7th ICML Workshop on Automated Machine Learning (2020)

Bayesian Optimization for Iterative Learning

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

Sebastian Schulze * SEBASTIAN.SCHULZE@ENG.OX.AC.UK
University of Oxford, UK

Michael A. Osborne MOSB@ROBOTS.OX.AC.UK
University of Oxford, UK

Abstract

Deep (reinforcement) learning systems are sensitive to hyperparameters which are notoriously expensive to tune, typically requiring running iterative processes over multiple epochs or episodes. Traditional tuning approaches only consider the final performance of a hyperparameter, ignoring intermediate information from the learning curve. In this paper, we present a Bayesian optimization approach which exploits the iterative structure of learning algorithms for efficient hyperparameter tuning. First, we transform each training curve into numeric scores representing training success as well as stability. Second, we selectively augment the data using the information from the curve. This augmentation step enables modeling efficiency. We demonstrate the efficiency of our algorithm by tuning hyperparameters for the training of deep reinforcement learning agents and convolutional neural networks. Our algorithm outperforms all existing baselines in identifying optimal hyperparameters in minimal time.

1. Introduction

Deep learning (DL) and deep reinforcement learning (DRL) have lead to impressive breakthroughs in a broad range of applications such as game play (Mnih et al., 2013; Silver et al., 2016), motor control (Todorov et al., 2012), and image recognition (Krizhevsky et al., 2012). To maintain general applicability, these algorithms expose sets of hyperparameters to adapt their behavior to any particular task at hand. This flexibility comes at the price of having to tune an additional set of parameters – poor settings lead to drastic performance losses or divergence (Smith, 2018; Henderson et al., 2018). The training process for DL and DRL is typically conducted in an iterative manner, such as based on stochastic gradient descent and carried out using multiple episodes. The knowledge accumulated during these training iterations can be useful to inform BO. However, most existing BO approaches (Shahriari et al., 2016) may have ignored the useful information contained in the preceding training steps.

In this paper, we present a Bayesian optimization approach for tuning algorithms where iterative learning is available – the cases of deep learning and deep reinforcement learning. First, we consider the joint space of input hyperparameters and number of training iterations to capture the learning progress at different time steps in the training process. We then propose to transform the whole training curve into a numeric score according to user preference. To learn across the joint space efficiently, we introduce a data augmentation technique leveraging intermediate information from the iterative process. By exploiting the iterative structure of training procedures, we encourage our algorithm to consider running a larger number of cheap (but high-utility) experiments, when cost-ignorant algorithms would only be able to run a few expensive ones. We demonstrate the efficiency

of our algorithm on training DRL agents on several well-known benchmarks as well as the training of convolutional neural networks. In particular, our algorithm outperforms existing baselines in finding the best hyperparameter in terms of wall-clock time. Our main contributions are: (1) an algorithm to optimize the learning curve of a ML algorithm by using training curve compression; (2) an approach to learn the compression curve from the data and data augmentation technique for increased sample-efficiency; (3) demonstration on tuning DRL and convolutional neural networks.

2. Bayesian Optimization for Iterative Learning (BOIL)

Problem setting. We consider training a machine learning algorithm given a d-dimensional hyperparameter $\mathbf{x} \in \mathscr{X} \subset \mathscr{R}^d$ for t iterations. This process has a training time cost $c(\mathbf{x},t)$ and produces training evaluations $r(. \mid \mathbf{x},t)$ for t iterations, $t \in [T_{\min}, T_{\max}]$. These could be episode rewards in DRL or training accuracies in DL. An important property of iterative training is that we know the whole curve at preceding steps $r_{t'}(\mathbf{x},t)$, $\forall t' \leq t$.

Given the raw training curve $r(. \mid \mathbf{x},t)$, we assume an underlying smoothed black-box function f, defined in Sec. 2.3. Formally, we aim to find $\mathbf{x}^* = \arg\max_{\mathbf{x} \in X} f(\mathbf{x}, T_{\max})$; at the same time, we want to keep the overall training time, $\sum_{i=1}^{N} c(\mathbf{x}_i, t_i)$, of evaluated settings $[\mathbf{x}_i, t_i]$ as low as possible. We summarize our variables in Table 1 in the supplement for ease of reading.

2.1 Selecting a next point using iteration-efficient modeling

We follow the popular designs in Krause and Ong (2011); Swersky et al. (2013); Song et al. (2019) to model the black-box function $f(\mathbf{x},t)$ and the cost $c(\mathbf{x},t)$ using two independent GPs to capture the correlation across hyperparameter $\mathbf{x}$ and iteration t as $f(\mathbf{x},t) \sim GP(0, K([\mathbf{x},t],[\mathbf{x}',t']))$ and $c(\mathbf{x},t) \sim GP(0, K_c([\mathbf{x},t],[\mathbf{x}',t']))$ where $[\mathbf{x},t] \in \mathscr{R}^{d+1}$; K and K_c are the respective covariance functions. In both models, we choose the covariance kernel as a product $k([\mathbf{x},t],[\mathbf{x}',t']) = k(\mathbf{x},\mathbf{x}') \times k(t,t')$ to induce similarities over parameter and iteration space. We estimate the predictive mean and uncertainty of both GPs at any input $\mathbf{z}_* = [\mathbf{x}_*, t_*]$ as

$$\mu(\mathbf{z}_*) = \mathbf{k}_* \left[\mathbf{K} + \sigma_y^2 \mathbf{I}\right]^{-1} \mathbf{y} \quad (1) \qquad \mu_c(\mathbf{z}_*) = \mathbf{k}_{c*} \left[\mathbf{K}_c + \sigma_c^2 \mathbf{I}\right]^{-1} \mathbf{c} \quad (3)$$

$$\sigma^2(\mathbf{z}_*) = k_{**} - \mathbf{k}_* \left[\mathbf{K} + \sigma_y^2 \mathbf{I}\right]^{-1} \mathbf{k}_*^T \quad (2) \qquad \sigma_c^2(\mathbf{z}_*) = k_{**} - \mathbf{k}_{c*} \left[\mathbf{K}_c + \sigma_c^2 \mathbf{I}\right]^{-1} \mathbf{k}_{c*}^T \quad (4)$$

where σ_y^2 is the noise variance for f and σ_c^2 is the noise variance for cost.

