BOHB: Robust and Efficient Hyperparameter Optimization at Scale

- 作者：Stefan Falkner, Aaron Klein, Frank Hutter
- 机构：University of Freiburg
- 会议：ICML2018
- 地址：https://arxiv.org/abs/1807.01774
- 代码：https://github.com/automl/HpBandSter

论文主要内容

摘要

Bayesian Optimization方法由于需要完整的评估黑盒函数，代价高。而bandit-based基于随机搜索，缺少引导。本文提出将两种组合，相比BO方法和hyperband方法在大量问题类型中都有一致性地提升(高维toy-function、SVM、BNN、FNN、deepRL、CNN).同时方法鲁棒、简单、通用、易于实现。

目前比较公认的SOTA超参优化方法（当然已经不是严格意义上的“黑盒”）

Motivation

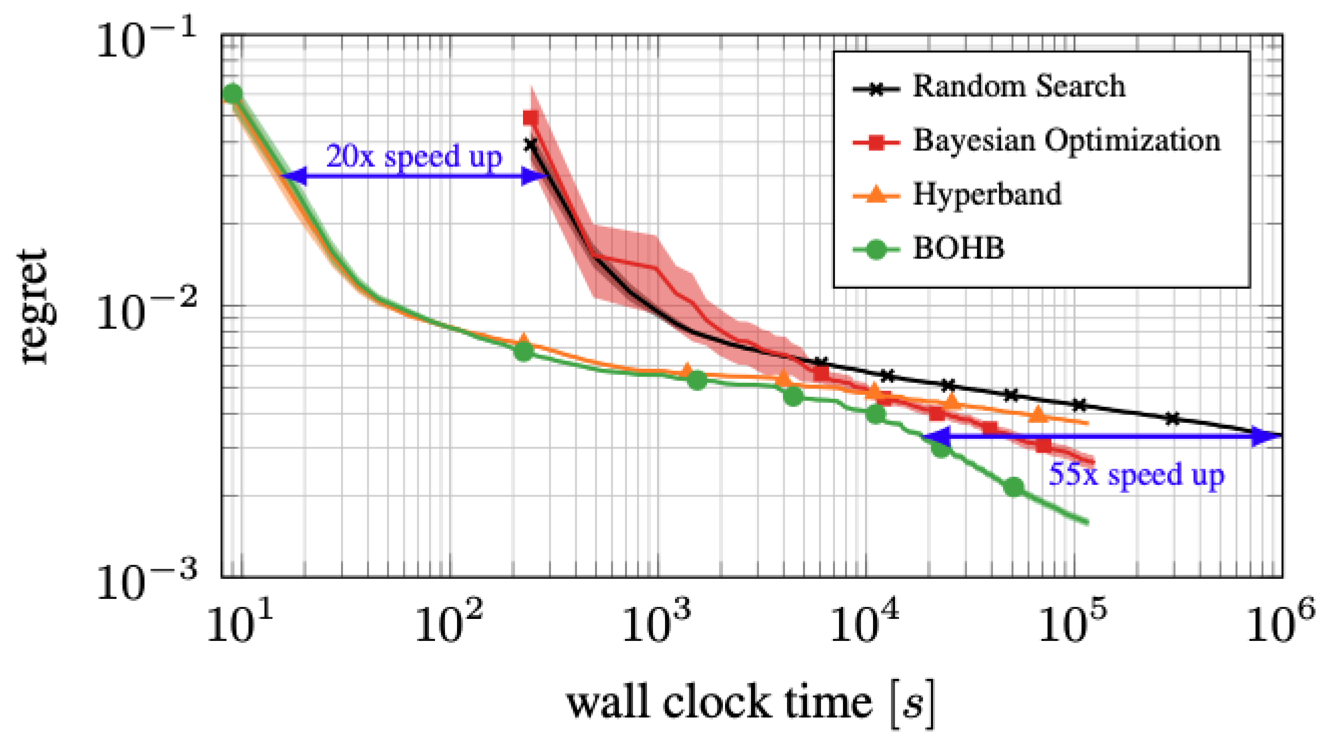


Figure 1. Illustration of typical results obtained, here for optimizing six hyperparameters of a neural network. We show the immediate regret of the best configuration found by 4 methods as a function of time. Hyperband has strong anytime performance, but for larger budgets does not perform much better than random search. In contrast, Bayesian optimization starts slowly (like random search), but given enough time outperforms Hyperband. Our new method BOHB achieves the best of both worlds, starting fast and also converging to the global optimum quickly.

- hyperband因为是随机采样config，所以无法从之前config的表现结果中学习
- budget足够多的时候，hyperband不如bayesian optimization
- hyperband是anytime算法，而BO不是

方法

介绍

BO

$$a(\boldsymbol{x}) = \int \max(0, \alpha - \hat{f}(\boldsymbol{x}))dp(\hat{f} \mid D)$$

1. 选 x ，使得最大化采集函数 a ： $\boldsymbol{x}_{new}=\arg\max_{\boldsymbol{x}\in\mathcal{X}} a(\boldsymbol{x})$
2. 在 x 上evaluate黑盒函数： $y_{\text{new}}=f\left(\boldsymbol{x}_{new}\right)+\epsilon$
3. 增广数据集 $D \leftarrow D \cup\left(\boldsymbol{x}_{new}, y_{new}\right)$ ，重新拟合模型回到*1.*

