e^{2}HPO: energy efficient Hyperparameter Optimization via energy-aware multiple information source Bayesian optimization

Antonio Candelieri[1 [0000−0003−1431−576X]] and Elena Signori[1 [0009−0003−8506−7270]]

Department of Economics Management and Statistics, University of Milano-Bicocca, 20126, Italy
antonio.candelieri@unimib.it,
e.signori1@campus.unimib.it

Abstract. The disclosure of Artificial Intelligence to everyone is significantly pushing the need for resource – especially energy – efficient Machine Learning models. While it is well-established that Artificial Intelligence can enable and support sustainability in different application domains, its own sustainability is a critical concern and an open challenge for research and industry. The need for more accurate Machine Learning models clashes with the fact that a linear gain in accuracy requires exponentially larger resources: a more complex model, more training data and experiments, and consequently more computational resources, entailing a higher energy consumption. This paper proposes an energy efficient hyperparameter optimization algorithm – namely e^{2}HPO – integrating into a unique schema recent advances on both cost-aware and multiple information source Bayesian optimization. Experiments on three common Machine Learning algorithms whose core hyperparameters have been optimized on five different classification datasets empirically prove the benefits of the proposed algorithm. On the other hand, it turned out that some Machine Learning algorithms exhibit an intrinsic energy efficiency and this could lead e^{2}HPO – and similar approaches – to underperform with respect to more naive approaches.

Keywords: GreenAutoML · Hyperparameter optimization · Bayesian Optimization.

1 Introduction

1.1 Rationale and motivation

In the last decade, Machine Learning (ML) and Deep Learning (DL) enabled the development of successful Artificial Narrow Intelligence (ANI) solutions in many domains. More recently, the advent of Generative Pre-trained Transformers (GPTs) has led to the escalating widespread and popularity of Generative AI (GenAI) for text and image generation. The disclosure of GenAI to everyone

implies a large use of these tools in everyday life, but with a limited awareness about their carbon footprint. This is surprising, given that the environmental crisis is one of the open challenges of this century. For example, asking GPT3 for 10-50 responses is equivalent to consuming a 500ml bottle of water [14] to cool the servers, with global AI demand projected to account up to 6.6 billion cubic meters of water withdrawal in 2027, that is more than the total annual water withdrawal of 4 – 6 Denmark or half of the United Kingdom [14]. Despite the efforts to reduce the AI's water consumption *per request*, the total water consumption is expected to rise further as a result of the growing demand for AI services and the increasing scale of AI applications [20].

ChatGPT is estimated to consume, daily, more than half a million kilowatt-hours of electricity to handle approximately 200 million requests. Thus, ChatGPT's energy usage is more than 17'000 times that of an average household. As reported by *Business Insider*, by 2027 the entire AI sector is estimated to consume 85 to 134 terawatthours annually, equivalent to 0.50% of global electricity consumption. In its environmental report, released in July 2024, Google declares that its carbon emissions rose by 48% largely due to AI and data centers.

The environmental impact of an AI model depends not only on its use, but also development, training, and validation. As reported in [8] and [22], training a Large Language Model (LLM) by also optimizing its hyperparameters leads to 300'000 kg of CO_2 emissions, equivalent to 125 round-trip flights between New York and Beijing. Focusing on improving AI model's accuracy without considering energy efficiency is usually (and negatively) referred as Red AI [8]: as a matter of fact, to obtain a linear gain in accuracy, an exponentially larger model is required, with consequent need for more training data, experiments, and computational resources, and, therefore, a worse carbon footprint.

As recently stated in [24], "*while AI for sustainability is quite commonly known, sustainability of AI has long been less of a concern*". Fortunately, researchers are directing their attention at making AI more respectful of environmental sustainability. In 2020, [19] coined the term Green AI (in opposition to Red AI) which "*refers to AI research that yields novel results while taking into account the computational cost, encouraging reduction in resources spent*". We are recently observing a paradigm shift especially in the ML community: [26] provides a review of existent Green AI literature and reports that 76% of the papers have been just published starting from 2020.

Moreover, the crucial importance of energy/resource efficiency in AI is strongly confirmed by most of the today news around DeepSeek – at the time of writing – with global investors dumping tech stocks as they are worried that the emerging low-cost Chinese AI models would threaten the dominance of AI leaders like Nvidia, evaporating $593 billion of the chipmaker's market value, a record one-day loss for any company on Wall Street[1].

This paper proposes a new energy efficient algorithm for Hyperparameter Optimization (HPO) of ML algorithms, namely e^{2}HPO.

[1] https://www.reuters.com/technology/chinas-deepseek-sets-off-ai-market-rout-2025-01-27

1.2 Accuracy vs energy consumption

As empirically demonstrated in [13], there exists a relation between the values of the hyperparameters of a ML algorithm and the computational time for its training and validation (results referred to 5000 randomly selected hyperparameter configurations, for five common ML algorithms). Although training time can be considered a proxy of energy consumption and CO_2 emissions, as recently discussed in [24], it is not a direct measure of these two metrics. In this paper we use a new software solution, namely **CodeCarbon**[2], to monitor energy consumed for training and validating a ML algorithm given a specific configuration of its hyperparameters and a target dataset.