Our intuition is to select a point with high function value (exploitation), high uncertainty (exploration) and low cost (cheap). At each iteration n, we query the input parameter $\mathbf{x}_n$ and the number of iteration t_n using the simple form (Snoek et al., 2012; Wu et al., 2019):

$$\mathbf{z}_n = [\mathbf{x}_n, t_n] = \arg \max_{\mathbf{x} \in \mathscr{X}, t \in [T_{\min}, T_{\max}]} \alpha(\mathbf{x},t) / \mu_c(\mathbf{x},t). \quad (5)$$

where $\alpha()$ is the acquisition function and μ_c is the estimated cost defined in Eq. (3).

2.2 Augmentation with intermediate observations from a curve

When evaluating a parameter $\mathbf{x}$ over t iterations, we obtain not only a final score but also all reward sequences $r(t' \mid \mathbf{x},t), \forall t' = 1,...,t$. The auxiliary information from the curve can bring useful information for BO. Therefore, we propose to augment the information from the curve into the sample set of our GP model.

Ill-conditioned issue with a full curve. A naïve approach for augmentation is to add a full curve of points $\{[\mathbf{x}, j], y_j\}_{j=1}^{t}$ where y_j is computed using Eq. (7). However, this approach imposes serious issues in the conditioning of the GP covariance matrix. As we cluster more evaluations closely, the conditioning of the GP covariance degrades further, as discussed in McLeod et al. (2018). This conditioning issue is especially serious in our noisy DRL settings. We highlight this effect in Fig. 1 where the *natural log* of condition number goes above 25 if we augment the whole curve.

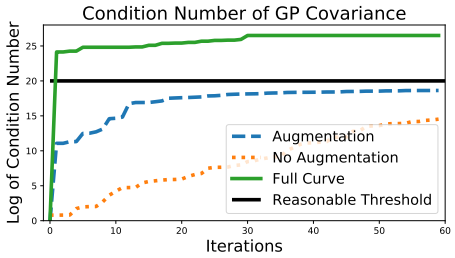

Figure 1: The condition number of GP covariance matrix goes badly if we add the whole curve of points into a GP. The large condition number measures the nearness to singularity.

Selecting subset of points from the curve. It is essential to selectively augment the observations from the learning curve. We can mitigate such conditioning issue by discouraging the addition of similar points close to each other. For this purpose, we can utilize several approaches, such as active learning (Osborne et al., 2012; Gal et al., 2017). We propose to use the simple, but effective, uncertainty sampling approach. We select a sample at the maximum of the GP predictive uncertainty. Formally, we sequentially select a set $Z = [z_1, ...z_M]$, $z_m = [\mathbf{x}, t_m]$, by varying t_m while keeping $\mathbf{x}$ fixed as

$$z_m = \arg\max_{\forall t' \leq t} \sigma([\mathbf{x}, t'] \mid D'), \forall m \leq M \text{ s.t. } \ln \text{of cond}(K) \leq \delta \tag{6}$$

where $D' = D \cup \{z_j = [\mathbf{x}, t_j]\}_{j=1}^{m-1}$. This sub-optimisation problem is done in a one-dimensional space of $t' \in \{T_{\min}, ..., t\}$, thus it is cheap to estimate using a gradient descent (the derivative of GP predictive variance is available (Rasmussen, 2006)).

These generated points Z will be used to calculate the output $r(z_m)$ and augmented into the observation set (X, Y) for fitting the GP. The number of samples M will be adaptively chosen such that the natural log of the condition number of the covariance matrix is less than a threshold. We illustrate the augmented observations and estimated scores in Fig. 2.

2.3 Training curve compression and estimating the transformation function

Existing BO approaches (Brochu et al., 2010; Chen et al., 2018; Li et al., 2018; Nguyen and Osborne, 2020) typically define the objective function as an average loss over the final learning episodes. However, this does not take into consideration how stable performance is or the training stage at which it has been achieved. We argue that averaging learning losses is likely misleading due to the noise and fluctuations of our observations (learning curves) – particularly during the early stages of training. We propose to compress the whole learning curve into a numeric score via a preference function representing the user's desired training curve. In the following, we use the Sigmoid function parameterized by a growth g_0 defining a slope and the middle point of the curve m_0 to compute the utility score as

$$y = \hat{y}(r, m_0, g_0) = r(. \mid \mathbf{x}, t) \bullet l(. \mid m_0, g_0) = \sum_{u=1}^{t} \frac{r(u \mid \mathbf{x}, t)}{1 + \exp(-g_0 [u - m_0])} \tag{7}$$

where $\bullet$ is a dot product. The Sigmoid preference has a number of desirable properties. As the early weights are small, less credit is given to fluctuations at the initial stages. However, as weights

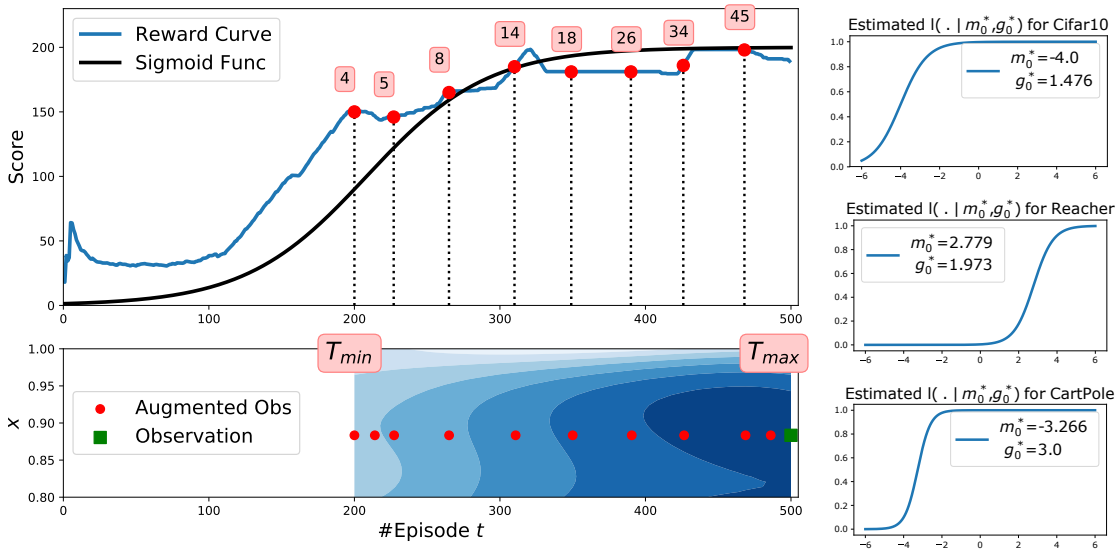

Figure 2: Left: the score in pink box is a convolution of the reward curve $r(. \mid \mathbf{x} = 0.9, t = 500)$ and a Sigmoid function up to this time step. Bottom: observations are selected to augment the data set (red dots). The heatmap indicates the GP predictive mean μ for f across number of episode t used to train an agent. T_{min} and T_{max} are two user-defined thresholds of number of used episodes for training. x is a hyperparameter to be tuned. Right: we learn the optimal parameter g_0^* and m_0^* for each experiment respectively.

monotonically increase, hyperparameters for improving performance are preferred. As weights saturate over time, stable, high performing configurations is preferred over short "performance spikes" characteristic of unstable training. Lastly, this utility score assigns higher values to the same performance if it is being maintained over more episodes.

