Università degli Studi di
Milano-Bicocca

Scuola di Economia e Statistica

Corso di laurea in Scienze Statistiche ed Economiche

Evaluation of Machine Learning Algorithm Efficiency and Performance in the Context of Promoting Green AI

Relatore: Prof. Candelieri Antonio

Co-relatore: Prof. Pelagatti Matteo Maria

Tesi di laurea di:
Piccarreta Domenico
Matr. N. 892845

Anno Accademico 2022/2023

Abstract

Artificial Intelligence (AI) stands as a highly influential force today, achieving noteworthy successes in addressing challenges across diverse application domains. These achievements are primarily credited to the utilization of robust computational capabilities provided by High-Performance Computing (HPC) environments, albeit at the expense of significant energy consumption. Furthermore, the environmental impact of this energy usage is reflected in the emission of greenhouse gases, with CO2 being particularly prominent. The assessment and development of Machine Learning (ML) algorithms typically prioritize predictive performance, yet there remains a limited understanding of their computational demands and energy usage. Consequently, addressing these gaps becomes imperative for steering computing practices towards sustainability.

The primary goal of this study is to evaluate the computational efficacy and energy efficiency of ML algorithms, specifically Multilayer Perceptron (MLP) and eXtreme Gradient Boosting (XGBoost) algorithms, utilizing Bayesian optimization (BO). This endeavor seeks to identify solutions that contribute to the advancement of sustainable and efficient AI. The research delves into the intricate connections between computational performance and energy efficiency under various execution approaches. It endeavors to scrutinize how the subsequent hyperparameter configurations, guided by different methodologies, impact the overall performance, energy consumption and, consequently, the associated CO2 equivalent (CO2e) emissions during execution. Keywords: GreenAI. Bayesian Optimization. Energy Efficiency.

Contents

List of Figures

Acronyms

AI Artificial Intelligence. i, 11

AutoML Automated Machine Learning. 2, 11, 13, 91, 93

BO Bayesian Optimization. i, 20, 36, 51, 56, 57

CO2 Carbon Dioxide. 1, 11, 36

CO2e Carbon Dioxide Equivalent. i, 13, 26

CPU Central Processing Unit. 11, 14

DSP Differential Statistical Parity. 3, 58, 61

FPO Floating Point Operations. 12, 28

GPU Graphics Processing Unit. 14, 25

HPC High Performance Computing. i, 13

HV Hyper Volume. 57, 82, 85, 89

MCCV Monte-Carlo Cross-Validation. 19

ML Machine Learning. i, 1, 11, 54

MLP MultyLayer Perceptron. i, 67, 69, 78, 81

MWh Megawatt Hour. 23

NAS Neural Architecture Search. 8, 15, 23, 25

NLP Natural Language Processing. 1, 6, 7, 30

SOTA State Of The Art. 15, 23

TPU Tensor Processing UniT. 32

XGBoost eXtreme Gradient Boosting. i, 49, 67, 69

Chapter 1

Introduction

1.1 Problem Formulation

Artificial Intelligence (AI) emerges as a highly transformative force in the contemporary landscape, yielding significant advancements in addressing diverse challenges across domains such as image and voice recognition, healthcare, cybersecurity, autonomous vehicles, personal assistants, and precision product recommendations. The success of AI, often facilitated by the integration of Machine Learning (ML) algorithms, is primarily attributed to the synergies between AI and High-Performance Computing (HPC). To put it simply, the current abundance of extensive data, combined with robust computational capabilities, is fostering the development and training of state-of-the-art ML algorithms. On the global stage, the United Nations General Assembly has outlined 17 Sustainable Development Goals (SDGs) to be achieved by 2030, spanning three major areas: Society, Economy, and Environment (NATIONS, 2021). Within the domain of AI applications, there is increasing evidence that the deployment of AI algorithms could positively contribute to attaining these goals, particularly within the Environmental category, addressing climate action, life below water, and life on land (Vinuesa et al., 2020). However, research (Vinuesa et al., 2020; UNESCO, 2021) suggests that efforts to combat climate change and environmental challenges may face obstacles due to the substantial energy requirements of AI applications, especially when utilizing non-carbon neutral sources. In such cases, AI algorithms prove to be costly both in terms of computational power (and consequently energy consumption) and environmentally, as reflected in the carbon footprint associated with computational environments. Furthermore, a recent investigation has uncovered that the environmental repercussions of developing a Natural Language Processing (NLP) model using Deep Learning can result in emitting as much carbon dioxide (CO2) as five cars over their entire lifespan (Strubell et al.,2019). Consequently, the ongoing trajectory of AI advancement is rapidly

becoming economically, technically, and environmentally unsustainable (Thompson et al., 2020). Taking these factors into account raises significant concerns about establishing ecological viability in AI development, given that training these algorithms involves the consumption of electrical energy and, consequently, the emission of carbon dioxide (CO2), a primary contributor to the greenhouse effect.