Energy consumptions and accuracies collected on a 15×15 grid of hyperparameters configurations, for three common ML algorithms, lead us to observe three different behaviors, with accuracy and energy consumption that can be: inversely correlated (Figure 1), correlated (Figure 2), or uncorrelated (Figure 3).

This characterization is important because it means there are cases where searching for the most accurate hyperparameters configurations can lead to a reduction in the accumulated energy consumption and, consequently, sample-efficiency translates into energy-efficiency.

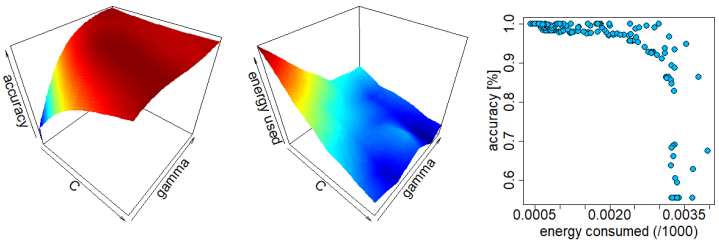

Fig. 1: SVC on banknote authentication dataset: accuracy (left) and energy consumption (middle) depending on SVC's hyperparameters ($\log_{10}$ scaled). Accuracy vs energy consumed (right).

[2] `https://codecarbon.io`

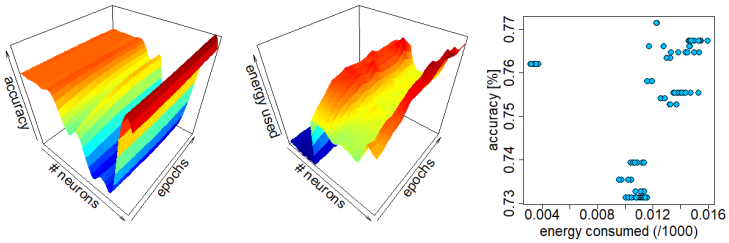

Fig. 2: MLP on blood transfusion dataset: accuracy (left) and energy consumption (middle) depending on MLP's hyperparameters. Accuracy vs energy (right).

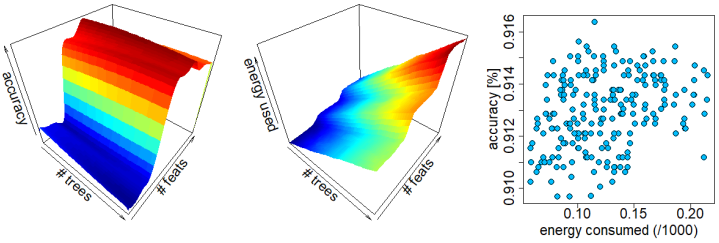

Fig. 3: RF on phoneme dataset: accuracy (left) and energy consumption (middle) depending on RF's hyperparameters. Accuracy vs energy consumed (right).

1.3 Related works

Automated ML (AutoML) – especially HPO – solutions are mostly based on Bayesian Optimization (BO) [2,10], which has also been recently used to investigate the impact of topic models' hyperparameters on the accuracy in Natural Language Processing tasks [23]. The emerging need for HPO methods leading to accurate but energy-efficient ML models required to extend the *vanilla* BO algorithm leading to GreenAutoML approaches.

One possibility is given by *multi-fidelity* and *multiple information source* BO methods, using *successive-halving* [11,12] or small portions of the original large dataset [6], also for multi-objective HPO (e.g., simultaneous maximization of accuracy and fairness, like in [7]) and combinatorial problems [18].

Another possibility is represented by *cost-aware* BO methods, where the seminal work [21] proposed to penalize the acquisition function, specifically the Expected Improvement (EI), by the *location-dependent* cost, $c(\mathbf{x})$. However, this approach resulted biased towards cheap locations, performing well only in the cases where optima are relatively cheap. To overcome this undesired behaviour, [13] proposed CArBO (Cost Apportioned BO), consisting of two consecutive

stages: (i) a cost-effective selection of initial solutions and (ii) a cost-cooling strategy where the penalty associated to the cost is modulated according to the query cost incurred so far. In [15] a cost-aware acquisition function based on Information Directed Sampling (IDS), and named CostIDS, balances cost along with regret and information gain. However, an additional constraint is introduced, in optimizing CostIDS, to avoid that extremely cheap points are chosen repeatedly without any significant increase in information. More recently, [27] has proposed an acquisition function for cost-aware BO based on a previously-unexplored connection with the Pandora's Box problem (from economics). The Pandora's Box problem admits a Bayesian-optimal solution based on the Gittins index, which can be seen as an acquisition function: empirical results demonstrate that the approach performs well, especially in medium-high dimensions.

Finally, a recent survey on GreenAutoML is provided in [24], even if very recent studies combining the two research directions (i.e., cost-aware and multiple information source optimization) are missing, such as [9] and – obviously – the approach presented in this paper.

2 The e^{2}HPO algorithm

The energy-efficient HPO method proposed in this paper, namely e^{2}HPO is based on the multiple information source (Bayesian) optimization approach initially proposed in [5] and successively refined to implement green and fair-and-green HPO tasks [6,7], that are, respectively, a single- and a bi-objective global optimization problem under multiple information sources. The aim of e^{2}HPO is to embed, in an integrated schema, recent advances from both multiple information source and cost-aware BO.