Learning the transformation function from data. Different compression curves $l()$, parameterized by different choices of g_0 and m_0 in Eq. (7), may lead to different utilities y and thus affect the performance. Therefore, we propose to learn the suitable parameter g_0^* and m_0^* directly from the data. Our intuition is that the 'optimal' compression curve $l(m_0^*, g_0^*)$ will lead to better fitting for the GP. This better GP surrogate model, thus, will result in better prediction as well as optimization performance. We parameterize the GP log marginal likelihood L (Rasmussen, 2006) as the function of m_0 and g_0:

$$L(m_0, g_0) = \frac{1}{2}\hat{\mathbf{y}}^T \left(K + \sigma_y^2 \mathbf{I}\right)^{-1} \hat{\mathbf{y}} - \frac{1}{2}\ln\left|K + \sigma_y^2 \mathbf{I}\right| + \text{const} \tag{8}$$

where $\hat{\mathbf{y}}$ is the function of m_0 and g_0 defined in Eq. (7). We optimize m_0 and g_0 (jointly with other GP hyperparameters) using gradient-based approach. We derive the derivative $\frac{\partial L}{\partial m_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial m_0}$ and $\frac{\partial L}{\partial g_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial g_0}$ which can be computed analytically as:

$$\frac{\partial L}{\partial \hat{y}} = \left(K + \sigma_y^2 \mathbf{I}_N\right)^{-1} \hat{y}; \quad \frac{\partial \hat{y}}{\partial m_0} = \frac{-g_0 \times \exp(-g_0\left[u - m_0\right])}{\left[1 + \exp(-g_0\left[u - m_0\right])\right]^2}; \; \frac{\partial \hat{y}}{\partial g_0} = \frac{-m_0 \times \exp(-g_0\left[u - m_0\right])}{\left[1 + \exp(-g_0\left[u - m_0\right])\right]^2}.$$

Algorithm 1 Bayesian Optimization with Iterative Learning (BOIL)

Input: #iter N, initial data D_0, $\mathbf{z} = [\mathbf{x}, t]$. **Output**: optimal $\mathbf{x}^*$ and $y^* = \max_{\forall y \in D_N} y$

1: **for** $n = 1....N$ **do**
2: Fit two GPs to calculate $\mu_f(), \sigma_f()$ and $\mu_c()$ from Eqs. (1,2,3).
3: Select $\mathbf{z}_n = \arg\max_{\mathbf{x},t} \alpha(\mathbf{x}, t)/\mu_c(\mathbf{x}, t)$ and observe a curve r and a cost c from $f(\mathbf{z}_n)$
4: Compressing the learning curve $r(\mathbf{z}_n)$ into numeric score using Eq. (7).
5: Sample augmented points $\mathbf{z}_{n,m}, y_{n,m}, c_{n,m}, \forall m \leq M$ given the curve and D_n in Eq. (6)
6: Augment the data into D_n and estimate Logistic curve hyperparameters m_0 and g_0.
7: **end for**

The estimated compression curves are illustrated in Right Fig. 2. We summarize the overall algorithm in Alg. 1.

3. Experiments

We demonstrate our proposed model by tuning hyperparameters for two deep reinforcement learning agents on three environments and a convolutional neural networks on two datasets. We provide additional illustrations and experiments in the supplement.

Experimental setup. We use square-exponential kernels for the GPs in our model and estimate their parameters by maximizing the marginal likelihood. We set the maximum number of augmented points to be $M = 15$ and a threshold for a natural log of GP condition number $\delta = 20$. We note that the optimization overhead is much less than the black-box function evaluation time. We follow Wang and de Freitas (2014) to use a slight modification of the expected improvement, i.e., using the incumbent $\mu_n^{\max}$ which is the maximum of GP mean to deal with noise.

Baselines. We extend the discrete (Swersky et al., 2013) to the continuous multi-task BO – which can also be seen as continuous multi-fidelity BO (Kandasamy et al., 2017; Song et al., 2019) because in our setting they both consider cost-sensitive and in the iteration-efficient manner. We, therefore, label the two baselines as continuous multi-task/fidelity BO (CM-T/F-BO). We have another comparison with a CM-T/F-BO using time kernel in Freeze-thaw (Swersky et al., 2014) in the supplement. We do not compare with Fabolas (Klein et al., 2017a) because Fabolas is designed for varying dataset sizes, not iteration axis. We have considered the ablation study in the appendix using a time kernel as the exponential decay proposed in Freeze-thaw method (Swersky et al., 2014).

Learning the compression function. We first illustrate the estimated compression function $l(m_0^*, g_0^*)$ in Right Fig. 2 from different experiments. These Logistic parameters g_0^* and m_0^* are estimated by maximizing the GP marginal likelihood in Eq. (8) and used for compressing the curve. We show that the estimated curve from CartPole tends to reach the highest performance much earlier than Reacher because CartPole is somewhat easier to train than Reacher.