1.2 Contributions

The primary research focus of this study is to propose approaches toward Green AI, defined as "AI research that is more environmentally friendly and inclusive" (Schwartz et al., 2020). As mentioned earlier, the training of ML algorithms entails energy consumption and, consequently, CO2 emissions. Therefore, the ecological impact of utilizing AI algorithms cannot be disregarded. Energy efficiency has been a longstanding focus of research in computing, leading to emerging research areas such as Green IT or Green Computing. Green Computing, for instance, can center on reducing the energy consumption of data centers or minimizing the energy used to cool these environments (Mair et al., 2015). Within these studies, some are directed at reducing the energy consumed by data centers by employing AI to forecast runtime and enhance work scheduling (Klôh et al., 2019). In this domain, numerous results have been obtained, with many contributing to the advancement and enhancement of computational architectures. Nonetheless, there is still a scarcity of research in Green AI, specifically in reducing the energy consumption of AI algorithms and their environmental impact.

In reality, despite the significance of AI algorithms, limited knowledge exists regarding their computational requirements and energy consumption. It is crucial to ascertain whether these algorithms exhibit satisfactory computational performance and whether they can be optimized for improved energy efficiency. Much of the research in the AI field primarily concentrates on enhancing the accuracy of algorithms (Martin et al., 2019). However, despite the evident benefits of enhancing the accuracy of AI models, focusing solely on this metric neglects the economic and environmental costs associated with achieving the desired accuracy (Schwartz et al., 2020), which are essential considerations in the context of Green AI. Therefore it's fundamental to discover a proxy that measures consumed energy and can be readily translated into CO2 consumption. Assessing the eco-friendliness of an AutoML (Automated Machine Learning) strategy is typically the initial phase in moving towards a more environmentally conscious approach. Nevertheless, gauging sustainability using a singular metric that serves as a benchmark for comparing various approaches poses a complex challenge. Once this aspect is delineated, the comparison among three types of approaches begins, emphasizing which of them at-

tains satisfactory results in both accuracy and Differential Statistical Parity (DSP), in this context, the discussion will revolve around Pareto efficiency, as well as the amount of CO2 consumed.

1.3 Outline

In the context of recent advancements in Green AI, this paper introduces the concept of Green AutoML, aiming to enhance the overall environmental friendliness of the entire AutoML process. The structure of this dissertation is as follows: Chapter 2 delves into the theoretical foundations that underpin this work, exploring key concepts of ML and AutoML. It emphasizes the need to transition from "red AI," a term coined by Schwartz in mid-2019, to "green AI." The achievements contributing to green AI are highlighted. Consequently, the chapter underscores the importance of quantifying the environmental impact of an AutoML tool. Moving on to Chapter 3, it introduces three algorithms after clarifying the concepts of multi-fidelity, multi-information, and multi-objective, along with their respective performance metrics. A novel Fair and Green Hyperparameter Optimization algorithm, named FanG-HPO, is presented, employing both multi-objective and multiple information source Bayesian Optimizations to simultaneously address Fairness-aware and Green AutoML.The computational assessment of FanG-HPO is compared with two other state-of-the-art BO suites facilitating both multi-objective and energy-efficient Hyperparameter Optimization (HPO): autogluon-FairBO (Schmucker et al., 2020), which treats HPO as a bi-objective optimization task, and BoTorch-MOMF (Multi-Objective and Multi-Fidelity) (Irshad et al., 2021), implementing a generic multi-objective and multi-fidelity BO framework. Chapter 4 outlines the experimental design and methodology, encompassing datasets, architecture, tools, and algorithms used, concluding with a discussion of the experiment results. Finally, Chapter 5 provides the concluding remarks, including achievements, limitations, and future perspectives.

Chapter 2

Green AI

Since 2012, the domain of artificial intelligence (AI) has witnessed notable advancements across various capabilities such as object recognition, game playing, speech recognition, and machine translation. Much of this progress has been propelled by increasingly extensive and computationally demanding deep learning models.

Deep learning, a technique within machine learning, empowers computers to learn through examples, mirroring the natural learning process of humans. This technology plays a crucial role in applications like driverless cars, enabling them to identify road signs and distinguish between pedestrians and lampposts. It also underlies voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. The attention garnered by deep learning is well-deserved, given its ability to achieve unprecedented results.

In deep learning, computer models acquire the capability to perform classification tasks directly from various inputs such as images, text, or sound. These models can achieve accuracy levels that sometimes surpass human performance. The training process involves using large sets of labeled data and neural network architectures with multiple layers that learn features directly from the data without manual feature extraction.

However, in recent times, efforts to enhance accuracy have shifted towards optimizing the search for model hyperparameters. This optimization is performed automatically and will be called AutoML. It involves minimizing or maximizing a black-box function, often accomplished through Bayesian optimization, as testing all possible combinations of hyperparameters has become impractical due to the increasing complexity of neural network structures. Specifically, neural network hyperparameters act as settings chosen before teaching the network a task, influencing aspects like the number of layers, learning speed, and internal value adjustments. Selecting the right hyperparameters is crucial for effective learning and accurate task-solving, akin to adjusting the knobs on a machine for optimal performance in a specific job. All this requires co2 consumption. To comprehensively account for

the carbon emissions generated by a single published AutoML paper, one must examine the entire production chain. This involves scrutinizing the data generation and storage, the computational resources, and memory utilization throughout the development of the respective AutoML system, along with the final benchmarking phase. Typically, the majority of carbon emissions stem from the AutoML process itself, specifically the evaluations of ML pipelines and the storage of intermediate results during the search process. Therefore, our focus throughout the dissertation primarily centers on this crucial aspect.