2.1 Energy saving by using a reduced dataset

Dealing with multiple information sources means choosing wether the next promising solution has to be evaluated on the actual expensive function (aka *ground-truth*) or a cheap approximation of it, while keeping limited the overall cost accumulated along the optimization process.

In a HPO task, the next promising solution is a hyperparameters' configuration, the ground-truth is the associated k-fold cross (kFCV) accuracy computed by using the entire dataset, while the cheap approximation is the same metric computed by using a small stratified portion of the dataset. In our experiments k=10 and the cost is the energy consumed to compute the 10FCV accuracy: although it is surely lower for the small dataset, nothing can be said a-priori about the quality of approximation for a certain hyperparameters' configuration, motivating the adoption of a multiple information source optimization method.

Denote with $\mathcal{D}_f = \{(\mathbf{x}^{(i)}, y_f^{(i)})\}_{i=1:n_f}$ and $\mathcal{D}_r = \{(\mathbf{x}^{(i)}, y_r^{(i)})\}_{i=1:n_r}$ the hyperparameters' configurations and their accuracies obtained by using the *full* and the *redux* dataset, respectively. A GP regression model is first fitted to $\mathcal{D}_f$

and then $\mathcal{D}_f$ is *augmented* by including queries in $\mathcal{D}_r$ which can be considered *reliable*, as follows:

$$\widehat{\mathcal{D}} \leftarrow \mathcal{D}_f \cup \left\{ (\mathbf{x}^{(i)}, y_r^{(i)}) \in \mathcal{D}_r : \left| \mu_f(\mathbf{x}^{(i)}) - y_r^{(i)} \right| \leq m \, \sigma_f(\mathbf{x}^{(i)}) \right\} \tag{1}$$

with $\mu_f(\mathbf{x})$ and $\sigma_f(\mathbf{x})$ the predictive mean and uncertainty of the GP fitted on $\mathcal{D}_f$, and m a parameter to manage the reliability of cheap observations (larger values of m increase the number of cheap queries from $\mathcal{D}_r$ considered reliable and included in $\widehat{\mathcal{D}}$). The most suitable value empirically suggested in [5] is $m = 1$.

Finally, another GP regression model is fitted to the augmented set of queries $\widehat{\mathcal{D}}$ leading to the so-called Augmented GP (AGP) that is at the core of [5,6,7]. It is important to remark that $\widehat{\mathcal{D}}$ is computed at each iteration of the optimization process, according to (1), thus it changes over time due to oncoming observations.

2.2 Embedding energy-awareness

Multiple information source optimization (MISO) methods are, in some sense, energy-efficient by-design, because they try to exploit cheap sources along the optimization process to keep low the accumulated query cost. However, they rely on the assumption that the cost entailed to query a source is constant over the search space $\mathcal{X}$, so they are not energy-aware in the sense of dealing with *location-dependent cost* [21,13]. A practical example about the importance of energy-awareness in a MISO schema is given by the following situation: an MLP with a large number of neurons and epochs on a small portion of the original dataset could require a larger training time (and therefore energy) than a smaller MLP on the original dataset.

The proposed e^{2}HPO algorithm aims at properly embedding energy-awareness into a MISO schema. To achieve this goal, we have to first define other two sets, $\mathcal{E}_f = \{(\mathbf{x}^{(i)}, e_f^{(i)})\}_{i=1:n_f}$ and $\mathcal{E}_r = \{(\mathbf{x}^{(i)}, e_r^{(i)})\}_{i=1:n_f}$, collecting the energy consumptions of the hyperparameter configurations evaluated by using the full and the redux dataset, respectively.

The acquisition function stems from [5,7] and is slightly modified to properly address energy-awareness. Specifically, the next promising hyperparameter configuration to evaluate and the source to use are identified by solving:

$$(s, \mathbf{x}) \in \underset{\substack{s \in \{f,r\} \\ \mathbf{x} \in \mathcal{X}}}{\arg\max} \quad \frac{\left[\widehat{\mu}(\mathbf{x}) + \beta \, \widehat{\sigma}(\mathbf{x})\right] - \widehat{y}^+}{1 + e_s^{\uparrow}(\mathbf{x}) \left| \widehat{\mu}(\mathbf{x}) - \mu_s(\mathbf{x}) \right|} \tag{2}$$

where $\widehat{\mu}(\mathbf{x}) + \beta \ \widehat{\sigma}(\mathbf{x})$ is simply the AGP's upper confidence bound and $\widehat{y}^+ = \max\{y : (\mathbf{x}, y) \in \widehat{\mathcal{D}}_f\}$. Thus, the numerator of (2) is the most optimistic improvement[3] with respect to $\widehat{y}^+$, and only depends on $\mathbf{x}$ and the current AGP. This quantity is penalized by the amount at the denominator, where $|\mu_s(\mathbf{x}) - \widehat{\mu}(\mathbf{x})|$ is a simple and efficient discrepancy measure between the AGP's

[3] it could be negative, so in that case is the minimum worsening

predictive mean and that of the GP fitted to $\mathcal{D}_s$, with $s \in \{f, r\}$. Finally, $e_s^{\uparrow}(\mathbf{x})$ is an upward estimate of the energy consumed to evaluate the hyperparameter configuration $\mathbf{x}$ on the source s. Specifically, $e_s^{\uparrow}(\mathbf{x})$ is the upper confidence bound of another GP regression model fitted to $\mathcal{E}_s$.