使用高斯过程模型作为surrogate model，随着观测点的增加，复杂度立方增加

TPE

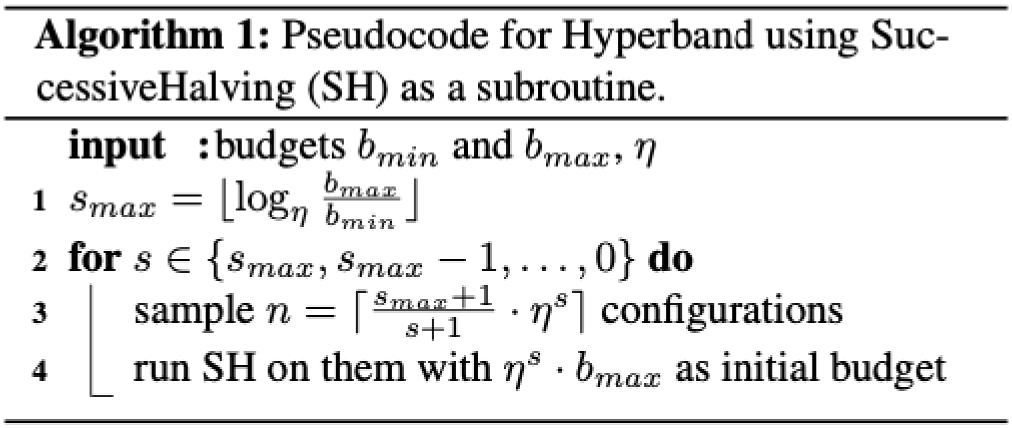
$l(\boldsymbol{x})=p(y<\alpha \mid \boldsymbol{x}, D) \quad g(\boldsymbol{x})=p(y>\alpha \mid \boldsymbol{x}, D)$

选 x 使用 $argmax\frac{l(x)}{g(x)}$ 方式

TPE复杂度随观测点线性增加

HyperBand

- 由于评估黑盒函数 f 很昂贵，采用近似版本 $\hat{f}$ 。
- 近似程度由*budget*来控制 $b \in\left[b_{\min }, b_{\max }\right]$
  - 近似质量随着*buget*增加而提升
  - 可以是神经网络模型的训练epoch数、数据集大小、算法迭代次数



Algorithm 1: Pseudocode for Hyperband using SuccessiveHalving (SH) as a subroutine.

```
    input :budgets $b_{min}$ and $b_{max}$, $\eta$
1   $s_{max} = \lfloor \log_\eta \frac{b_{max}}{b_{min}} \rfloor$
2   for $s \in \{s_{max}, s_{max}-1, \ldots, 0\}$ do
3       sample $n = \lceil \frac{s_{max}+1}{s+1} \cdot \eta^s \rceil$ configurations
4       run SH on them with $\eta^s \cdot b_{max}$ as initial budget
```

方法

Algorithm 2: Pseudocode for sampling in BOHB

```
input  : observations $D$, fraction of random runs $\rho$,
         percentile $q$, number of samples $N_s$,
         minimum number of points $N_{min}$ to build a
         model, and bandwidth factor $b_w$
output : next configuration to evaluate
1 if $rand() < \rho$ then return random configuration
2 $b = \arg\max\{D_b : |D_b| \geq N_{min} + 2\}$
3 if $b = \emptyset$ then return random configuration
4 fit KDEs according to Eqs. (2) and (3)
5 draw $N_s$ samples according to $l'(\boldsymbol{x})$ (see text)
6 return sample with highest ratio $l(\boldsymbol{x})/g(\boldsymbol{x})$
```

- 第1行加入随机比例，保证在最差情况（when the lower fidelities are misleading）下比随机的hyperband差一个常数倍 $\left(\rho^{-1} \cdot (s_{\max} + 1)\right)$
- 第2行每次选择budget最大的数据用来拟合KDE，因为 $\hat{f}$ 近似 f 随着budget增大变准
- 第4行选择用来拟合KDE的good、bad样本

$$\begin{aligned} N_{b,l} &= \max(N_{min}, q \cdot N_b) \\ N_{b,g} &= \max(N_{min}, N_b - N_{b,l}) \end{aligned} \qquad (3)$$

  - 在budget b 下TPE中good样本数和bad样本数选择方式
    - 即保证有足够的样本数(N_{min})
    - 又保证最少的重合度
- 第5行是优化TPE模型(第6行)的技术： l 实际上是概率密度函数， l' 是 l 的bandwidth乘上一个 b_w 的概率密度函数，bandwidth增加，鼓励exploration。（优化6的方式类似local search）

并行

- 算法1中的for可以并行（hyperband）
- SH内的运行本身可以并行
- TPE可以并行：通过限制优化EI的样本数，优化不要太彻底以获得解的diversity

具体并行实施：

1. 从第一个SH（最激进的early-stopping，平均每个config的budget最低）运行开始
2. 根据第一个SH策略的结果用算法2进行采样configs，直到足够多configs够新的SH运行