Inspired by the recent endeavors in Green AI, there is an intention to extend these ideas and concerns to the realm of AutoML, creating a paradigm termed Green AutoML (Tornede et al., 2023). The cornerstone of this paradigm lies in quantifying the environmental impact of AutoML approaches. To reduce the environmental footprint of research on AutoML, the community may consider certain measures. Both the design and benchmarking of AutoML systems play pivotal roles in enhancing the environmental footprint. Additionally, transparency regarding this footprint provides valuable supplementary information about an AutoML approach. Lastly, establishing appropriate research incentives could steer AutoML research towards a more sustainable trajectory.

2.1 AI Dualism

AI appears to have a dual role ahead. On one side, it holds the potential to contribute to mitigating the impacts of the climate crisis, particularly through applications like smart grid design, water utilities, the development of low-emission infrastructure, and the modeling of climate change predictions. However, on the flip side, AI itself emerges as a significant source of carbon emissions. This awareness gained broader attention in the latter part of 2019 when researchers at the University of Massachusetts Amherst conducted an analysis of various online natural language processing (NLP) training models. Their goal was to estimate the energy cost, measured in kilowatts, required to train these models. When converting this energy consumption into approximate carbon emissions and electricity costs, the researchers concluded that training a single large language model equated to around 300,000 kg of carbon dioxide emissions. To provide perspective, this is comparable to the carbon footprint of approximately 125 round-trip flights between New York and Beijing, making it a quantification that can be easily visualized by the general public.

However, the carbon cost associated with training extensive machine learning models, as highlighted in the UMass study, is only one aspect of the issue. To gain a comprehensive understanding, closer attention must be paid to the broader carbon

impact of the infrastructure surrounding the deployment of AI by major tech companies. In the past year, there has been a growing movement among tech workers urging their employers to acknowledge their role in the climate crisis. Thousands participated in the global climate strike in September 2019 to draw attention to big tech's collaboration with fossil fuel companies and its involvement in the repression of climate refugees and frontline communities. Under the banner of the Tech Workers Coalition, employees from companies such as Amazon, Google, Microsoft, Facebook, and Twitter rallied to demand commitments from their employers, including a pledge to achieve zero emissions by 2030, refraining from contracts with fossil fuel companies, discontinuing funding for climate change deniers, and ceasing the exploitation of climate refugees and frontline communities.

2.2 State of the art

It's time to switch. This is the warning conveyed in the iconic paper by Strubell et al., which examines the carbon impact of training state-of-the-art AI models. The findings suggest the need to decrease the carbon footprint associated with developing and running AI models. In 2018, a study led by Dario Amodei and Danny Hernandez from OpenAI in California, an organization committed to ensuring that artificial general intelligence benefits humanity, uncovered a startling trend. The computational power used in various large AI training models had been doubling every 3.4 months since 2012, a significant departure from Moore's Law, which predicts a doubling interval of 18 months. This surge accounted for a remarkable 300,000× increase. The trend was even more pronounced in (NLP) word-embedding approaches, including ELMo34, BERT, openGPT-2, XLNet, Megatron-LM, T5, and GPT-3.4, mirroring the advancements in the AI industry.

While algorithmic innovation and data, crucial factors in AI growth, are challenging to quantify, compute provides a tangible metric. However, in their blog post, the authors noted that the nebulousness surrounding the exact amounts of compute can serve as a fig leaf, obscuring the deficiencies of current algorithms. In a 2019 position paper, Roy Schwartz and collaborators labeled this trend as 'red AI,' signifying the pursuit of stronger results through massive compute. Achieving linear gains in performance necessitates exponentially larger models, achieved by increasing training data or the number of experiments, leading to elevated computational costs and carbon emissions.

To illustrate the prevalence of red AI, Schwartz et al. analyzed over 60 papers from top conferences, finding that the majority prioritized accuracy over efficiency.

Three factors contributed to red AI: the cost of executing a model on a single example, the size of the training dataset controlling model executions, and the number

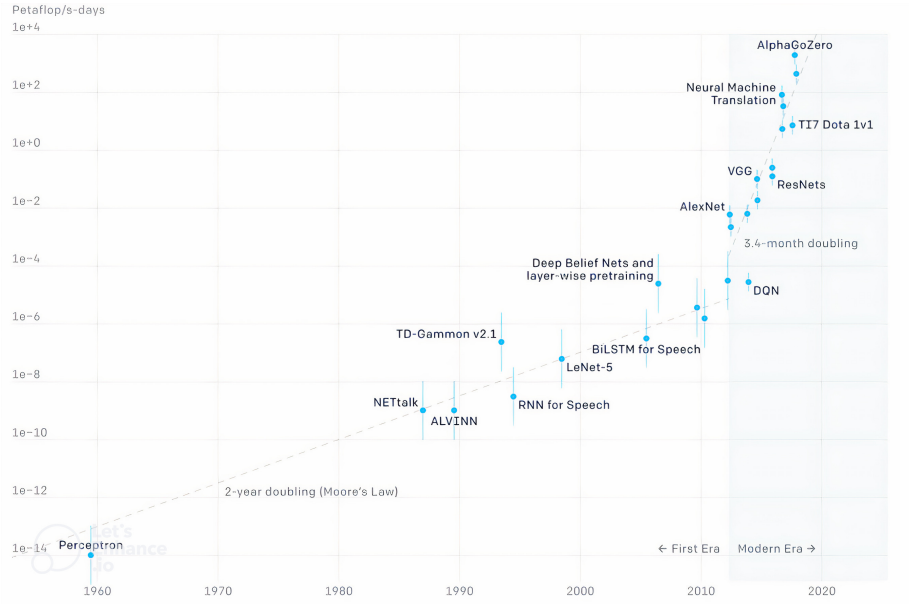

Figure 2.1: Total amount of compute used for final training runs of selected AI systems. The amount of compute used to train deep learning models has increased 300,000x in six years. Source: (Amodei and Hernandez, 2018)