In summary, in the presence of two information sources, namely f and r, our e^{2}HPO algorithm uses five GPs: two for modeling 10FCV accuracy with respect to the full dataset ($s = f$) and the redux dataset ($s = r$), respectively, an AGP integrating them into a unique model, and two GPs to approximate the energy consumption with respect to the full and redux dataset, separately.

3 Experiments

3.1 ML algorithms to optimize and datasets

In our experiments we have considered three different ML algorithms, specifically: (*i*) a Support Vector Machine Classifier (SVC) with regularization parameter C and radial kernel, (*ii*) a MultiLayer Perceptron (MLP), and (*iii*) Random Forest (RF). For each algorithm we have just considered two crucial hyperparameters to optimize, which are well-known to affect accuracy and training time[4]. Hyperparameters and associated search spaces are reported in Table 1.

Algorithm	Hyperparams	Type	Domain	Scaling
SVC	C	Real	$\{0.01, ..., 100\}$	$\log_{10}$
	γ	Real	$\{0.01, ..., 100\}$	$\log_{10}$
MLP	Size	Integer	$\{\max\{8, n_f\}, .., \lfloor 1.5 * \max\{8, n_f\}\rfloor\}$	min-max
	Epochs	Integer	$\{100, ..., 300\}$	min-max
RF	N_{tree}	Integer	$\{300, ..., 700\}$	min-max
	p_{feats}	Real	$\{0.25, ..., 0.75\}$	min-max

Table 1: Search spaces of the ML algorithms for the HPO task.

As far as MLP is concerned, we have to clarify that its architecture is prefixed and consists of three hidden layers. The number of neurons in the first hidden layer is given by the hyperparameter *Size*, and the number of neurons in the next hidden layers is iteratively halved (and rounded), as shown in Figure 4. Thus, the value of *Size* uniquely determines the overall structure of the network.

For the RF algorithm, N_{tree} and p_{feats} denote, respectively, the number of trees in the forest and the percentage of input features sampled in the decision tree learning procedure. Finally, C and γ are, respectively, the regularization of the SVM classifier and the hyperparameter of its radial basis function kernel.

[4] Although other *arguments* are available and commonly treated as hyperparameters, their impact on accuracy and training time is not so easy to estimate. They are usually related to the specific implementation of the learning algorithm, such as the optimization method internally used by the MLP to minimize the training loss.

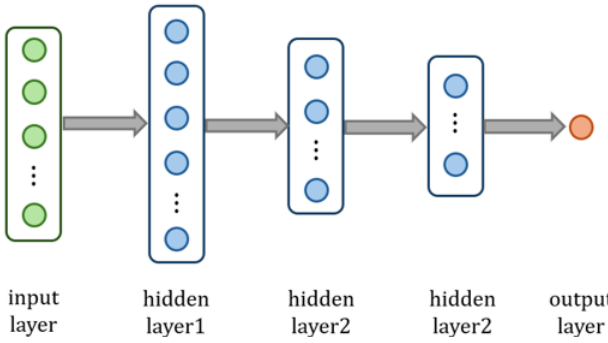

Fig. 4: MLP's basic architecture.

HPO of these three ML algorithms has been performed on five different datasets, all related to binary classification problems. The main characteristics of these datasets are summarized in Table 2.

Dataset Name	Observations	Features	Class 1	Class 2
Banknote authentication	1372	4	762	610
Blood transfusion	748	4	570	178
Heloc	10000	22	5000	5000
Phoneme	5404	5	3818	1586
WDBC	569	30	357	212

Table 2: Main characteristics of the datasets considered in this study.

The datasets were downloaded from OpenML[5] and underwent to the same pre-processing procedure, where input features – all numerical – were rescaled using min-max scaling. After pre-processing, a redux version of each dataset was generated by randomly selecting 10% of the observations while preserving the original class distribution through stratified sampling.

3.2 Performance metric and energy monitoring

As already stated, the 10FCV accuracy of a given hyperparameter configuration is the target performance metric for our HPO tasks.

To monitor the amount of energy consumed in computing the target metric, we have adopted CodeCarbon, a lightweight software package, compatible with Python, created to quantify energy consumed and the amount of CO_2 emissions produced by a computing task running on a monitored system. Although

[5] OpenML ids of the original (not preprocessed) datasets: Banknote authentication 1462, Blood transfusion 1464, Heloc 45026, Phoneme 1489, WDBC 1510.

CodeCarbon provides many other functionalities (like the possibility to compute CO_2 emissions depending on different energy mixes), in this paper we have just considered the amount of energy consumed, measured as the power supplied to the hardware (GPU, CPU, RAM, and overall) at frequent time intervals.