TPE的observations D 对所有SH共享

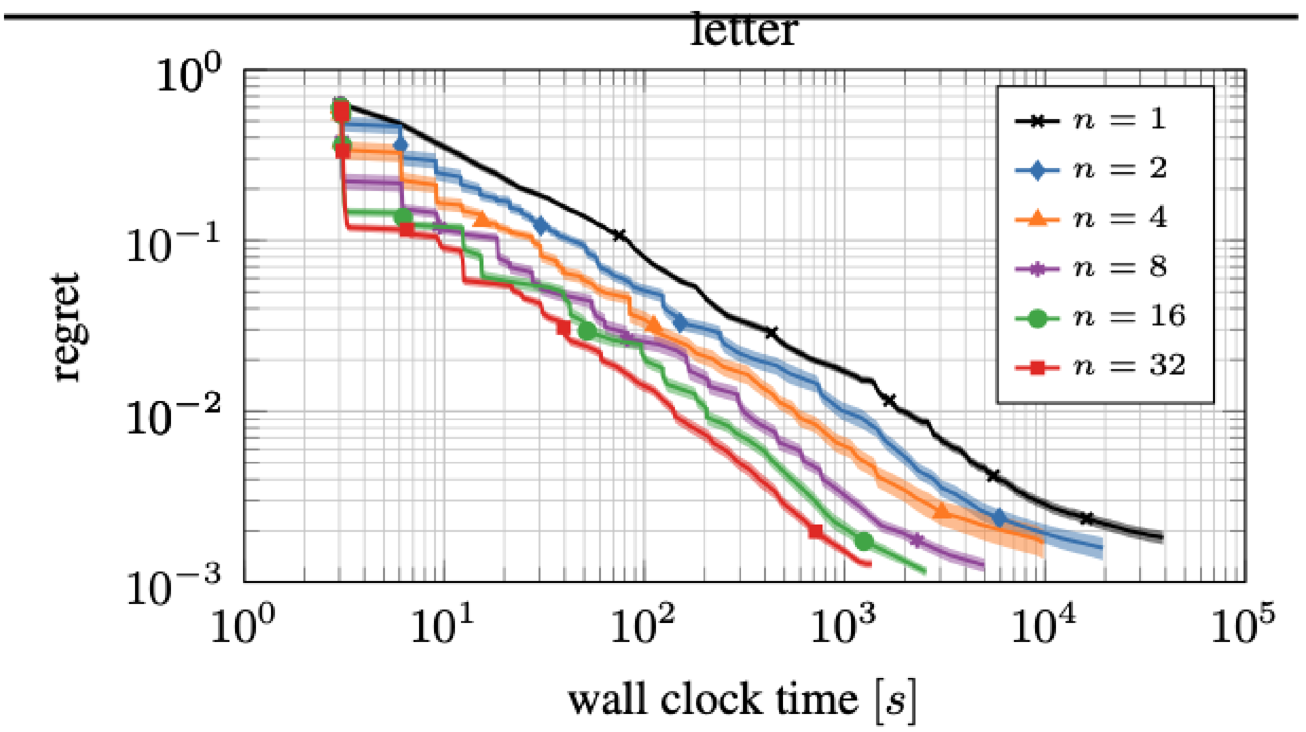


Figure 2. Performance of our method with different number of parallel workers on the letter surrogate benchmark (see Sec. 5) for 128 iterations. The speedup for two and four workers is close to linear, for more workers it becomes sublinear. For example, the speedup to achieve a regret of 10^{-2} for one vs. 32 workers is ca. $2000s/130s \approx 15$. We plot the mean and twice the standard error of the mean over 128 runs.

实验结果

实验设置：

- $\eta = 3$

Results

Toy function：counting ones

$$f(\boldsymbol{x}) = -\left(\sum_{i=0}^{N_{cat}} x_i + \sum_{j=N_{cat}+1}^{N_{cat}+N_{cont}} \mathbb{E}_{X\sim B_j(X)}[X]\right).$$

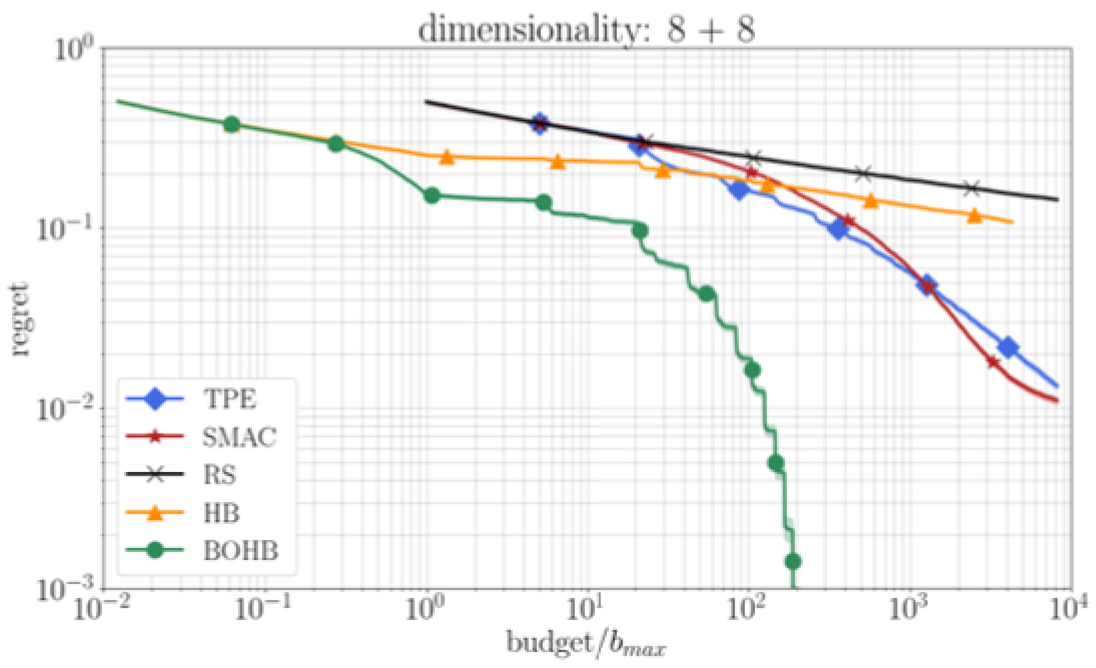


Figure 3. Results for the counting ones problem in 16 dimensional space with 8 categorical and 8 continuous hyperparameters. In higher dimensional spaces RS-based methods need exponentially more samples to find good solutions.

