the matrix of one of size $\tilde{N}$. Each covariance matrix $\tilde{K}^i, K^i$ and K^i_{time} will have the same dimension of $\left[\tilde{N} \times B\right] \times \left[\tilde{N} \times B\right]$. We have each entry in K^i is bounded $[0, 1]$ and the absolute value of each entry in $K^i_{time} - \mathbf{1}_{\tilde{N}}$ is bounded by $1 - (1 - \omega)^{\frac{|i-j|}{2}} \leq \omega\,|i - j|$.

Therefore, we can bound the Frobenius norm of

$$||A^i||_F \leq \sum_{i,j} (i-j)^2 \omega^2 = \frac{1}{6} \left[\tilde{N} \times B\right]^2 \left(\left[\tilde{N} \times B\right]^2 - 1\right) \omega^2 \leq \left[\tilde{N} \times B\right]^4 \omega^2.$$

Let $U = K^i + A^i$ and $V = K^i$, we define $\Delta_k^i = \lambda_k \left(\tilde{K^i}\right) - \lambda_k \left(K^i\right)$, we have

$$\begin{aligned}\sum_{j=1}^{\tilde{N}\times B} \left(\Delta_j^i\right)^2 &= \sum_{j=1}^{\tilde{N}\times B} \left(\lambda_j \left(\tilde{K^i}\right) - \lambda_j \left(K^i\right)\right)^2 \\ &= ||diag\left(\lambda_1\left(\tilde{K^i}\right), ..., \lambda_{\tilde{N}\times B}\left(\tilde{K^i}\right)\right) - diag\left(\lambda_1\left(K^i\right), ..., \lambda_{\tilde{N}\times B}\left(K^i\right)\right)||_F^2 \\ &\leq ||A_{\tilde{N}\times B}||_F^2 \leq \left[\tilde{N} \times B\right]^4 \omega^2\end{aligned}$$

where we have utilized Lem. 6. We now bound the maximum information gain defined in each block $\tilde{\gamma}^i$ and connect it to the standard (time invariant) maximum information gain γ^i shown in Srinivas et al. (2010):

$$\begin{aligned}\tilde{\gamma}^i =& \frac{1}{2} \log\det\left(\mathbf{I}_{\tilde{N}B\times\tilde{N}B} + \sigma^{-2}\left(\tilde{K^i}\right)\right) = \sum_{i=1}^{\tilde{N}\times B} \log\left(1 + \sigma^{-2}\lambda_i\left(K^i + A^i\right)\right) \\ =& \sum_{i=1}^{\tilde{N}\times B} \log\left(1 + \sigma^{-2}\lambda_i\left(K^i\right) + \sigma^{-2}\Delta_k^i\right) \leq \sum_{i=1}^{\tilde{N}\times B} \log\left(1 + \sigma^{-2}\lambda_i\left(K^i\right)\right) + \sum_{i=1}^{\tilde{N}\times B} \log\left(1 + \sigma^{-2}\Delta_k^i\right) \quad (9) \\ =& \gamma_{\tilde{N}} + \sum_{i=1}^{\tilde{N}\times B} \log\left(1 + \sigma^{-2}\lambda_i\Delta_k^i\right) \leq \gamma_{\tilde{N}} + \tilde{N} \times B \times \log\left(1 + \sigma^{-2}\frac{1}{\tilde{N}\times B}\sum_{i=1}^{\tilde{N}\times B}\Delta_k^i\right) \quad (10) \\ \leq& \gamma_{\tilde{N}} + \tilde{N} \times B \times \log\left(1 + \sigma^{-2}\left[\tilde{N}\times B\right]^{3/2}\omega\right) \leq \gamma_{\tilde{N}} + \sigma^{-2}\left[\tilde{N}\times B\right]^{5/2}\omega \quad (11)\end{aligned}$$

where Eqn (9) is by $\log(1 + a + b) \leq \log(1 + a) + \log(1 + b)$, for $a, b > 0$. Eqn (10) is using Jensen inequality for $\log(1 + x)$. In Eqn (11), we utilize the inequality that $\sum_{i=1}^{\tilde{N}\times B} \Delta_k^i \leq \sqrt{\tilde{N}\times B}\sqrt{\sum_{i=1}^{\tilde{N}\times B} \Delta_k^i} \leq \sqrt{\tilde{N}\times B}\left[\tilde{N}\times B\right]^2 \omega$ and $\log(1 + x) \leq x$ for $\forall x > -1$.