Ablation study of curve compression. To demonstrate the impact of our training curve compression, we compare BOIL to vanilla Bayesian optimization (BO) and with compression (BO-L) given the same number of iterations at $T_{\max}$. We show that using the curve compression will lead to stable performance, as opposed to the existing technique of averaging the last iterations. We plot the learning curves of the best hyperparameters identified by BO, BO-L and BOIL. Fig. 3 shows

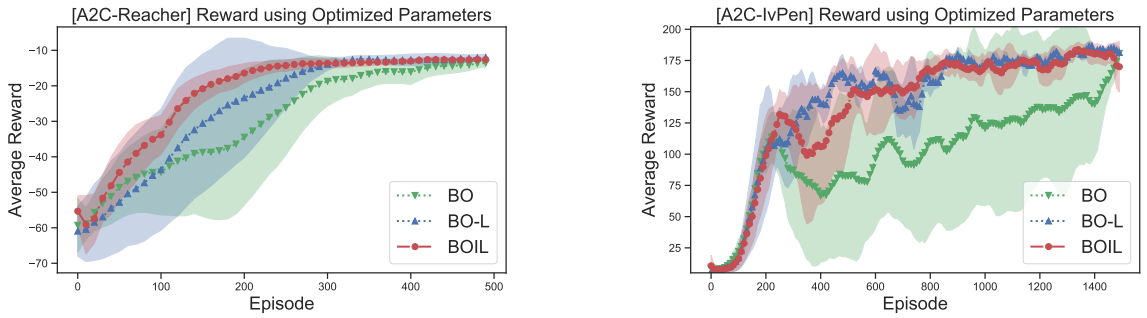

Figure 3: The best found learning curves. The curves show that BO-L and BOIL reliably identify parameters leading to stable training. BOIL takes only half total time to find this optimal curve.

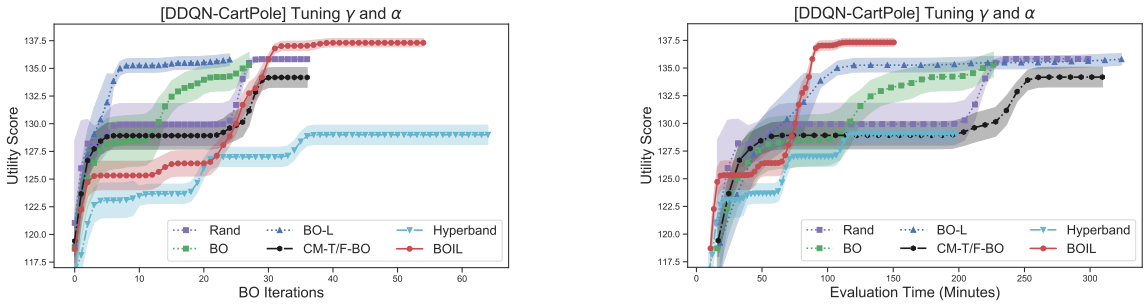

Figure 4: Comparison over BO evaluations (Left) and real-time (Right).

the learning progress over $T_{\max}$ episodes for each of these. The curves are smoothed by averaging over 100 consecutive episodes for increased clarity. We first note that all three algorithms eventually obtain similar performance at the end of learning. However, since BO-L and BOIL take into account the preceding learning steps, they achieve higher performance more quickly. Furthermore, they achieve this more reliably as evidenced by the smaller error bars (shaded regions).

Tuning DRL and CNN. We now optimize hyperparameters for deep reinforcement learning algorithms; in fact, this application motivated the development of BOIL. The combinations of hyperparameters to be tuned, target DRL algorithm and environment are detailed in the supplement.

Fig. 4 illustrates the performance of different algorithms against the number of iterations as well as real-time. The performance is the utility score of the best hyperparameters identified by the baselines. Across all three tasks, BOIL identifies optimal hyperparameters using significantly less computation time than other approaches. The plots show that other approaches such as BO and BO-L can identify well-performing hyperparameters in fewer iterations than BOIL. However, they do so only considering costly, high-fidelity evaluations resulting in significantly higher evaluation times. Hyperband (Li and Jamieson, 2018) exhibits similar behavior in that it uses low fidelity (small t) evaluations to reduce a pool of randomly sampled configurations before evaluating at high fidelity (large t). This puts Hyperband at a disadvantage particularly in the noisy DRL tasks. Since early performance fluctuates hugely, Hyperband can be misled in where to allocate evaluation effort. In contrast to this, BOIL is more flexible than Hyperband in that it can freely explore-exploit the whole joint space. The GP surrogate hereby allows BOIL to generalize across hyperparameters and propagate information through the joint space.

References

E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas. Bayesian optimization in AlphaGo. *arXiv preprint arXiv:1812.06855*, 2018.

Z. Dai, H. Yu, B. K. H. Low, and P. Jaillet. Bayesian optimization meets Bayesian optimal stopping. In *International Conference on Machine Learning*, pages 1496–1506, 2019.

T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

S. Falkner, A. Klein, and F. Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1436–1445, 2018.

Y. Gal, R. Islam, and Z. Ghahramani. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1183–1192, 2017.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1799–1808, 2017.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, pages 528–536, 2017a.

A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. *International Conference on Learning Representations (ICLR)*, 2017b.

A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C. Li, R. Santu, S. Gupta, V. Nguyen, S. Venkatesh, A. Sutti, D. R. D. C. Leal, T. Slezak, M. Height, M. Mohammed, and I. Gibson. Accelerating experimental design by incorporating experimenter hunches. In *IEEE International Conference on Data Mining (ICDM)*, pages 257–266, 2018.

L. Li and K. Jamieson. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:1–52, 2018.

M. McLeod, S. Roberts, and M. A. Osborne. Optimization, fast and slow: Optimally switching between local and Bayesian optimization. In *International Conference on Machine Learning*, pages 3440–3449, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

V. Nguyen and M. A. Osborne. Knowing the what but not the where in bayesian optimization. In *International Conference on Machine Learning*, 2020.

M. Osborne, R. Garnett, Z. Ghahramani, D. K. Duvenaud, S. J. Roberts, and C. E. Rasmussen. Active learning of model evidence using Bayesian quadrature. In *Advances in neural information processing systems*, pages 46–54, 2012.

C. E. Rasmussen. Gaussian processes for machine learning. 2006.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *International Conference on Learning Representations*, 2016.

R. Sen, K. Kandasamy, and S. Shakkottai. Multi-fidelity black-box optimization with hierarchical partitions. In *International conference on machine learning*, pages 4538–4547, 2018.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

J. Song, Y. Chen, and Y. Yue. A general framework for multi-fidelity Bayesian optimization with Gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3158–3167, 2019.