在64维的时候发现SMAC、TPE表现超过BOHB（更noisy，low budget可靠性降低）

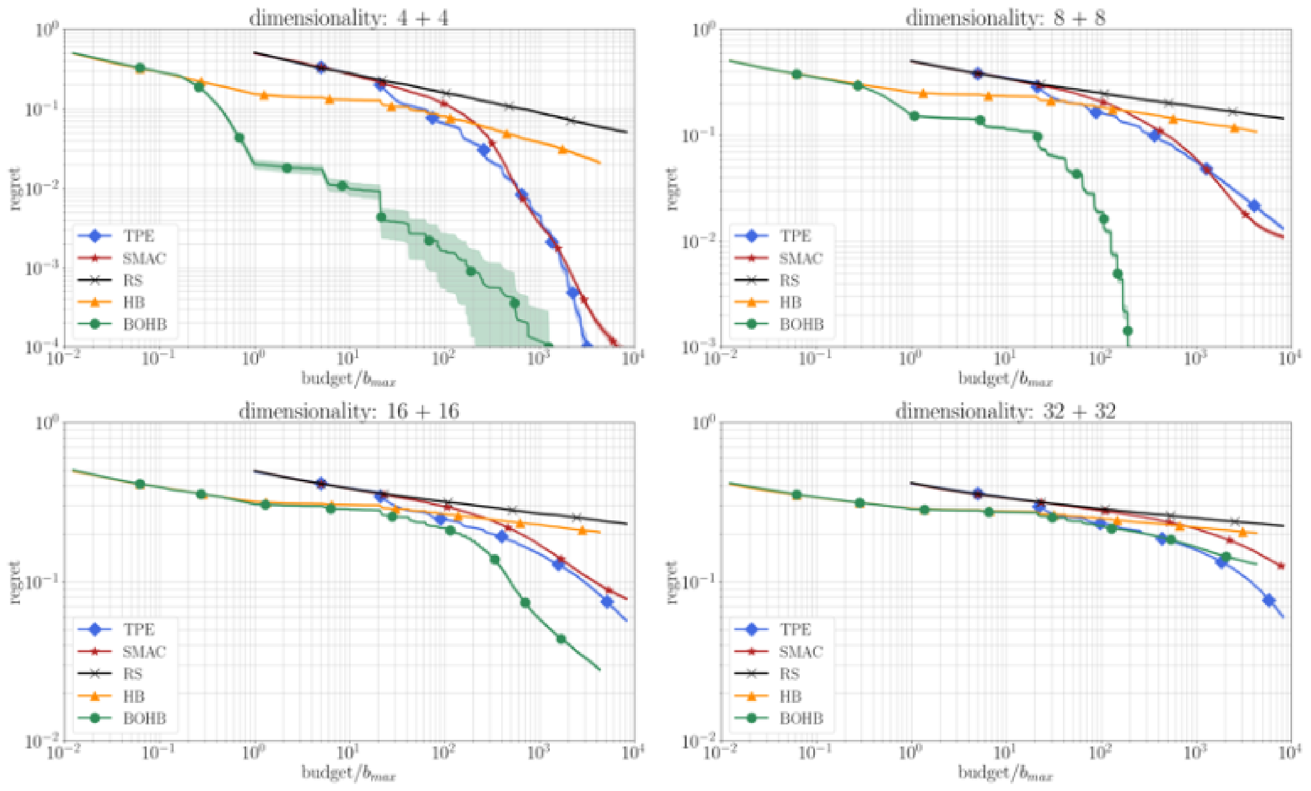


Figure 7. Mean performance of BOHB, HB, TPE, SMAC and RS on the mixed domain counting ones function with different dimensions. As uncertainties, we show the standard error of the mean based on 512 runs.

Surrogate Benchmarks

评估现实的目标函数非常昂贵，Surrogate benchmark是一些offline data来避免实际评估黑盒函数。（可以进行非常多次独立运行，得出更有意义的结论）

SVM

- Search space：正则化超参 C 、kernel参数 γ
- budget：由datapoints数量定义。最小budget为1/512的训练数据，最大budget为整个训练数据

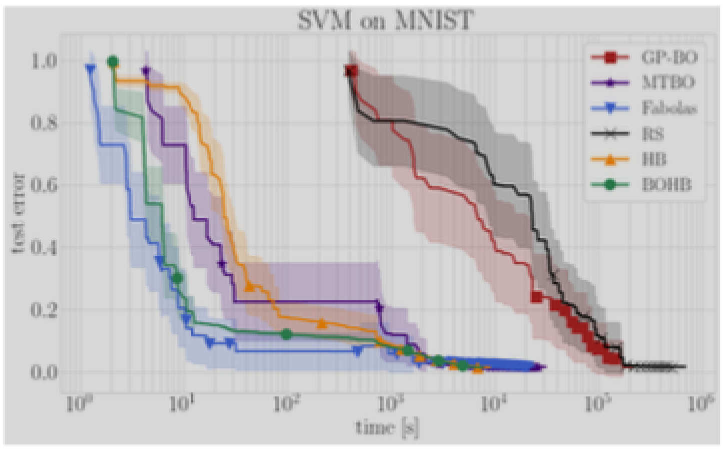


Figure 4. Comparison on the SVM on MNIST surrogates as described in Klein et al. (2017a). BOHB works similarly to Fabolas on this two dimensional benchmark and outperforms MTBO and HB.

Feed-forward NN

- Search space：initial learning rate、batch-size、dropout、exponential-decay for LR、architecture(# of layers、units per layers)
- datasets：6个不同数据集，来自OpenML