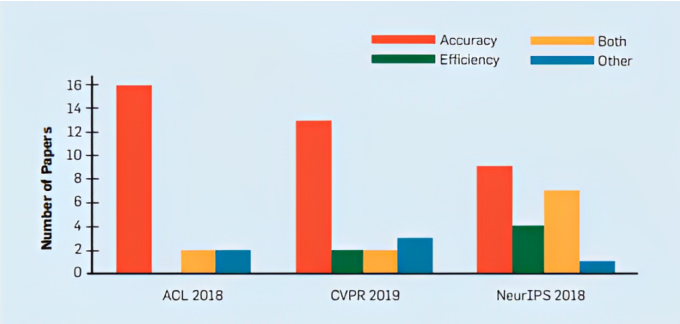

Figure 2.2: AI papers tend to target accuracy rather than efficiency. The figure shows the proportion of papers that target accuracy, efficiency, both or other from a random sample of 60 papers from top AI conferences. Source: (R.Schwartz, 2020)

of hyperparameter experiments affecting model training frequency. The total cost of producing machine learning results increases linearly with each of these factors. The growing inclination toward neural architecture search (NAS) and automated hyperparameter optimization, which heavily rely on compute, significantly contributes to red AI. Schwartz and colleagues advocated for '**Green AI**' defined as AI research yielding novel results without increasing or ideally reducing computational costs, in contrast to red AI. However, they acknowledged the potential value of red AI in pushing the boundaries of model and dataset sizes, as well as the hyperparameter search space, even if substantial resources are required. Google AI's Justin Burr highlighted that training more efficient models, such as those found through NAS, can ultimately save more energy over time, outperforming older hand-tuned versions

in energy efficiency within a week.

AI scientist Richard Sutton, often referred to as the "father of reinforcement learning" penned a blog post in early 2019 titled "The bitter lesson" asserting that AI methods leveraging computation surpass those relying on human knowledge in terms of effectiveness and accuracy. This perspective has caused division within the AI industry. Nevertheless, it is undeniable that the escalating reliance on increasing amounts of compute and data necessitates a corresponding rise in power consumption and infrastructure, leading to a growing carbon footprint.

To comprehensively understand AI's carbon impact, solely scrutinizing the compute costs associated with training large models is insufficient. The reluctance of tech companies to share data, coupled with a lack of incentives to do so, complicates efforts to quantify emissions. Roel Dobbe from the AI Now Institute at New York University emphasizes the peculiar complexity arising from the rapid globalization and consolidation of the computer industry into a few dominant players. This trend challenges societies' ability to maintain control over critical infrastructure. Dobbe underscores the importance of regional data centers for global compute efficiency but notes that, dominated by three American players, these companies extend their influence worldwide. This situation necessitates the implementation of checks and balances, including regulations, to retain local agency over such infrastructures, as these companies invest significantly in lobbying against regulations that may curtail their power.

The concentration of power among a few industry leaders prompts questions about the impact on compute prices and the strategies of other players unconstrained by stringent energy requirements. Companies, however, remain hesitant to disclose their energy mix. Greenpeace's Clicking Clean report from 2017 highlights that despite commitments to a 100% renewable future, many companies are stuck in a status quo, with Amazon's emissions increasing by 15% despite net-zero-by-2040 pledges. The report also notes a dual trend of renewable energy promotion in some markets and a simultaneous push for fossil-fuel-based energy in others, exemplified by Virginia, USA, where only 1% of electricity comes from renewable sources. The report further draws attention to the connection between Big Data and Big Oil, indicating that companies like Amazon, Google, Microsoft, and Royal Dutch Shell market AI solutions to fossil fuel extraction and usage companies. Estimating the carbon footprint of AI technologies, according to Dobbe, should be straightforward given the hardware running and known algorithm operation counts. He draws parallels to the aerospace industry, where emissions are easily tracked due to standards and reports. Achieving transparency in the computing industry, including data centers and network infrastructure, primarily relies on political will and consumer awareness, Dobbe suggests.

Alexandra Luccioni advocates for more tax incentives for cloud providers to establish data centers in locations with hydro or solar energy, citing Quebec's low-carbon grid predominantly powered by hydro. She points out the potential benefits, such as utilizing the heat generated by computing centers to warm homes, emphasizing the positive impact this could have compared to locations like Texas, where the grid relies heavily on coal. The imperative need to transition to green AI is emphasized. The Copenhagen Centre on Energy Efficiency, a collaboration between the United Nations Environment Programme and the Technical University of Denmark, focuses on researching and advising on climate, energy, and sustainable development. Gabriela Prata Dias, the center's head, and Xiao Wang, a program officer, underscore that environmental sustainability should be a fundamental principle in the responsible development and application of AI. They emphasize that AI is not just a tool but also a resource demander, and the benefits of its use should outweigh its drawbacks.