Using the chain rule for mutual information as discussed in Bogunovic et al. (2016), we have $\tilde{\mathbf{I}}_T\left(f_T, y_T\right) \leq \sum_{i=1}^{T/\tilde{N}} \tilde{\mathbf{I}}^i\left(\mathbf{f}^i; \mathbf{y}^i\right)$. Maximizing both sides $\tilde{\gamma}_T := \max \tilde{\mathbf{I}}(\mathbf{f}_T; \mathbf{y}_T)$, we have

$$\tilde{\gamma}_{TB} \leq \sum_{i=1}^{T/\tilde{N}} \tilde{\gamma}^i \leq \left(\frac{T}{\tilde{N}\times B} + 1\right)\left(\gamma_{\tilde{N}} + \sigma_f^{-2}\left[\tilde{N}\times B\right]^{5/2}\omega\right)$$

where the bounds are similar across blocks $\tilde{\gamma}^i \leq \gamma_{\tilde{N}\times B} + \frac{1}{\sigma_f^2}\left[\tilde{N}\times B\right]^{5/2}\omega, \forall i \leq \frac{T}{\tilde{N}}$ and the addition of one is for the case when $\frac{T}{\tilde{N}}$ is not an integer. □

Uncertainty Sampling (US). We next derive an upper bound over the maximum information gain obtained from a batch $\mathbf{x}_{t,b}, \forall b = 1, ..., B$. In other words, we want to show that the information gain by our chosen points $\mathbf{x}_{t,b}$ will not go beyond the ones by maximizing the uncertainty. For this, we define an uncertainty sampling (US) scheme which fills in a batch $\mathbf{x}_{t,b}^{\text{US}}$ by maximizing the GP predictive variance. Particularly, at iteration t, we select $\mathbf{x}_{t,b}^{\text{US}} = \arg\max_{\mathbf{x}} \sigma_t(\mathbf{x} \mid D_{t,b-1}), \forall b \leq B$ and the data set is augmented over time to include the information of the new point, $D_{t,b} = D_{t,b-1} \cup \mathbf{x}_{t,b}^{\text{US}}$. We note that we use $\mathbf{x}_{t,b}^{\text{US}}$ to derive the upper bound, but this is not used by our PB2 algorithm.

Lemma 8. *Let $\mathbf{x}_{t,b}^{PB2}$ be the point chosen by our algorithm and $\mathbf{x}_{t,b}^{US}$ be the point chosen by uncertainty sampling (US) by maximizing the GP predictive variance $\mathbf{x}_{t,b}^{US} = \arg\max_{\mathbf{x} \in D} \sigma_t(\mathbf{x} \mid D_{t,b-1}), \forall b = 1, ...B$ and $D_{t,b} = D_{t,b-1} \cup \mathbf{x}_{t,b}$, We have*

$$\sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}^{PB2}\right) \leq \sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}^{US}\right) \leq \sigma_{t,b}\left(\mathbf{x}_{t,b}^{US}\right), \forall t \in \{1, ..., T\}, \forall b \in \{1, ...B\}.$$

Proof. The first inequality is straightforward that the point chosen by uncertainty sampling will have the highest uncertainty $\sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}^{\text{PB2}}\right) \leq \sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}^{\text{US}}\right) = \arg\max_{\mathbf{x}} \sigma_t(\mathbf{x} \mid D_{t,b-1})$.