K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Table 1: Notation List

Parameter	Domain	Meaning
d	integer, $\mathscr{N}$	dimension, no of hyperparameters to be optimized
$\mathbf{x}$	vector,$\mathscr{R}^d$	input hyperparameter
N	integer, $\mathscr{N}$	maximum number of BO iterations
$T_{\min}$,$T_{\max}$	integer, $\mathscr{N}$	the min/max no of iterations for training a ML algorithm
t	integer, $\{T_{\min},...T_{\max}\}$	index of training steps
M	integer, $\mathscr{N}$	the maximum number of augmentation. We set $M = 15$.
δ	scalar, $\mathscr{R}$	threshold for rejecting augmentation when ln of cond$(K) > \delta$
m	integer, $\{1,2,...M\}$	index of augmenting variables
n	integer, $\{1,2,...,N\}$	index of BO iterations
$\mathbf{z} = [\mathbf{x},t]$	vector, $\mathscr{R}^{d+1}$	concatenation of the input parameter $\mathbf{x}$ and training iteration t
$c_{n,m}$	scalar, $\mathscr{R}$	training cost (sec)
y_n	scalar, $\mathscr{R}$	transformed score at the BO iteration n
$y_{n,m}$	scalar, $\mathscr{R}$	transformed score at the BO iteration n, training step m
$\alpha(\mathbf{x},t)$	function	acquisition function for performance
$\mu_c(\mathbf{x},t)$	function	GP predictive mean of the cost given $\mathbf{x}$ and t
$r(. \mid \mathbf{x},t)$	function	a raw learning curve, $r(\mathbf{x},t) = [r(1 \mid \mathbf{x},t),...r(t' \mid \mathbf{x},t),r(t \mid \mathbf{x},t)]$
$f(\mathbf{x},t)$	function	a black-box function which is compressed from the above $f()$
$l(. \mid m_0,g_0)$	function	Logistic curve $l(u \mid m_0,g_0) = \frac{1}{1+\exp(-g_0[u-m_0])}$
g_0,g_0^*	scalar, $\mathscr{R}$	a growth parameter defining a slope, $g_0^* = \arg\max_{g_0} L$
m_0,m_0^*	scalar, $\mathscr{R}$	a middle point parameter, $m_0^* = \arg\max_{m_0} L$
L	scalar, $\mathscr{R}$	Gaussian process log marginal likelihood

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

Z. Wang and N. de Freitas. Theoretical analysis of Bayesian optimisation with unknown Gaussian process hyper-parameters. *arXiv preprint arXiv:1406.7758*, 2014.

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003, 2016.

J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *35th Conference on Uncertainty in Artificial Intelligence*, 2019.

Appendix

We summarize all of the notations used in our model in Table 1 for ease of reading.

Appendix A. Related Work in Iteration-Efficient Bayesian Optimization

The first category employs stopping criteria to terminate some training runs early and allocate resources towards more promising settings. These criteria typically involve projecting a final score using earlier training stages. Freeze-thaw BO (Swersky et al., 2014) models the training loss over time using a GP regressor under the assumption that the training loss roughly follows an exponential decay. Based on this projection, training resources are allocated to the most promising settings. Hyperband (Li and Jamieson, 2018; Falkner et al., 2018) dynamically allocates the computational resources (e.g., training epochs or dataset size) through random sampling and eliminates underperforming hyperparameter settings by successive halving.

Attempts have also been made to improve the epoch efficiency of other hyperparameter optimization algorithms, including (Domhan et al., 2015; Klein et al., 2017b; Dai et al., 2019) which predict the final learning outcome based on partially trained learning curves to identify hyperparameter settings that are predicted to under-perform and early-stop it. In the context of DRL, however, these stopping criteria, including the exponential decay assumed in Freeze-thaw BO (Swersky et al., 2014), may not be applicable, due to the unpredictable fluctuations of DRL reward curves. In the supplement, we illustrate the noisiness of DRL training.

The second category (Swersky et al., 2013; Klein et al., 2017a; Kandasamy et al., 2017; Li and Jamieson, 2018; Wu et al., 2019) aims to reduce the resource consumption of BO by utilizing low-fidelity functions which can be obtained by using a subset of the training data or by training the ML model for a small number of iterations. Multi-task BO (Swersky et al., 2013) requires the user to define a division of the dataset into pre-defined and discrete subtasks. Multi-fidelity BO with continuous approximation (BOCA) (Kandasamy et al., 2017) and its hierarchical partition (Sen et al., 2018) extends this idea to continuous settings. Specifically, BOCA first selects the hyperparameter input and then the corresponding fidelity to be evaluated at. The fidelity in this context refers to the use of different number of learning iterations. Analogous to BOCA's consideration of continuous fidelities, Fabolas (Klein et al., 2017a) proposes to model the joint space of input hyperparameter and dataset size. Then, Fabolas optimizes them jointly to select the optimal input and dataset size.

The above approaches typically identify performance of hyperparameters via the average (either training or validation) loss of the last learning iterations. Thereby, they do not account for potential noise in the learning process (e.g., they might select unstable settings that jump to high performance in the last couple of iterations).

Appendix B. Algorithm Illustration and Further Experiments

Fig. 5 illustrates the behavior of our proposed algorithm BOIL for optimizing the discount factor γ of Dueling DQN (Wang et al., 2016) on the CartPole problem. The inclusion of augmented observations in BOIL is also illustrated.

In both cases, we plot the GP predictive mean in Eq. (1), GP predictive variance in Eq. (2), the acquisition function, the predicted function and the final decision function in Eq. (5). These equations are defined in the main manuscript.

As shown in the respective figures the final decision function balances between utility and cost of any pair (γ, t) to achieve iteration efficiency. Especially in situations where multiple locations share the same utility value, our decision will prefer to select the cheapest option. Using the augmented observations in Fig. 5, our joint space is filled quicker with points and the uncertainty (GP variance)

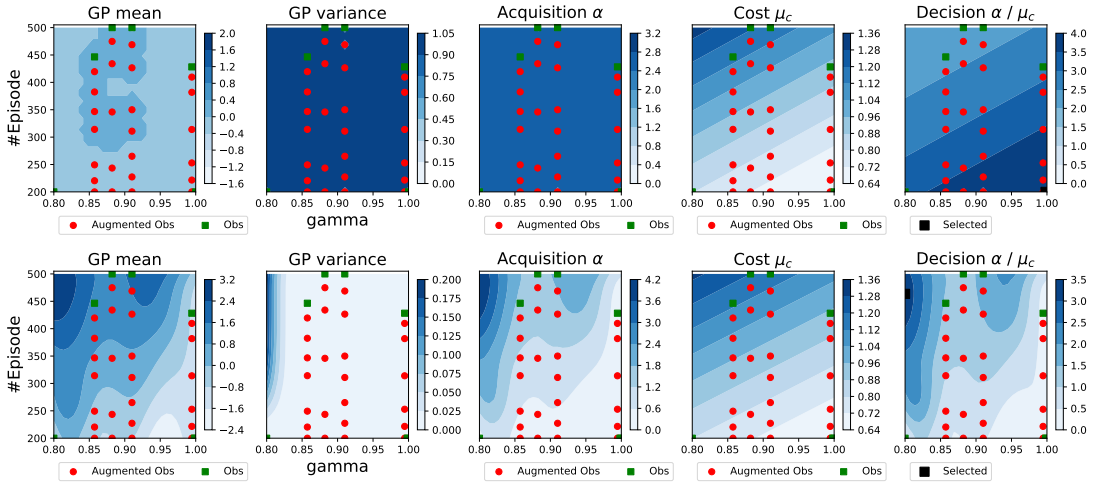

Figure 5: Illustration of BOIL on a 2-dimensional optimization task of DDQN on CartPole. The augmented observations fill the joint hyperparameter-iteration space quickly to inform our surrogate. Our decision balances utility against cost for iteration-efficiency. Especially in situations of multiple locations sharing the same utility value, our algorithm prefers to select the cheapest option.