To promote cleaner AI practices, they propose several steps. First, they assert that the definition of green AI should be actionable for all relevant stakeholders in the industry, rather than remaining an abstract concept for technical experts. They advocate for the crucial role of standards in driving the adoption of green AI. Developing environmental standards is suggested to ensure the mitigation of environmental impacts, with the introduction of green AI certifications to facilitate the industry process in promoting environmentally friendly AI development. For organizations and companies utilizing AI technologies, they recommend practical industry frameworks and guidelines supporting green procurement of AI technologies to encourage the adoption of environmentally friendly practices.

Moreover, they stress the importance of governments considering the long-term impacts when establishing regulatory frameworks and legislation. These measures should legally address transparency and sustainability in AI development.

Deepika Sandeep, an AI scientist heading the AI and ML program at Bharat Light & Power (BLP), a clean energy generation company in Bengaluru, advocates for judicious use of deep learning. Not every problem, she argues, requires a machine learning-based solution. Acknowledging that training is a major contributor to computational power consumption and carbon footprint, BLP minimizes training cycles. Once deployed in production, there is no further training, with retraining occurring only once every three or six months.

Sandeep cautions against employing solutions based on deep neural networks and deep learning architectures for simple problems that could be addressed by other, less compute-intensive AI methods, emphasizing the potential negative environmental impact of such practices. The goal of all these interventions is to encourage the AI community to recognize the value of work by researchers that take a different path,

optimizing efficiency rather than accuracy.

2.3 Carbon Footprint or Efficiency

In the paragraphs to come it is analyzed more specifically at how to reduce CO2 consumption, but to achieve this, there is a need to obtain a way to quantify the CO2 consumption of a model. According to Crawford and Joler's essay, there is a lack of specific details about the costs associated with large-scale AI systems in the public perception, leading laypeople to potentially underestimate the complexity of building a ML based system. The absence of a standardized measurement further contributes to this mystery. Primarily, it is crucial to distinguish between the efficiency of a method and the environmental footprint of a particular experiment employing that method. Furthermore, while any logical metric for efficiency should be applicable regardless of the hardware used (aka hardware-independent), gauging the environmental impact of a particular experiment on a specific machine necessitates consideration of the hardware in use (aka hardware-dependent). When assessing various criteria for their ability to gauge efficiency, the performance curve of the approach is consistently examined, as detailed in Section 2.5. This curve illustrates how the metric of interest (e.g., runtime) correlates with the achieved performance (e.g., accuracy) after a certain metric budget has been utilized. For instance, if runtime were a suitable measure, a tool achieving an accuracy of 0.8 within 30 seconds might be considered more efficient than one requiring 60 seconds. It is crucial that metrics for efficiency assessment remain independent of the hardware used, similar to the O notation in theoretical computer science, albeit considering constants as relevant in this context. In practical terms, however, approaches are often executed on diverse hardware with varying characteristics, such as energy consumption. Additionally, experiments may vary in scope, exemplified by two studies employing different approaches one evaluating on two datasets with an old laptop and a high energy consumption CPU, and the other assessing 100 datasets with more efficient hardware from a computing center. Despite the potential for the latter to be a more efficient approach, its larger energy consumption likely results in a greater environmental footprint than the former. Therefore, the evaluation of Green AutoML research cannot be based solely on the efficiency of an approach or the environmental impact of a specific set of experiments. Rather, a comprehensive assessment should consider both aspects simultaneously. In the subsequent discussion, various metrics are explored in terms of their appropriateness for measuring both energy efficiency and environmental impact, as summarized in Table 2.1.

	Properties			Quantification	
Measure	Hardware Independence	Measurability	Human Interpretability	Efficiency	Environmental Footprint
Runtime	×	✓	×	×	○
CPU/GPU Hours	×	✓	×	×	○
Floating Point Operations	×	✓	×	×	×
Energy Consumption	×	×	✓	×	✓
CO_2e	×	×	✓	×	✓

Table 2.1: Summary of discussed measures to quantify efficiency and environmental impact, together with their properties (fulfilled: X; partially fulfilled: ○; not fulfilled: ×).

Runtime

While runtime does not take into account factors like memory consumption, it exhibits a strong correlation with the energy consumption of the corresponding experiment, often following a linear relationship between runtime and energy consumption of the components. However, interpreting runtime alone can be challenging for a human in terms of understanding the scale of the environmental footprint, as it lacks a direct comparison to commonly used metrics, such as the footprint of a flight.

Nevertheless, when supplemented with additional information like the energy consumption of the utilized hardware (per time unit) and the composition of the energy mix, runtime becomes useful for estimating the overall CO2 footprint of the experiment at the specific location and time of its execution. Additionally, compared to other metrics discussed later, measuring runtime is relatively straightforward on most hardware.

In summary, while runtime proves to be an inadequate measure of efficiency due to its lack of hardware independence, it serves as a practical proxy for assessing environmental impact.