3.3 Limitations implied by the usage of CodeCarbon

As already remarked in recent literature, CodeCarbon has some limitations. First, it relies on the data provided by the computer's components (e.g., Intel Power Gadget for Intel CPUs), with values that can differ for each module and manufacturer. Furthermore, the tracking is not continuous, because CodeCarbon measures the energy usage at a certain frequency. However, as remarked in [16], CodeCarbon is a good tool to track energy consumption and estimate carbon emissions, despite the negative points mentioned above. It is a software-based solution that makes emission monitoring simpler and more convenient without needing specialized equipments, such as wattmeters. A more detailed analysis has been recently given in [25] and [3], where CodeCarbon has been compared against other software tools, like CarbonTracker [1], resulting the most precise tool when accuracy of measures is evaluated against physical wattmeters. The main difference regards the estimation of the energy consumed by the RAM: CodeCarbon employs a fixed value of 3 Watts for every 8GB of DDR3 or DDR4 RAM, while CarbonTracker relies on the Intel RAPL for this purpose.

Although relevant limitations with respect to specialized equipments, CodeCarbon is the best software solution for energy monitoring of large computer systems [17] as well as Green AutoML solutions (i.e., it is the first tool suggested by AutoML Group on its website[6]).

Finally, since all the experiments in this paper have been performed on the on a single computer, the bias implied by CodeCarbon can be considered irrelevant because it should be constant over all the experiments.

4 Experiments and results

4.1 Preliminary considerations

As a preliminary investigation, a 15×15 grid of hyperparameter configurations, for each ML algorithm, has been evaluated separately on each data set. The collected observations led to conveniently grouping the pairs *ML algorithm - dataset* into three categories (analogously to Figures 1-3): high accuracy hyperparameters configurations associated to (a) low energy consumption, (b) high energy consumption, and (c) any energy consumption. For immediate consultation, the resulting grouping is reported in Table 3. It is evident that SVC results energy-efficient "by-design", with accurate hyperparameter configurations requiring small amounts of energy, for all the datasets. Moreover, the dataset *Phoneme* seems to be easy to classify, with the most accurate hyperparameter

[6] https://www.automl.org/green-automl/ (last access on 2025-05-15).

configurations, for all the ML algorithms, requiring the lowest amount of energy.

	Banknote auth.	Blood transf.	Phoneme	WDBC	Heloc
SVC	Energy ↓	Energy ↓	Energy ↓	Energy ↓	Energy ↓
MLP	Energy ↑	Energy ↑	Energy ↓	Energy ∼	Energy ∼
RF	Energy ∼	Energy ∼	Energy ↓	Energy ∼	Energy ∼

Table 3: Energy consumption associated to hyperparameters configurations with high accuracy: (↓) and (↑) denote, respectively, a low and high energy consumption, while (∼) is for no relation between accuracy and energy.

4.2 Experiment 1: HPO given a max number of queries.

In this experiment we compare results obtained on (a) the 15×15 grid search against (b) HPO via vanilla BO and (c) the proposed e^{2}HPO algorithm.

In this conference paper, the comparison against other cost-aware or multiple information source optimization methods is not considered. Indeed, multiple information source optimization already proved to be more effective and efficient than cost-aware optimization [4] as well as multi-fidelity approaches [7].

The termination criterion is a maximum number of evaluated hyperparameter configurations, that is equal to the size of the grid. For HPO and e^{2}HPO, this number is divided into 5 runs, each starting from a different set of 5 random hyperparameter configurations (shared by HPO and e^{2}HPO) and 20 sequential queries (i.e., 5 runs $\times$ (5+20 hyperparameter configurations) = 225). Independent runs are here considered as restart of HPO and e^{2}HPO, and the best model over the runs is chosen, to mitigate the effect of the random initialization that is well-known to affect performances of BO algorithms.

Table 4 shows the best 10FCV accuracy obtained on the 225 evaluated configurations and the overall energy consumption as measured by CodeCarbon. For e^{2}HPO, the accuracy refers to the best value on the full dataset only; the percentage of full dataset usage by e^{2}HPO is also reported. The most relevant results can be summarized as follows.

Result#1. As expected, HPO is more sample-efficient than grid search (except for MLP on Heloc and RF on WDBC), since based on BO: it offers equally or more accurate models than the grid search, given the same number of queries.

Result#2. In some cases e^{2}HPO provides a lower accuracy than HPO (and even grid search), specifically when the full dataset is less used (i.e., 51%-53%).

Result#3. In the face of a slightly higher accuracy, HPO entails an energy consumption close to that of the grid search (even larger in some cases), which is approximately twice as e^{2}HPO. Thus, sample-efficiency of HPO (Result#1) does not translate into energy-efficiency.

Result#4. The use of the full dataset, by e^{2}HPO, depends on both the ML algorithm and the dataset. Thus, e^{2}HPO is able to adapt its beheviour according to the target HPO task. When the full dataset usage increases the overall energy consumption obviously becomes closer – never higher – to that of HPO and grid search, but it can offer higher accuracies, as for RF on the first three datasets.