Table 2: Dueling DQN algorithm on CartPole problem (left) and CNN for SVHN model (right).

Variables	Min	Max	Best Found $\mathbf{x}^*$
discount factor	0.8	1	0.95586
learning rate	$1e^{-6}$	0.01	0.00589
#Episodes	300	800	-

Variables	Min	Max	Best Found $\mathbf{x}^*$
filter size	1	8	5
pool size	1	5	5
batch size	16	1000	8
learning rate	$1e^{-6}$	0.01	0.000484
momentum	0.8	0.999	0.82852
decay	0.9	0.999	0.9746
#epoch	30	150	-

Table 3: A2C algorithm on Reacher (left) and InvertedPendulum (right).

Variables	Min	Max	Best Found $\mathbf{x}^*$	Min	Max	Best Found $\mathbf{x}^*$
γ discount factor	0.8	1	0.8	0.8	1	0.95586
learning rate actor	$1e^{-6}$	0.01	0.00071	$1e^{-6}$	0.01	0.00589
learning rate critic	$1e^{-6}$	0.01	0.00042	$1e^{-6}$	0.01	0.00037
#Episodes	200	500	-	700	1500	-

Table 4: Further specification for DRL agents

Hyperparameter	Value
A2C	
Critic-network architecture	$[32, 32]$
Actor-network architecture	$[32, 32]$
Entropy coefficient	0.01
Dueling DQN	
Q-network architecture	$[50, 50]$
ε-greedy (start, final, #steps)	$(1.0, 0.05, 10000)$
Buffer size	10000
Batch size	64
PER-α Schaul et al. (2016)	1.0
PER-β (start, final, #steps)	$(1.0, 0.6, 1000)$

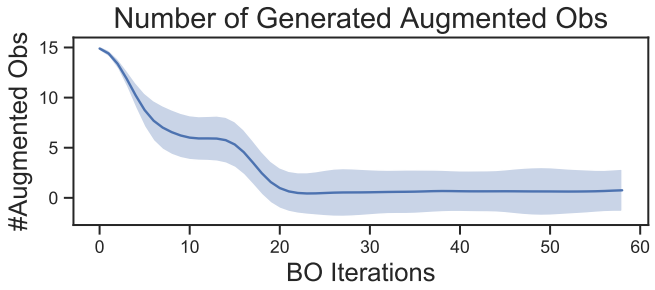

Figure 6: DDQN on CartPole. The number of augmented observations is reducing over time.

across it reduces. A second advantage of having augmented observations is that the algorithm is discouraged to select the same hyperparameter setting at lower fidelity than a previous evaluation. We do not add the full curve as this will make the conditioning problem of the GP covariance matrix.

B.1 Experiment settings

Task descriptions. We consider three DRL settings including a Dueling DQN (DDQN) (Wang et al., 2016) agent in the CartPole-v0 environment and Advantage Actor Critic (A2C) (Mnih et al., 2016) agents in the InvertedPendulum-v2 and Reacher-v2 environments. In addition to the DRL applications, we tune 6 hyperparameters for training a convolutional neural network (LeCun et al., 1998) on the SVHN dataset and CIFAR10.

We summarize the hyperparameter search ranges for A2C on Reacher and InvertedPendulum in Table 3, CNN on SHVN and DDQN on CartPole are in Table 2. Additionally, we present the best found parameter $\mathbf{x}^*$ for these problems. Further details of the DRL agents are listed in Table 4.

B.2 The Number of Augmented Points Over Time

We examine the count of augmented observations generated per iteration in Fig. 6. Although this number is fluctuating, it tends to reduce over time. BOIL does not add more augmented observations at the later stage when we have gained sufficient information and GP covariance conditioning falls below our threshold $\delta = 20$.

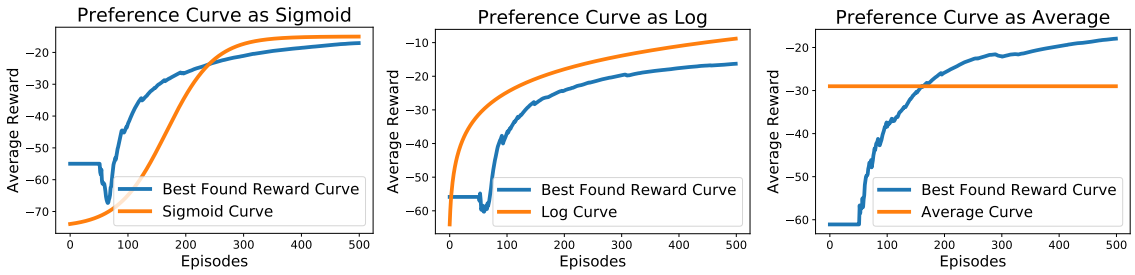

Figure 7: To highlight the robustness, we examine the results using different preference functions such as Sigmoid curve, Log curve, and Average curve on Reacher experiments.

B.3 Robustness over Different Preference Functions

We next study the learning effects with respect to different choices of the preference functions. We pick three preference functions including the Sigmoid, Log and Average to compute the utility score for each learning curve. Then, we report the best found reward curve under such choices. The experiments are tested using A2C on Reacher-v2. The results presented in Fig. 7 demonstrate the robustness of our model with the preference functions.

B.4 Visualizing Logistic Function with Different Parameters

We present the Logistic curve $l(u \mid m_0, g_0) = \frac{1}{1+\exp(-g_0[u-m_0])}$ using different choices of g_0 and m_0 in Fig. 8. We then learn from the data to get the optimal choices g_0^* and m_0^* presented in Fig. 2.