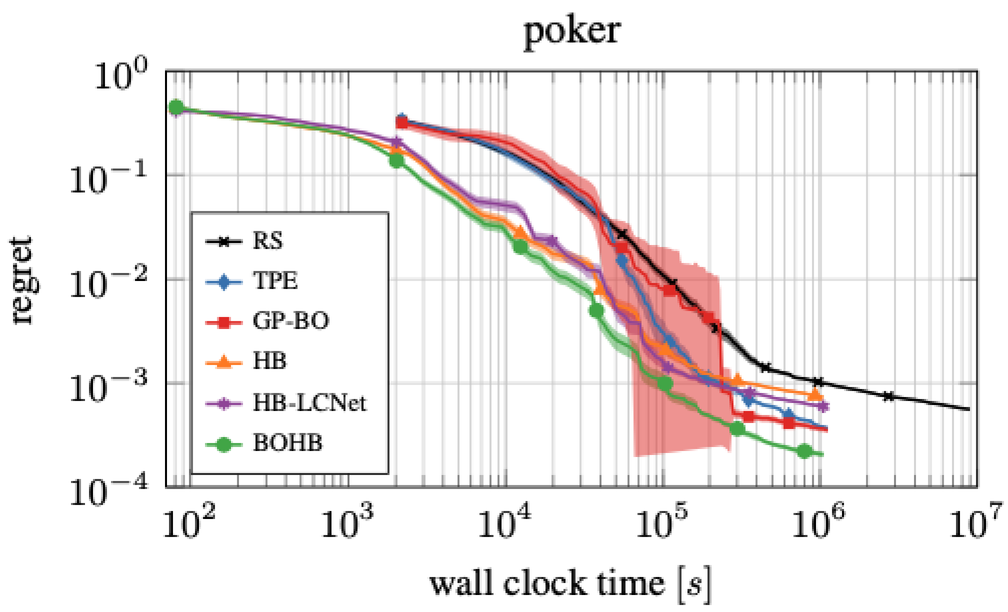


Figure 5. Optimizing six hyperparameter of a feed-forward neural network on featurized datasets; results are based on surrogate benchmarks. Results for the other 5 datasets are qualitatively similar and are shown in Figure 1 in the supplementary material.

Bayesian NN

- 网络架构：由两层全连接的BNN组成，训练方式为MCMC

- Search space：burn-in 长度、每层units数、动量的decay参数
- datasets： *Boston housing* and *protein structure*
- Budget范围：[500步MCMC，10000步MCMC]
- RS、TPE使用完整Budget

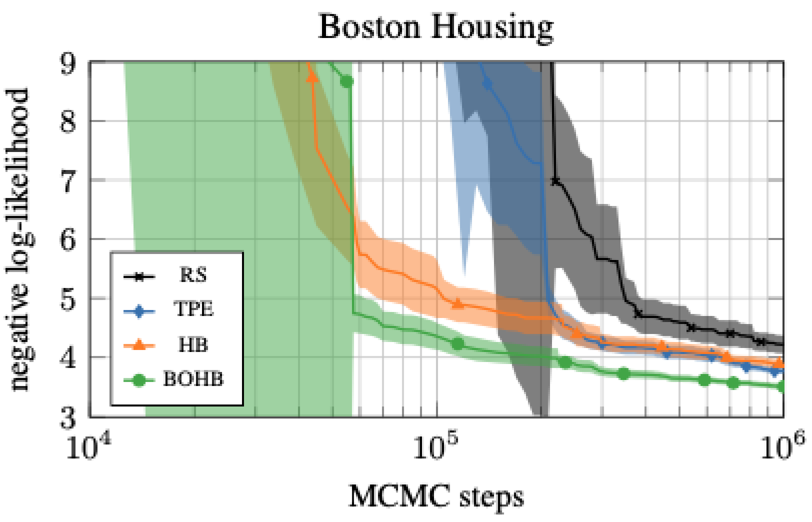


Figure 6. Optimization of 5 hyperparameters of a Bayesian neural network trained with SGHMC. Many random hyperparameter configurations lead to negative log-likelihoods orders of magnitude higher than the best performing ones. We clip the y-axis at 9 to ensure visibility in the plot.

Reinforcement Learning

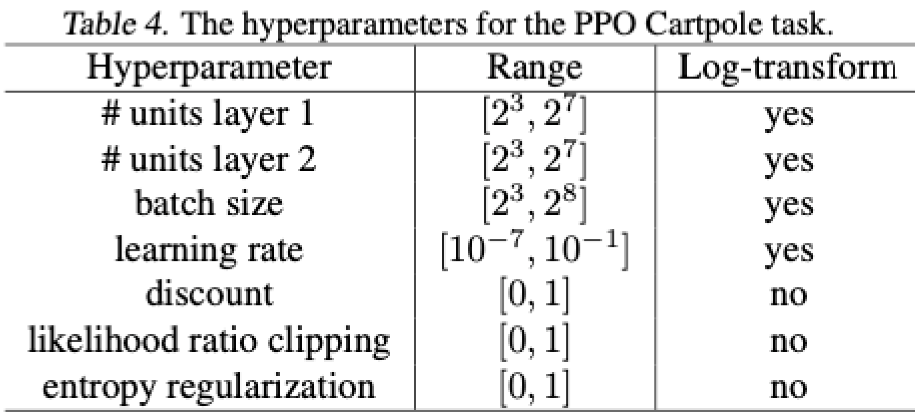


Table 4. The hyperparameters for the PPO Cartpole task.

Hyperparameter	Range	Log-transform
# units layer 1	$[2^3, 2^7]$	yes
# units layer 2	$[2^3, 2^7]$	yes
batch size	$[2^3, 2^8]$	yes
learning rate	$[10^{-7}, 10^{-1}]$	yes
discount	$[0, 1]$	no
likelihood ratio clipping	$[0, 1]$	no
entropy regularization	$[0, 1]$	no

- 优化PPO算法的超参，算法任务为cartpole
- Budgets范围：[1trial, 9trial]