The second inequality is obtained by using the principle of "information never hurts" (Krause et al., 2008), we know that the GP uncertainty for all locations $\forall \mathbf{x}$ decreases with observing a new point. Therefore, the uncertainty at the future iteration σ_{t+1} will be smaller than that of the current iteration σ_t, i.e., $\sigma_{t+1,b}\left(\mathbf{x}_{t+1,b}^{\text{US}}\right) \leq \sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right), \forall b \leq B, \forall t \leq T$. We thus conclude the proof $\sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}^{\text{US}}\right) \leq \sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right), \forall t \in \{1, ..., T\}, \forall b \in \{1, ...B\}$. □

Lemma 9. *The sum of variances of the points selected by the our PB2 algorithm $\sigma(.)$ is bounded by the sum of variances by uncertainty sampling $\sigma^{US}(.)$. Formally, w.h.p.,*

$$\sum_{t=2}^{T} \sigma_{t,1}\left(\mathbf{x}_{t,1}\right) \leq \frac{1}{B} \sum_{t=1}^{T} \sum_{b=1}^{B} \sigma_{t,b}\left(\mathbf{x}_{t,b}^{US}\right).$$

Proof. By the definition of uncertainty sampling in Lem. 8, we have $\sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}\right) \leq \sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right), \forall t \in \{1, ..., T\}, \forall b \in \{2, ...B\}$ and $\sigma_{t,1}\left(\mathbf{x}_{t,1}\right) \leq \sigma_{t,1}\left(\mathbf{x}_{t,1}^{\text{US}}\right)$ where $\mathbf{x}_{t,1}$ is the point chosen by our PB2 and $\mathbf{x}_{t,1}^{\text{US}}$ is from uncertainty sampling. Summing all over B, we obtain

$$\sigma_{t,1}\left(\mathbf{x}_{t,1}\right) + (B-1)\,\sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}\right) \leq \sigma_{t,1}\left(\mathbf{x}_{t,1}^{\text{US}}\right) + \sum_{b=2}^{B} \sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right)$$

$$\sum_{t=1}^{T} \sigma_{t,1}\left(\mathbf{x}_{t,1}\right) + (B-1) \sum_{t=1}^{T} \sigma_{t+1,1}\left(\mathbf{x}_{t+1,1}\right) \leq \sum_{t=1}^{T} \sum_{b=1}^{B} \sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right) \qquad \text{by summing over } T$$

$$\sum_{t=2}^{T} \sigma_{t,1}\left(\mathbf{x}_{t,1}\right) \leq \frac{1}{B} \sum_{t=1}^{T} \sum_{b=1}^{B} \sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right).$$

The last equation is obtained because of $\sigma_{1,1}\left(\mathbf{x}_{1,1}\right) \geq 0$ and $(B-1)\,\sigma_{T+1,1}\left(\mathbf{x}_{T+1,1}\right) \geq 0$. □

Lemma 10. *Let $C_1 = \frac{32}{\log\left(1+\sigma_f^{-2}\right)}$, σ_f^2 be the measurement noise variance and $\tilde{\gamma}_{TB} := \max \tilde{\mathbf{I}}$ be the maximum information gain of time-varying kernel, we have*

$$\sum_{t=1}^{T}\sum_{b=1}^{B} \sigma_{t,b}^2(\mathbf{x}_{t,b}^{US}) \leq \frac{C_1}{16}\tilde{\gamma}_{TB}$$

where $\mathbf{x}_{t,b}^{US}$ is the point selected by uncertainty sampling (US).

Proof. We show that $\sigma_{t,b}^2(\mathbf{x}_{t,b}^{\text{US}}) = \sigma_f^2\left(\sigma_f^{-2}\sigma_{t,b}^2(\mathbf{x}_{t,b}^{\text{US}})\right) \leq \sigma_f^2 C_2 \log\left(1+\sigma_f^{-2}\sigma_{t,b}^2\left(\mathbf{x}_{t,b}^{\text{US}}\right)\right), \forall b \leq B, \forall t \leq T$ where $C_2 = \frac{\sigma_f^{-2}}{\log\left(1+\sigma_f^{-2}\right)} \geq 1$ and σ_f^2 is the measurement noise variance. We have the above inequality because $s^2 \leq C_2 \log\left(1+s^2\right)$ for $s \in \left[0, \sigma_f^{-2}\right]$ and $\sigma_f^{-2}\sigma_{t,b}^2\left(\mathbf{x}_{t,b}^{\text{US}}\right) \leq \sigma^{-2}k\left(\mathbf{x}_{t,b}^{\text{US}}, \mathbf{x}_{t,b}^{\text{US}}\right) \leq \sigma_f^{-2}$. We then use Lemma 5.3 of Desautels et al. (2014) to have the information gain over the points chosen by a time-varying kernel

$$\tilde{\mathbf{I}} = \frac{1}{2}\sum_{t=1}^{T}\sum_{b=1}^{B}\log\left(1+\sigma_f^{-2}\sigma_{t,b}^2\left(\mathbf{x}_{t,b}^{\text{US}}\right)\right).$$