B.5 Ablation Study using FreezeThraw Kernel for Time

In the joint modeling framework of hyperparameter and time (iteration), we can replace the kernel either $k(\mathbf{x},\mathbf{x})$ or $k(t,t)$ with different choices. We, therefore, set up a new baseline of using the time-kernel $k(t,t')$ in FreezeThaw approach Swersky et al. (2014) which encodes the monotonously exponential decay from the curve. Particularly, we use the kernel defined as

$$k(t,t') = \frac{\beta^\alpha}{(t+t'+\beta)^\alpha}$$

for parameters $\alpha, \beta > 0$ which are optimized in the GP models.

We present the result in Fig. 9 that CM-T/F-BO is still less competitive to BOIL using this specific time kernel. The results again validate the robustness our approach cross different choices of kernel.

B.6 Additional Results for Tuning DRL and CNN

We present the additional experiments for tuning DRL model on InvertedPendulum and Reacher environment; and CNN model on SHVN and (subset of) CIFAR10 in Fig. 10. Again, we show that the proposed model clearly gain advantages again the baselines in tuning hyperparameters for model with iterative learning available.

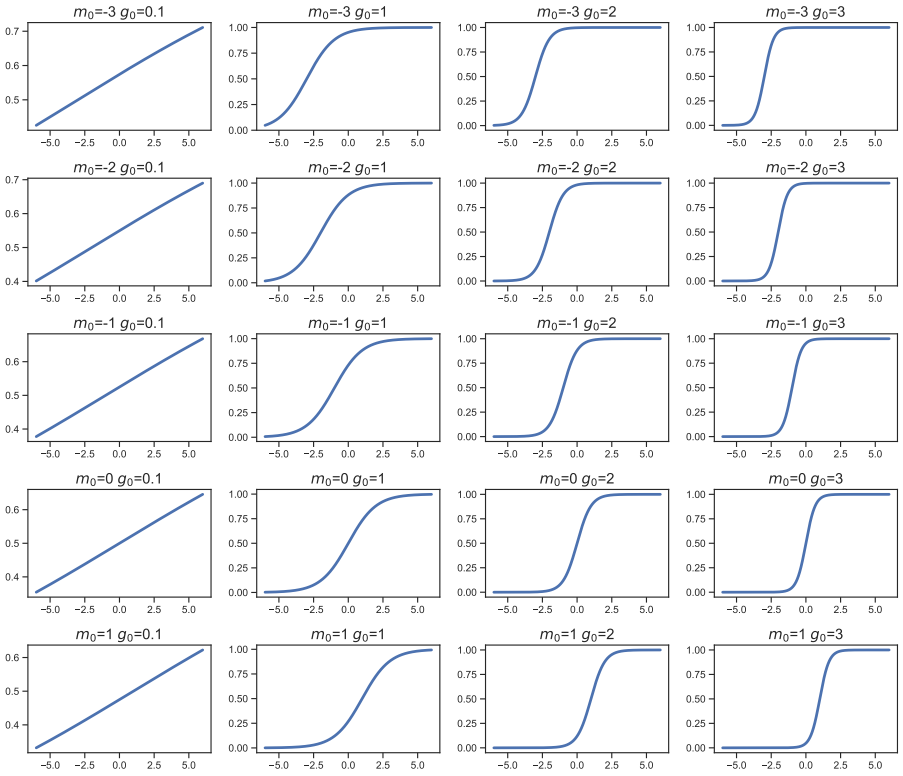

Figure 8: Examples of Logistic function $l(u \mid m_0, g_0) = \frac{1}{1+\exp(-g_0[u-m_0])}$ with different values of middle parameter m_0 and growth parameter g_0.

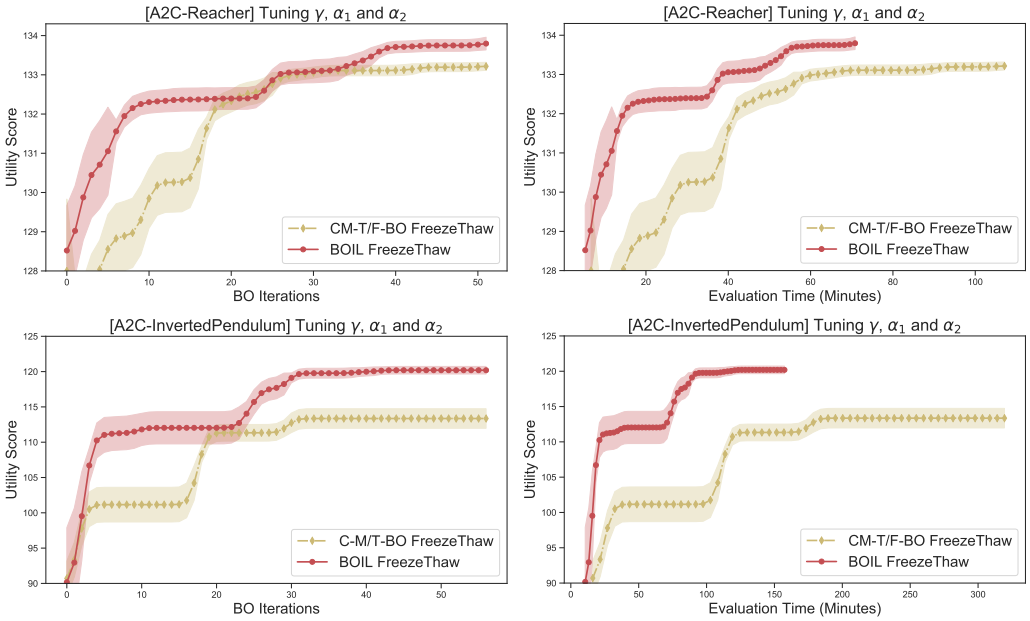

Figure 9: Comparison using freezethaw kernel for time component.