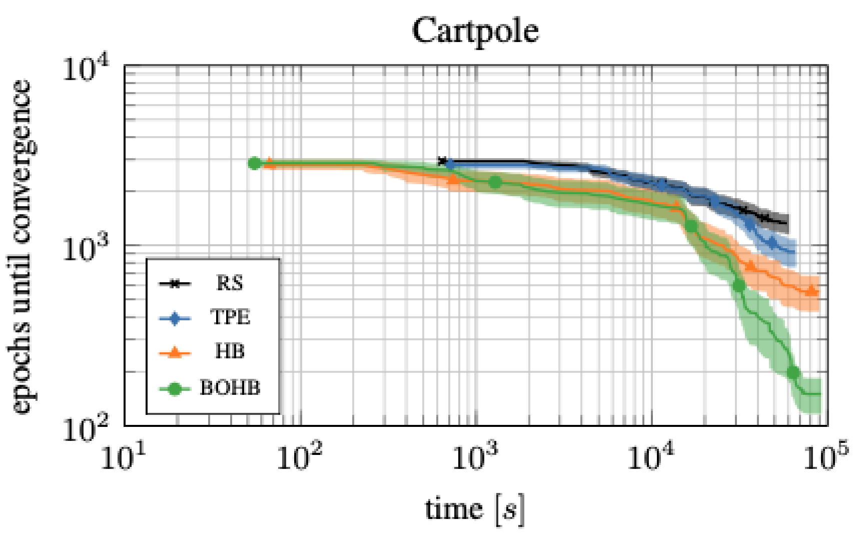


Figure 7. Hyperparameter optimization of 8 hyperparameters of PPO on the cartpole task. BOHB starts as well as HB but converges to a much better configuration.

CNN on CIFAR-10

- 网络架构：20层残差网络
- Search space：LR、动量、weight decay、batch-size
- Budgets：22，66，200，600 epochs
- BOHB整个实验需要33个GPU days（相比之前的RL、ES快60-95倍）
- Test error：2.78% ± 0.09%

超参研究

- 用来优化采集函数的样本数
- 算法中生成随机config的比例： ρ
- 折半比率： η
- Bandwidth 乘子： b_w

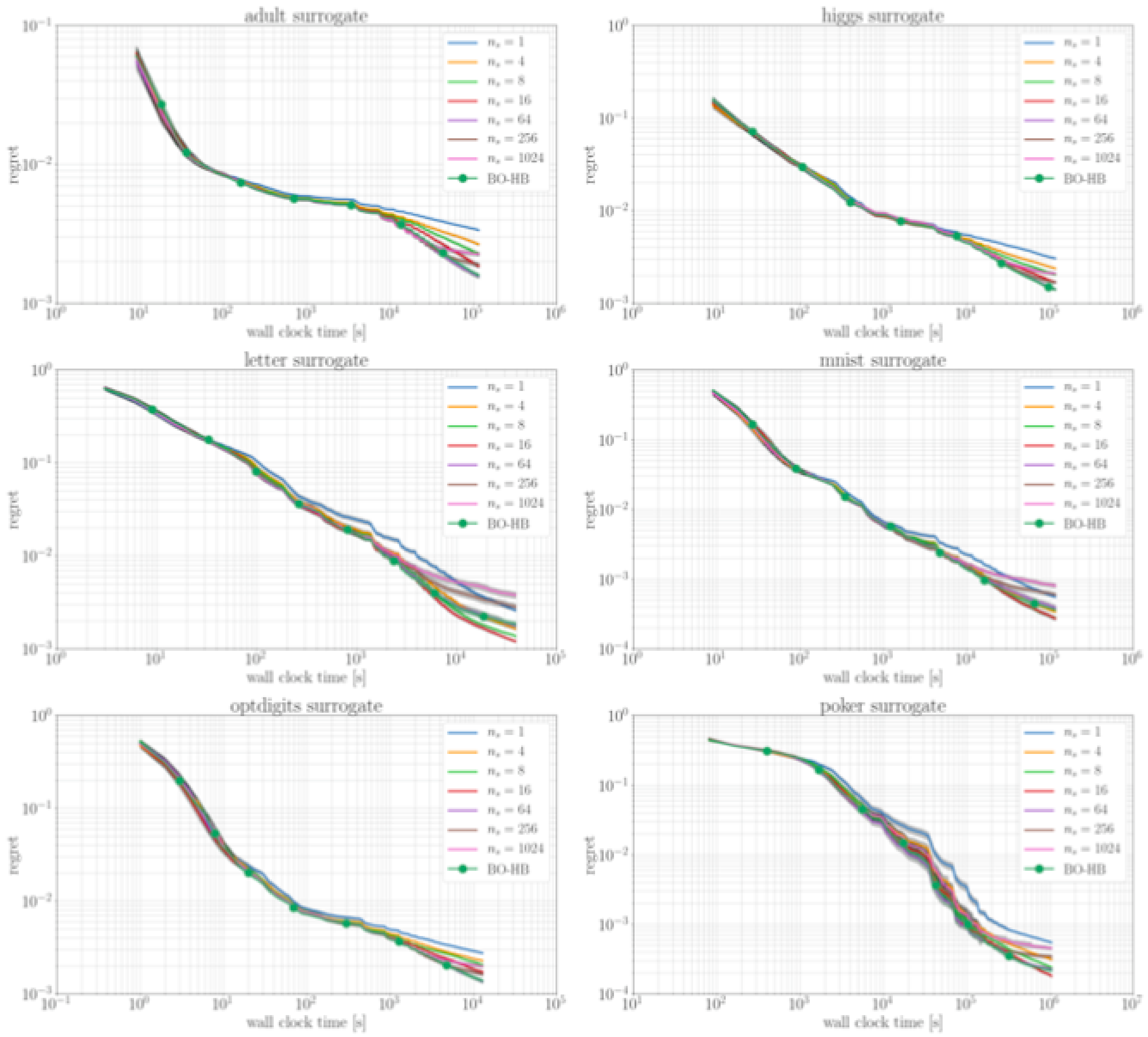


Figure 3. Performance on the surrogates for all six datasets for different number of samples