Floating Point Operations

Contrary to intuition, floating-point operations (FPO) represent a hardware-dependent metric, making them unsuitable for effectively quantifying efficiency. This dependence on hardware arises from the compiler optimization during the code compilation process. Depending on the level of optimization and the specific hardware targeted for optimization, the count of FPOs can vary. Additionally, using FPOs as a proxy for environmental footprint is problematic, as it, like previous metrics,

overlooks other factors. it doesn't consider memory usage, which can often lead to additional energy and monetary costs. The energy consumption of a model is not only influenced by the amount of work, but also from other factors such as the communication between the different components, which is not captured by FPO. Nevertheless, FPOs are frequently straightforward to measure, as many CPU/GPUs feature accessible counters for this purpose. However, interpreting this measure and comparing it to others remains challenging from a human perspective.

Electricity usage

Energy consumption, much like runtime, is not independent of hardware and heavily relies on the energy efficiency of the hardware in use. Although it may not serve as an ideal metric for gauging the efficiency of an approach, it proves to be an effective measure for assessing the environmental impact of a specific experiment on particular hardware. This is because energy, apart from the hardware itself, stands as a crucial external resource for conducting AutoML experiments. Additionally, analyzing the amount of consumed energy allows for a reasonable estimation of the actual CO2 emissions resulting from the experiment, taking into account the specific location and time of execution, given the availability of sufficient supplementary information such as the energy mix.

Moreover, individuals can easily relate to and comprehend energy consumption, often having awareness of their own energy usage at home. Unfortunately, practical challenges arise when attempting to measure energy consumption, especially in scenarios involving the use of HPC systems for experimental evaluations. The complexity increases when measuring the energy consumption across multiple nodes of a cluster, which might be shared among different users. For an in-depth exploration of available estimation methods and corresponding software, one can refer to García-Martín et al..

Carbon emission

CO2e represents an excellent and arguably the most direct metric for assessing the environmental impact of an experiment, provided the physical location and execution time are specified. However, the measurability of CO2e is even more challenging than that of energy consumption because it is not directly measurable. Instead, it requires computation based on energy consumption and additional information about the energy mix. Obtaining such information in practice is often difficult, and the energy mix may even fluctuate due to external factors like weather conditions, particularly if it includes renewable energy components.

Furthermore, while the energy consumption of an experiment is largely indepen-

dent of the time and location of execution, the CO2e footprint can vary significantly based on these factors. Hence, the energy mix is a crucial component of the environmental footprint. For instance, conducting an experiment at a compute center exclusively powered by renewable energy yields no direct CO2e emissions. In contrast, running the same experiment on a unit powered by coal-generated energy results in considerably higher CO2e emissions. Patterson et al. demonstrate that factors such as the choice of the (deep learning) model, the compute center, and the compute unit can influence the carbon footprint of a study by a factor of 1000. Additionally, simply selecting the appropriate compute center location can lead to a factor of up to 10 in terms of impact.

CPU/GPU hours

Similarly, the measurement of CPU/GPU hours is both practical and facilitates the quantification of environmental impact. Unfortunately, the term CPU/GPU time is often used ambiguously, as one can measure either wall clock CPU/GPU time or true CPU/GPU time, leading to different interpretations in quantifying environmental impact. When wall clock time is used, the impact of other operations like memory access is implicitly considered, including situations where the computer begins swapping due to overloaded main memory. Conversely, when real CPU/GPU time is used, these operations are partially disregarded.

Like in the case of runtime, using CPU/GPU hours makes it challenging for a human to assess the impact in comparison to other familiar benchmarks from daily life. Furthermore, counting CPU/GPU hours serves as a limited proxy for efficiency, given its dependence on hardware. Nonetheless, currently, it stands as one of the more practical proxies due to its ease of measurement and relatively straightforward conversion into CO2e, provided that the CPU/GPU consistently consumes a known amount of energy and the energy mix is known.

To gauge the environmental impact of a specific experiment, the most convenient proxy is to employ wall clock time-based CPU/GPU hours. This metric is typically straightforward to obtain without delving into extensive investigations concerning the energy mix and energy consumption, which may be impractical in the case of High-Performance Computing (HPC). However, if such additional information is readily available, it is possible to estimate the environmental footprint by multiplying the CPU/GPU hours by the energy consumption per unit and the CO2e associated with the energy mix.

It's important to note that even with a green energy mix, computational processes are not entirely devoid of environmental impact. The utilization of resources, such as tool wear, inherently contributes to a footprint. Therefore, minimizing CPU/GPU hours for any energy mix or eliminating unnecessary computational time

can effectively reduce the overall environmental footprint of the research.

2.4 An efficient way of evaluating the model

The assessment of AutoML pipelines is likely the primary contributor to the adverse environmental effects stemming directly from AutoML research. While evaluating these systems incurs costs, they align with the overall research process. Notably, the appraisal of candidate solutions tends to demand significant resources. Consequently, it is posited that the development of approaches that inherently factor in their ecological impact represents a pivotal concept for promoting environmentally conscious AutoML. Designing such pipelines is a time and resource-intensive task due to the vast array of conceivable solutions, each addressing the same problem but varying in terms of performance (yielding models with greater or lesser predictive accuracy). Such complexity is due because a machine learning (ML) pipeline is an amalgamation of appropriately configured ML algorithms within an overarching (aka software) solution that can be applied to a specific learning task, typically characterized by a dataset on which a predictive model is to be trained. Consequently, in the pursuit of finding the optimal pipeline in terms of performance, candidates are evaluated on the available dataset to further tailor and enhance their composition and configuration. With the aim of automating this process, AutoML develops methods to systematically explore the ML pipeline space, usually with a predefined timeout after which the most promising candidate is used to train the final model. Interest in the field of AutoML has rapidly grown, expanding in scope, although predictive performance remains the primary measure of interest.