Finally, we obtain

$$\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}^2(\mathbf{x}_{t,b}^{\text{US}}) \leq \sigma_f^2 C_2 \sum_{t=1}^{T}\sum_{b=1}^{B}\log\left(1+\sigma_f^{-2}\sigma_{t,b}^2\left(\mathbf{x}_{t,b}^{\text{US}}\right)\right) = 2\sigma_f^2 C_2 \tilde{\mathbf{I}} = \frac{C_1}{16}\tilde{\gamma}_{TB}$$

where $C_1 = \frac{2}{\log\left(1+\sigma_f^{-2}\right)}$ and $\tilde{\gamma}_{TB} := \max \tilde{\mathbf{I}}$ is the definition of maximum information gain given by $T \times B$ data points from a GP for a specific time-varying kernel. □

Theorem 11. *Let the domain $\mathcal{D} \subset [0,r]^d$ be compact and convex where d is the dimension and suppose that $f \sim GP(0,k)$ where the kernel k is almost surely continuously differentiable and satisfies Lipschitz assumptions for some a, b. Let fix $\delta \in (0,1)$ and set $\beta_T = 2\log\frac{\pi^2 T^2}{2\delta} + 2d\log rdbT^2\sqrt{\log\frac{da\pi^2T^2}{2\delta}}$. By defining $C_1 = 32/\log(1+\sigma_f^2)$, the PB2 algorithm satisfies the following regret bound after T time steps with probability at least $1-\delta$:*

$$R_{TB} = \sum_{t=1}^{T} f_t(\mathbf{x}_t^*) - f_t(\mathbf{x}_t) \leq \sqrt{C_1 T \beta_T \left(\frac{T}{\tilde{N}B}+1\right)\left(\gamma_{\tilde{N}B} + \left[\tilde{N}B\right]^{\frac{5}{2}}\omega\right)} + 2$$

the bound holds for any block length $\tilde{N} \in \{1, ..., T\}$ and $B \ll T$.

Proof. Let $\mathbf{x}_t^* = \arg\max_{\forall \mathbf{x}} f_t(\mathbf{x})$ and $\mathbf{x}_{t,b}$ be the point chosen by our algorithm at iteration t and batch element b, we define the (time-varying) instantaneous regret as $r_{t,b} = f_t(\mathbf{x}_t^*) - f_t(\mathbf{x}_{t,b})$ and the (time-varying) batch instantaneous regret over B points is as follows

$$\begin{aligned} r_t^B &= \min_{b \leq B} r_{t,b} = \min_{b \leq B} f_t(\mathbf{x}_t^*) - f_t(\mathbf{x}_{t,b}), \forall b \leq B \\ &\leq f_t(\mathbf{x}_t^*) - f_t(\mathbf{x}_{t,1}) \leq \mu_t(\mathbf{x}_t^*) + \sqrt{\kappa_t}\sigma_t(\mathbf{x}_t^*) + \frac{1}{t^2} - f_t(\mathbf{x}_{t,1}) && \text{by Lem. 4} \\ &\leq \mu_t(\mathbf{x}_{t,1}) + \sqrt{\kappa_t}\sigma_t(\mathbf{x}_{t,1}) + \frac{1}{t^2} - f_t(\mathbf{x}_{t,1}) \leq 2\sqrt{\kappa_t}\sigma_t(\mathbf{x}_{t,1}) + \frac{1}{t^2} && (12) \end{aligned}$$

where we have used the property that $\mu_t(\mathbf{x}_{t,1}) + \sqrt{\beta_t}\sigma_t(\mathbf{x}_{t,1}) \geq \mu_t(\mathbf{x}_t^*) + \sqrt{\beta_t}\sigma_t(\mathbf{x}_t^*)$ by the definition of selecting $\mathbf{x}_{t,1} = \arg\max_{\mathbf{x}} \mu_t(\mathbf{x}) + \sqrt{\beta_t}\sigma_t(\mathbf{x})$. Next, we bound the cumulative batch regret as