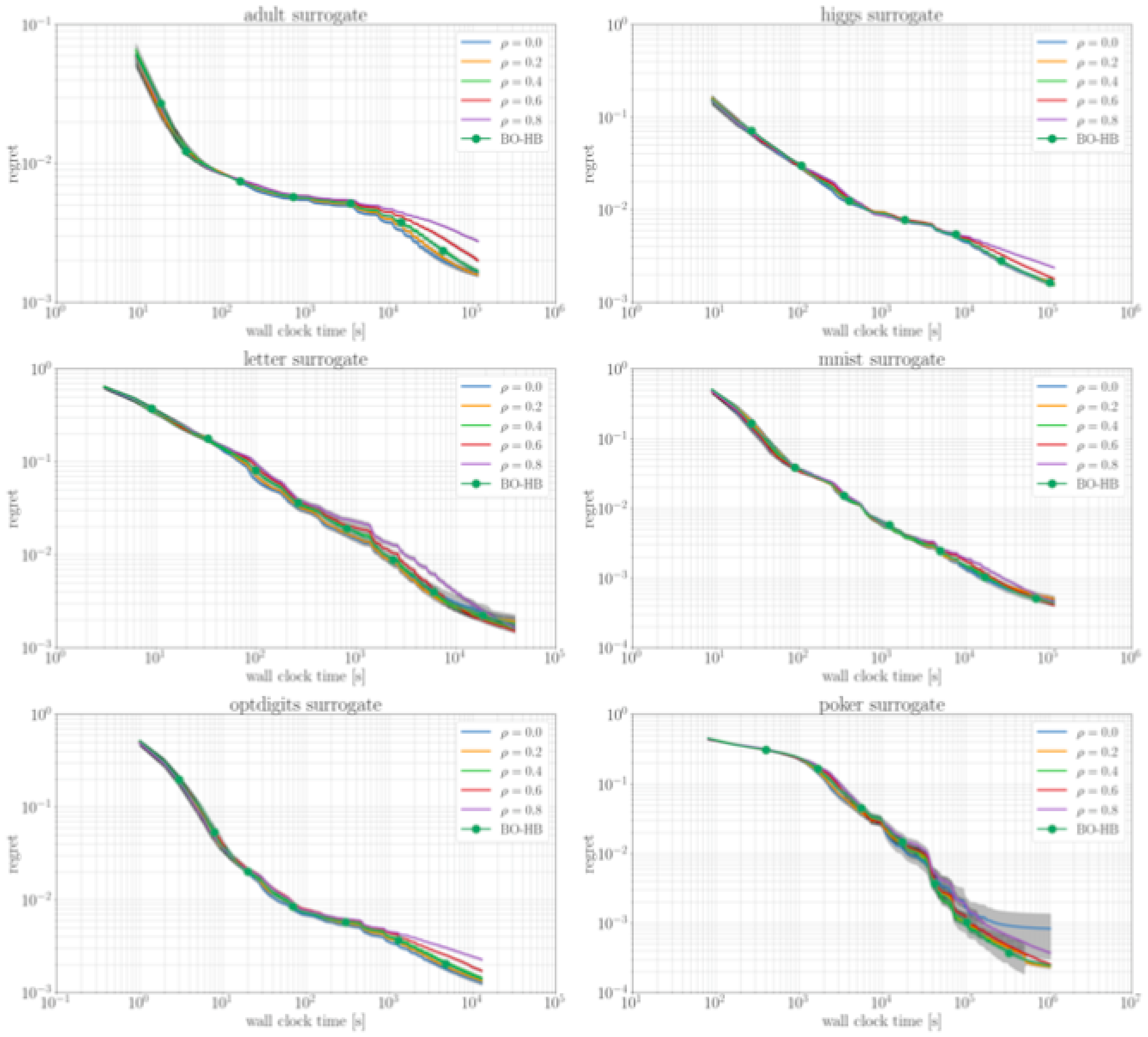


Figure 4. Performance on the surrogates for all six datasets for different random fractions