In the realm of AutoML, theoretical results are challenging to obtain as the problem is complex and not easily amenable to theoretical analysis. As a result, most research contributions are empirical in nature: the proposal of a new technique or approach is accompanied by extensive experimental research demonstrating the benefits of the proposed techniques. Since the focus is on techniques that perform well in general across a wide range of problems, a large number of datasets are required for evaluation. Furthermore, as there is no single AutoML system that represents the state of the art (SOTA) and dominates others, multiple competitors need evaluation. To achieve this, typically each competitor is re-evaluated on the same datasets and ideally under the same conditions, such as the same search space and hardware constraints. Coupled with the long evaluation times of a single candidate solution, which can extend to several hours or even days, as in the case of Neural Architecture Search (NAS), all this translates into a field that demands significant resources and therefore generates immense carbon emissions, many of which could be mitigated, as discussed later. Note that carbon emissions or CO2e refers to the CO2 equivalents,

i.e., the amount of CO2 with the same global warming potential as the gas actually emitted.

To fully account for the carbon emissions produced through a single published AutoML paper, one must consider the entire production chain, starting from data generation and storage, the computational effort and memory required during the development of the corresponding AutoML system, and also the final benchmark. Typically, the AutoML process, encompassing the evaluations of ML pipelines and the storage of intermediate results of the search process, is the main contributor to carbon emissions. This aspect is the primary focus throughout the paper.

However, it is important to emphasize that papers with a significant environmental footprint do not necessarily have to be condemned (this will be discussed in more detail later). What is crucial, instead, is to carefully balance the costs against the benefits. For example, a paper that invests many resources in creating a benchmark, allowing other researchers to save resources in their papers, provides a much greater benefit than a paper that invests the same resources in evaluating a method that only yields a 0.001 % improvement over SOTA methods.

In this section, the exploration revolves around the most widely adopted approach among researchers to diminish CO2 consumption during pipeline evaluation. Specifically, it involves striking a balance between identifying effective ML pipelines and minimizing the energy consumption associated with the research process itself.

2.4.1 Warmstarting

Warmstarting is a mechanism that incorporates knowledge acquired from previous executions into the current execution, ensuring that the optimization process doesn't commence entirely from scratch, devoid of any prior information. The fundamental concept behind warmstarting is to identify promising candidates early in the process, enabling shorter timeouts with confidence in their proximity to optimal solutions. The exploration of warmstarting techniques has gained significant momentum, particularly in response to AutoML challenges with tight time constraints (refer to Section 5).

Typically, warmstarting relies on some form of meta-learning (J. Vanschoren, 2018), drawing insights from past experiences, which may or may not be linked to specific dataset properties, to provide recommendations for the current dataset.

A straightforward instance of warmstarting involves following a constant initial sequence. For instance, ML-Plan employs a fixed order of algorithms determined by the overall average performance across previous datasets. Similarly, auto-sklearn 2.0 (M. Feurer, 2007) explores a static portfolio designed to cover various use cases.

Alternatively, recommendations can be influenced by dataset properties. Al-

though diverse forms of warmstarting have been proposed in hyperparameter optimization, the pioneering AutoML approach that employs this form of warmstarting is auto-sklearn. In this case, dataset meta-features (such as the number of instances, features, skewedness, etc.) are utilized to match them with datasets encountered in the past. Subsequently, priority is given to pipelines that performed exceptionally well on those comparable datasets.

Over time, warmstarting has evolved into a standard technique adopted by many AutoML systems.

2.4.2 Zero-Shot AutoML

A striking manifestation of warmstarting lies in the concept of zero-shot AutoML, drawing inspiration from the principles of zero-shot learning (Y. Xian et al., 2017). In this paradigm, the warmstarting mechanism proposes a single pipeline candidate, which the system adopts without undergoing any form of evaluation.

Notably, zero-shot AutoML was among the pioneering approaches in this domain. For example, the Meta-Miner approach relied on recommendations derived from an ontology encompassing dataset and algorithm properties (P. Nguyen, 2011). Although the term "zero-shot AutoML" wasn't explicitly coined at that time, the recommendation process didn't involve any evaluation.

Several recent proposals have introduced diverse variations of zero-shot AutoML (I. Drori et al., 2019; N. Singh et al., 2021; J. Mellor et al., 2021; M. Lin et al., 2021). These approaches leverage transfer (L. Torrey et al., 2010) and meta-learning (J. Vanschoren, 2018) during an offline phase preceding their usage. In this phase, the approaches either leverage existing performance data of ML pipelines on various datasets or autonomously generate such data. The objective is to learn a mapping from datasets to ML pipelines, facilitating nearly instantaneous querying during actual usage. To enable this learning, datasets are typically represented in terms of features, known as meta-features (A. Rivolli et al., 2018). For example, in a new dataset (I. Drori et al., 2019), meta-features are computed based on a learned embedding of the dataset description. The closest dataset from the offline training phase, identified in terms of a measure defined on the meta-feature space, dictates the pipeline that performed best on that dataset.