$$\begin{aligned}
R_{TB} = \sum_{t=1}^{T} r_t^B &\leq \sum_{t=1}^{T} \left(2\sqrt{\kappa_t}\sigma_t(\mathbf{x}_{t,1}) + \frac{1}{t^2}\right) && \text{by Eqn (12)} \\
&\leq 2\sqrt{\kappa_T}\sigma_1(\mathbf{x}_{1,1}) + \frac{2\sqrt{\kappa_T}}{B}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right) + \sum_{t=1}^{T}\frac{1}{t^2} && \text{by Lem. 8 and } \kappa_T \geq \kappa_t, \forall t \leq T \\
&\leq \frac{4\sqrt{\kappa_T}}{B}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right) + \sum_{t=1}^{T}\frac{1}{t^2} && (13) \\
&\leq \frac{4}{B}\sqrt{\kappa_T \times TB\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}^2(\mathbf{x}_{t,b}^{\text{US}})} + 2 \leq \sqrt{C_1\frac{T}{B}\kappa_T\tilde{\gamma}_{TB}} + 2 && (14) \\
&\leq \sqrt{C_1\frac{T}{B}\kappa_T\left(\frac{T}{\tilde{N}\times B} + 1\right)\left(\gamma_{\tilde{N}B} + \frac{1}{\sigma_f^2}\left[\tilde{N}\times B\right]^{5/2}\omega\right)} + 2. && (15)
\end{aligned}$$

where $C_1 = 32/\log(1+\sigma_f^2)$, $\mathbf{x}_{t,b}^{\text{US}}$ is the point chosen by uncertainty sampling – used to provide the upper bound in the uncertainty. In Eqn (13), we take the upper bound by considering two possible cases: either $\sigma_1(\mathbf{x}_{1,1}) \geq \frac{1}{B}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right)$ or $\frac{1}{B}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right) \geq \sigma_1(\mathbf{x}_{1,1})$. It results in $\frac{2}{B}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right) \geq \frac{1}{B}\sum_{t=1}^{T}\sum_{b=1}^{B}\sigma_{t,b}\left(\mathbf{x}_{t,b}^{\text{US}}\right) + \sigma_1(\mathbf{x}_{1,1})$. In Eqn (14) we have used $\sum_{t=1}^{\infty}\frac{1}{t^2} \leq \pi^2/6 \leq 2$ and $||z||_1 \leq \sqrt{T}||z||_2$ for any vector $z \in \mathcal{R}^T$. In Eqn (15), we utilize Lem. 7.

Finally, given the squared exponential (SE) kernel defined, $\gamma_{\tilde{N}B}^{SE} = \mathcal{O}(\left[\log \tilde{N}B\right]^{d+1})$, the bound is $R_{TB} \leq \sqrt{C_1\frac{T}{B}\beta_T\left(\frac{T}{\tilde{N}B}+1\right)\left((d+1)\log\left(\tilde{N}B\right) + \frac{1}{\sigma_f^2}\left[\tilde{N}\times B\right]^{5/2}\omega\right)} + 2$ where $\tilde{N} \leq T$ and $B \ll T$. We refer to Theorem 5 in Srinivas et al. (2010) for the bound of other common kernels including linear, Mattern and squared exponential. □

In our time-varying setting, if the time-varying function is highly correlated, i.e., the information between $f_1(.)$ and $f_T(.)$ does not change significantly, we have $\omega \rightarrow 0$ and $\tilde{N} \rightarrow T$. Then, the regret bound grows sublinearly with the number of iterations T, i.e., $\lim_{T\rightarrow\infty}\frac{R_{TB}}{TB} = 0$. This bound suggests that the gap between $f_t(\mathbf{x}_t)$ and the optimal $f_t(\mathbf{x}_t^*)$ vanishes asymptotically using PB2. In addition, our regret bound is tighter and better with increasing batch size B.

On the other hand in the worst case, if the time-varying function is not correlated, such as $\tilde{N} \rightarrow 1$ and $\omega \rightarrow 1$, then PB2 achieves the linear regret (Bogunovic et al., 2016).