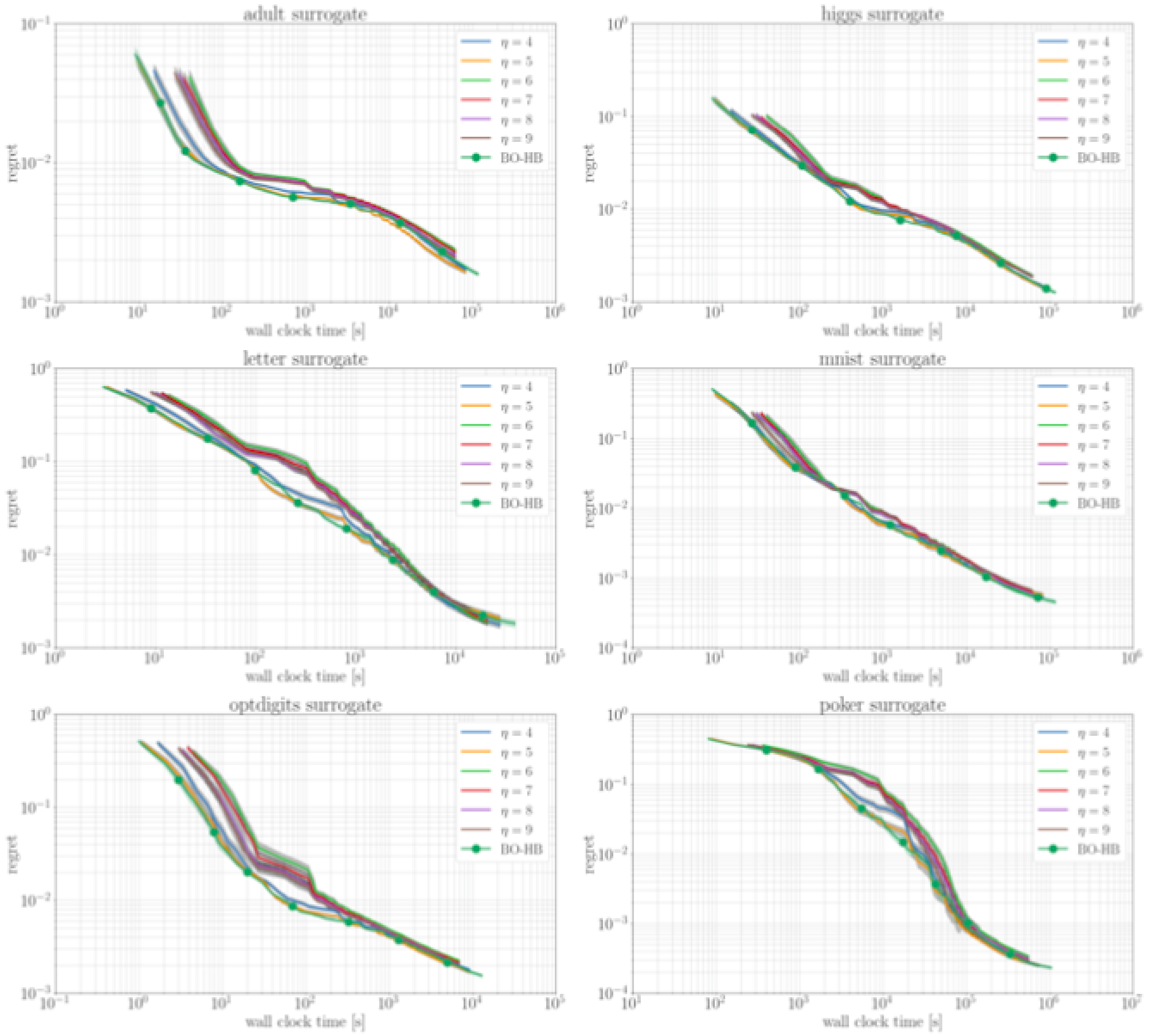


Figure 5. Performance on the surrogates for all six datasets for different values of η.

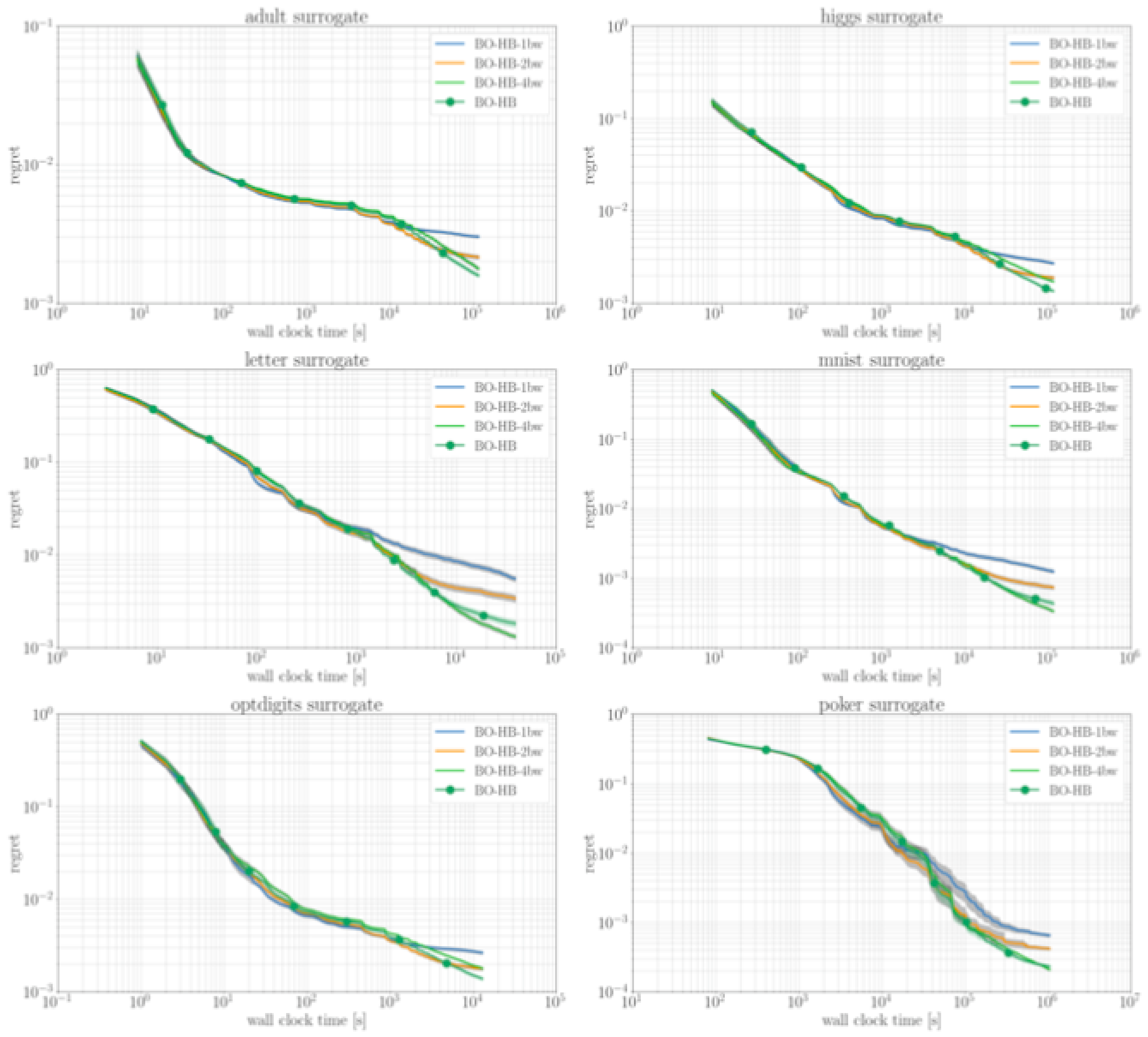


Figure 6. Performance on the surrogates for all six datasets for different bandwidth factors.