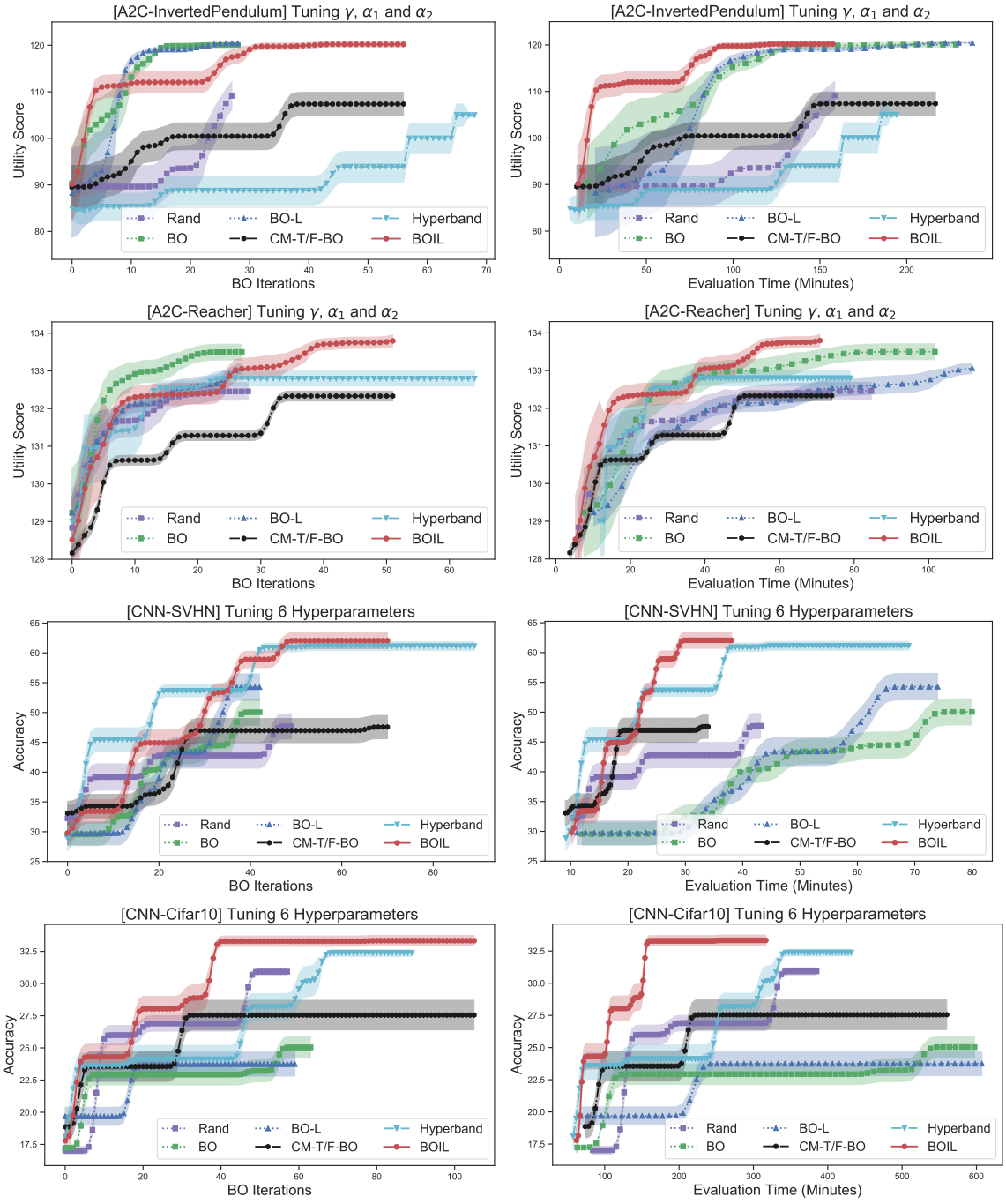

Figure 10: Tuning hyperparameters of a DRL on InvertedPendulum, Reacher and a CNN on CIFAR10 and SVHN.

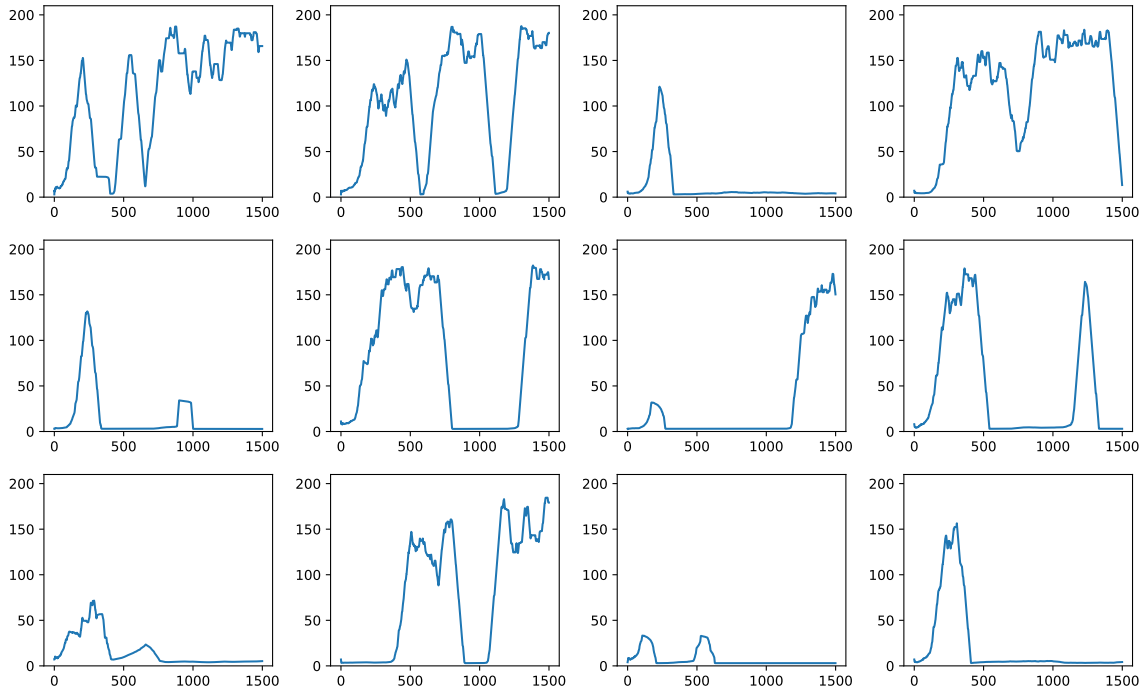

Figure 11: Examples of reward curves using A2C on InvertedPendulum-v2. Y-axis is the reward averaged over 100 consecutive episodes. X-axis is the episode. The noisy performance illustrated is typical of DRL settings and complicates the design of early stopping criteria. Due to the property of DRL, it is not trivial to decide when to stop the training curve. In addition, it will be misleading if we only take average over the last 100 iterations.

B.7 Examples of Deep Reinforcement Learning Training Curves

Finally, we present examples of training curves produced by the deep reinforcement learning algorithm A2C in Fig. 11. These fluctuate widely and it may not be trivial to define good stopping criteria as done for other applications in previous work Swersky et al. (2014).