ML algorithm	Dataset	Grid Accuracy [%]	Grid Energy [Wh]	HPO Accuracy [%]	HPO Energy [Wh]	e^2HPO Accuracy [%]	e^2HPO energy [Wh]	e^2HPO full dataset usage [%]
SVC	Banknote auth.	100.00	0.375	100.00	0.325	100.00	0.214	54.22
	Blood transf.	77.41	0.217	**77.54**	0.238	77.27	0.153	51.11
	Phoneme	90.16	7.859	**90.30**	8.082	90.08	4.155	51.11
	WDBC	98.24	0.219	**98.42**	0.156	98.24	0.149	67.56
	Heloc	71.85	79.778	**71.90**	64.832	71.86	36.114	51.11
MLP	Banknote auth.	100.00	5.487	100.00	5.319	100.00	3.419	51.56
	Blood transf.	77.14	1.991	**78.61**	2.716	77.27	1.422	52.44
	Phoneme	83.25	20.961	83.25	23.364	83.23	12.327	52.00
	WDBC	97.89	2.429	97.89	2.377	97.89	1.999	70.67
	Heloc	**71.50**	48.648	71.35	40.893	71.35	23.863	51.11
RF	Banknote auth.	99.49	43.630	99.49	44.516	**99.56**	38.356	87.56
	Blood transf.	67.53	41.920	67.66	42.325	**67.93**	36.355	59.56
	Phoneme	91.64	28.101	91.62	29.228	**91.65**	25.881	86.67
	WDBC	**97.19**	25.036	97.02	25.387	96.84	23.460	84.89
	Heloc	71.60	68.221	**71.66**	67.807	71.63	43.962	68.44

Table 4: HPO given a max number of queries. 10FCV accuracy and overall energy consumption for: grid search, a simple HPO based on vanilla BO, and e^{2}HPO.

4.3 Experiment 2: HPO given a limit on the energy consumption.

Results from the previous experiment have been reorganized so that each run of HPO is considered as finished when its accumulated energy consumption reaches – without exceeding – that of e^{2}HPO on the same run. The new values of the 10FCV accuracy and energy consumption are reported in Table 5, where grid search is omitted because it makes no sense in this case. Moreover, the improvement/worsening of e^{2}HPO with respect to HPO is also reported.

Since the energy consumed by the two approaches is now comparable, the most relevant results just refer to 10FCV accuracy:

Results#5. According to considerations in Table 3, SVC is the only energy-efficient "by-design" ML algorithm among the three considered, with highly accurate hyperparameters configurations being also energy-efficient. When e^{2}HPO tries to reduce energy consumption by using the redux dataset it decreases the number of configurations evaluated on the full dataset – they are just 51%-67% – also reducing the chance to achieve a higher accuracy than HPO. This is mainly due to the fact that SVC training time increases as n^3 with n the number of data samples into the training set. As a consequence, e^{2}HPO is significantly "tempted" to use the redux dataset instead of the full one.

ML algorithm	Dataset	HPO Accuracy [%]	HPO Energy [Wh]	e^2HPO Accuracy [%]	e^2HPO energy [Wh]	full dataset usage [%]	Accuracy improvement [%]
SVC	Banknote auth.	100.00	0.210	100.00	0.214	54.22	0.00
	Blood transf.	**77.54**	0.151	77.27	0.153	51.11	-0.27
	Phoneme	**90.29**	4.036	90.08	4.155	51.11	-0.21
	WDBC	**98.42**	0.145	98.24	0.149	67.56	-0.18
	Heloc	**71.89**	34.812	71.86	36.114	51.11	-0.03
MLP	Banknote auth.	100.00	3.355	100.00	3.419	51.56	0.00
	Blood transf.	77.01	1.383	**77.27**	1.422	52.44	+0.26
	Phoneme	**83.25**	12.078	83.23	12.327	52.00	-0.02
	WDBC	97.89	1.9573	97.89	1.999	70.67	0.00
	Heloc	71.35	23.461	71.35	23.863	51.11	0.00
RF	Banknote auth.	99.49	37.891	**99.56**	38.356	87.56	+0.07
	Blood transf.	67.66	35.872	**67.93**	36.355	59.56	+0.27
	Phoneme	91.62	25.576	**91.65**	25.881	86.67	+0.03
	WDBC	**97.02**	23.230	96.84	23.460	84.89	-0.18
	Heloc	**71.66**	43.185	71.63	43.962	68.44	-0.03

Table 5: HPO given a limit on the overall energy consumption. 10FCV accuracy and energy for a simple HPO based on vanilla BO and e^{2}HPO.

Result#6. With respect to MLP, HPO and e^{2}HPO resulted comparable, except on the blood transfusion dataset. From Table 3, this is one of the cases where accuracy and energy consumption are inversely correlated. As HPO goes towards high accurate hyperparameter configurations it increases the accumulated energy consumption by quickly reaching the limit and, consequently, reducing the number of evaluated configurations (i.e., 25 on median and 19-30 as min-max, over the five independent runs). On the contrary, e^{2}HPO reduces its energy consumption by using the redux dataset while the number of configurations evaluated on the full dataset are sufficient to identify a more accurate MLP model than HPO.

Result#7. e^{2}HPO can provide more accurate RF models than HPO because accuracy and energy consumption resulted uncorrelated (as in Table 3).

In summary, a simple HPO using vanilla BO could result more effective and energy efficient than e^{2}HPO if the ML algorithm to be optimized is "natively" energy-efficient with respect to the hyperparameters cosnidered, just like SVC resulted in our experiments. Indeed, SVC models identified by e^{2}HPO are 0.14% less accurate, on average, than those identified by HPO (0.17% if tie is omitted).

Whenever a positive correlation between accuracy and energy consumption exists, e^{2}HPO is able to exploit it leading to equally or more accurate models than HPO, as empirically observed for MLP. Indeed, MLP models identified by e^{2}HPO are 0.05% more accurate, on average, than those identified by HPO (0.12% if ties are omitted).

Finally, e^{2}HPO was able to provide more accurate RF models than HPO because accuracy and energy consumption are uncorrelated, with an accuracy

improvement of 0.03% on average and up to 0.27% when using the redux dataset is reliable (i.e., RF on the blood transfusion dataset).

Figure 5 shows two examples clarifying the role of the cheap source's reliability. On the left, 10FCV accuracy with respect to MLP's hyperparameters on blood transfusion is depicted, separately for the ground-truth (i.e., full dataset) and the cheap source (i.e., redux dataset). The two surfaces are obtained as interpolation on $\mathcal{D}_f$ and $\mathcal{D}_r$, respectively. Due to their correlation, optimizing with respect to the cheap source leads to solutions close to the optimizer of the ground-truth – indeed, the full dataset was used for just 52.44% of the overall configurations. Instead, on the right it is a case where the cheap source is significantly different from the ground-truth, that is the experiment with the highest use of the full dataset, 87.56% (i.e., RF on banknote authentication).

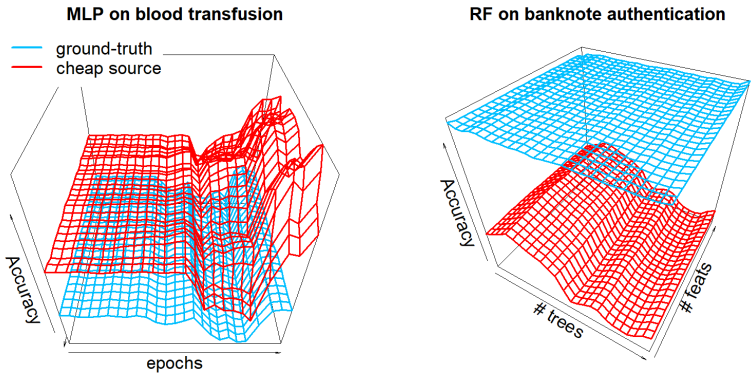

Fig. 5: 10FCV accuracy as interpolation on $\mathcal{D}_f$ (ground-truth, in blue) and $\mathcal{D}_r$ (cheap source, in red) for two representative cases. On the left, optimizing the cheap source leads to solutions close to the ground-truth's optimizer, $\mathbf{x}^*$, due too their correlation. On the right, the cheap source results not reliable because significantly different from the ground-truth.

4.4 Code and data availability, and technical details.

The proposed e^{2}HPO has been developed in R starting from the code provided in [5] and [7]. GP regression is performed via the R package `DiceKriging`, while the ML algorithms are from the Python library `scikit-learn` (aka `sklearn`) and run under a Python environment, as required by CodeCarbon. Basically, e^{2}HPO launches a Python script associated to the target ML algorithm by also specifying the dataset and the values for the ML algorithm's hyperparameters. The resulting 10FCV accuracy is returned to e^{2}HPO by the Python script, while

the energy consumption is shared between CodeCarbon and e^{2}HPO through an output comma-separated-value file (along with other computational metrics).

To guarantee reproducibility of the results and support knowledge sharing and research collaboration on the topic, the entire code and datasets are publicly available at the following a GitHub repository:
`https://github.com/ElenaSignori/E2HPO.git`

5 Conclusions

We have presented a novel energy efficient HPO algorithm which integrates into a unique framework recent advances from both cost-aware and multiple information source Bayesian optimization. We would like to remark that this intuition has recently emerged in the (optimization) community, such as in [9], but – at the authors's knowledge – our paper is the first one specifically addressing HPO of ML algorithms with such an integrated view.

Focusing on HPO of three common ML algorithms, and specifically on their core hyperparemeters, allowed us to draw important conclusions. Unsurprisingly, e^{2}HPO is more energy efficient than a simple vanilla BO based HPO, given the same number of queries, but there is no clear winner in terms of 10FCV accuracy.

However, when comparison is performed with respect to a threshold on the accumulated energy consumption, e^{2}HPO results, on average, better than the simple vanilla BO based method, but not for all the three ML algorithms considered. Indeed, some ML algorithms (i.e., SVC) proved to be energy-efficient "by-design" with respect to their core hyperparameters, so vanilla BO quickly converges close to the maximum 10FCV accuracy while cumulating low energy consumptions. On the contrary, e^{2}HPO tries to (further) reduce energy consumption by querying the redux dataset, from time to time, at the cost of a lower number of queries on the full dataset and, therefore, reducing the chance of reaching the maximum 10FCV accuracy (on the full dataset).

In conclusion, we are aware that the extent to which e^{2}HPO techniques contribute to energy efficiency remains ambiguous, particularly given the variability in energy consumption across different ML algorithms and dataset. Future works will consider a larger set of datasets and ML algorithms, especially large-scale models to evaluate scalability of the proposed approach. This will also allow us to improve e^{2}HPO so that it can better deal with settings where 10FCV accuracy and energy are negatively correlated and the gain in using the cheap information source exceeds that of querying the ground-truth.

Another important step we are going to address is to include the comparison against other cost-aware and multi-fidelity approaches as well as other research works combining the two methodologies analogously to the proposed e^{2}HPO.

Finally, we would like to remark (and discuss) that focusing on the core hyperparameters is definitely not a limitation. Indeed, other ML algorithm's *arguments* – usually treated as hyperparameters – refer to options related to the specific implementation of the learning procedure. Although the ML community

is active in proposing more efficient implementations for ML – that is another way to address sustainability in AI – the core hyperparameters of common ML algorithms are always the same. Consider for instance SVC and MLP, which a plethora of possible implementations have been provided for: although they became more efficient, their core hyperparameters never changed. Thus, if every implementation is considered as a ML algorithm *per-se*, the crucial information is if it is energy efficient by design or not.

References

1. Anthony, L.F.W., Kanding, B., Selvan, R.: Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. arXiv preprint arXiv:2007.03051 (2020)
2. Archetti, F., Candelieri, A.: Bayesian optimization and data science, vol. 849. Springer (2019)
3. Bouza, L., Bugeau, A., Lannelongue, L.: How to estimate carbon footprint when training deep learning models? a guide and review. Environmental Research Communications **5**(11), 115014 (2023)
4. Candelieri, A., Archetti, F.: Miso-wildcosts: Multi information source optimization with location dependent costs. arXiv preprint arXiv:2102.04951 (2021)
5. Candelieri, A., Archetti, F.: Sparsifying to optimize over multiple information sources: an augmented gaussian process based algorithm. Structural and Multidisciplinary Optimization **64**, 239–255 (2021)
6. Candelieri, A., Perego, R., Archetti, F.: Green machine learning via augmented gaussian processes and multi-information source optimization. Soft Computing **25**(19), 12591–12603 (2021)
7. Candelieri, A., Ponti, A., Archetti, F.: Fair and green hyperparameter optimization via multi-objective and multiple information source bayesian optimization. Machine Learning **113**(5), 2701–2731 (2024)
8. Dhar, P.: The carbon impact of artificial intelligence. Nat. Mach. Intell. **2**(8), 423–425 (2020)
9. Foumani, Z.Z., Shishehbor, M., Yousefpour, A., Bostanabad, R.: Multi-fidelity cost-aware bayesian optimization. Computer Methods in Applied Mechanics and Engineering **407**, 115937 (2023)
10. Garnett, R.: Bayesian optimization. Cambridge University Press (2023)
11. Jamieson, K., Talwalkar, A.: Non-stochastic best arm identification and hyperparameter optimization. In: Artificial intelligence and statistics. pp. 240–248. PMLR (2016)
12. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast bayesian optimization of machine learning hyperparameters on large datasets. In: Artificial intelligence and statistics. pp. 528–536. PMLR (2017)
13. Lee, E.H., Perrone, V., Archambeau, C., Seeger, M.: Cost-aware bayesian optimization. arXiv preprint arXiv:2003.10870 (2020)
14. Li, P., Yang, J., Islam, M.A., Ren, S.: Making ai less "thirsty": Uncovering and addressing the secret water footprint of ai models (2025), `https://arxiv.org/abs/2304.03271`
15. Paria, B., Neiswanger, W., Ghods, R., Schneider, J., Póczos, B.: Cost-aware bayesian optimization via information directed sampling. In: Adaptive Experimental Design and Active Learning in the Real World Workshop at ICML (2020)

16. Pop, M.: Measuring the energy consumption and carbon footprint of encrypted databases using CodeCarbon. B.S. thesis, University of Twente (2025)
17. Posthuma, M.: The energy consumption and carbon footprint of HTTPS. B.S. thesis, University of Twente (2025)
18. Sabbatella, A., Ponti, A., Candelieri, A., Archetti, F.: Bayesian optimization using simulation-based multiple information sources over combinatorial structures. Machine Learning and Knowledge Extraction **6**(4), 2232–2247 (2024)
19. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai. Communications of the ACM **63**(12), 54–63 (2020)
20. Shehabi, A., Hubbard, A., Newkirk, A., Lei, N., Siddik, M.A.B., Holecek, B., Koomey, J., Masanet, E., Sartor, D., et al.: 2024 united states data center energy usage report (2024)
21. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
22. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for modern deep learning research. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 13693–13696 (2020)
23. Terragni, S., Candelieri, A., Fersini, E.: The role of hyper-parameters in relational topic models: Prediction capabilities vs topic quality. Information Sciences **632**, 252–268 (2023)
24. Tornede, T., Tornede, A., Hanselle, J., Mohr, F., Wever, M., Hüllermeier, E.: Towards green automated machine learning: Status quo and future directions. Journal of Artificial Intelligence Research **77**, 427–457 (2023)
25. Ukarande, A., Basaklar, T., Cao, M., Ogras, U.: Pact: Accurate power analysis and carbon emission tracking for sustainability. In: Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design. pp. 1–6 (2024)
26. Verdecchia, R., Sallou, J., Cruz, L.: A systematic review of green ai. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **13**(4), e1507 (2023)
27. Xie, Q., Astudillo, R., Frazier, P.I., Scully, Z., Terenin, A.: Cost-aware bayesian optimization via the pandora's box gittins index. arXiv preprint arXiv:2406.20062 (2024)