Certainly, these methods involve energy-consuming computations, but they strategically shift the computational burden away from the actual search phase to a preceding offline phase, offering dual advantages. Firstly, this enables scheduling the offline phase at times when renewable energy is abundantly available. Simultaneously, the system remains accessible at any time to propose a pipeline for a given dataset. Secondly, significant energy savings can be realized if the AutoML system

is used extensively, as the initial training phase might require less energy than using a standard AutoML system during the search.

In addition to zero-shot AutoML, less extreme variants such as one-shot (B. Lake et al., 2011) or few-shot (Y. Wang et al., 2020) approaches can be explored. The ongoing research in this field has been invigorated by recent challenges and is actively pushing the boundaries of innovation.

2.4.3 Avoiding Evaluations with a Timeout

In the realm of AutoML systems that operate on a trial-and-error paradigm where candidate ML pipelines undergo training and validation on provided data, it's customary to establish a predefined timeout for evaluating these candidates. The rationale behind these timeouts lies in the potential time-intensive nature of evaluating certain candidates, posing a risk to exploration and, in extreme cases, causing the optimization process to grind to a halt. While imposing time limits on evaluations and prematurely halting the assessment of candidate pipelines when exceeding the specified budget is a technical necessity, it significantly compromises the overall efficiency of such AutoML systems. Mohr et al. aptly point out that a substantial chunk of computational resources is channeled towards evaluating pipelines that will eventually be cut short due to a timeout, yielding minimal information for the ongoing search process. Frequently, this results in a complete absence of information, leading to a literal waste of CPU/GPU time.

A natural aspiration is to curtail the occurrences of such premature terminations. To tackle this challenge, (Mohr et al., 2021) propose enhancing AutoML systems that execute solution candidates with a "safeguard." This safeguard estimates the runtime of a pipeline before execution and intervenes to prevent its evaluation if a timeout is anticipated. Similarly, (Yang et al., 2019) introduce a runtime prediction component aimed at maximizing information gain in comparison to the time invested in evaluating a pipeline.

The task of predicting the runtime of entire pipelines is undeniably more intricate than predicting the runtime of individual "atomic" learning algorithms. Despite extensive work on algorithm runtime prediction in general (F. Hutter et al., 2014; A. Tornede et al., 2020; L. Huang et al., 2010; K. Smith-Miles et al., 2011; K. Eggensperger et al., 2020) highlight that predicting pipeline runtimes involves more than simply aggregating the runtimes of their components. This is because the output of components, such as pre-processing steps, often ripples through subsequent components, impacting the runtime of other pre-processors or the learner.

Similar strategies, akin to the aforementioned safeguard, find application in the realm of algorithm configuration (F. Hutter et al., 2009; C. Ansòtegui et al., 2009).

In this domain, configuring an algorithm to optimize its runtime involves employing racing or adaptive capping mechanisms (F. Hutter et al., 2009). Adaptive capping, at its core, prematurely halts the evaluation of solution candidates to expedite the optimization process based on specific criteria, such as predefined performance bounds.

2.4.4 Multi-Fidelity Performance Measurements

A different avenue involves transforming evaluations into such a low-cost operation that the concept of timeouts becomes obsolete. The core idea is to employ an inexpensive computational function that approximates the relative performance of a candidate pipeline. Although the performance estimation for a candidate is always present, this is often achieved through the use of "costly" cross-validation procedures involving substantial datasets. The notion of low-fidelity estimation entails utilizing a low-cost estimator trained on economical approximations while maintaining fidelity to the candidate ordering.

One potential strategy is to select models based on evaluations using data subsamples where the models are inexpensive to assess. This approach, proposed in the early stages, has proven to be remarkably effective (J. Petrak, 2000). Rather than assigning a constant prior evaluation sample size, recent approaches in multi-fidelity optimization introduce evaluation fidelity (aka sample size) as a dynamic variable for the optimizer. Crucial parameters influencing the degree of evaluation fidelity include the size of the training set or the number of iterations for iterative learning algorithms such as gradient descent. By evaluating performance at varying degrees of fidelity, customized Bayesian optimization methods can be employed to optimize machine learning pipelines in a cost-effective manner (K. Kandasamy et al., 2017; A. Klein et al., 2017, J. Wu et al., 2019). For example, the FABOLAS approach (A. Klein et al., 2017) actively balances the sample size against expected performance. Similarly, other optimization methods rooted in multi-armed bandits or differential evolution can incorporate concepts from multi-fidelity optimization (L. Li et al., 2017; N. Awad et al., 2021).

Moreover, in an orthogonal direction, there's the prospect of diminishing the number of repetitions in cross-validation. This involves trading the stability achieved through various validation iterations for enhanced evaluation speed. While k-fold cross-validation lacks flexibility in this regard, Monte-Carlo Cross-Validation (MCCV) can be intricately configured, adjusting both the sample size and the number of iterations. Surprisingly, there are currently no studies that have scrutinized, let alone dynamically fine-tuned, MCCV to curtail computational time without sacrificing the order of the candidates.

2.4.5 Early Discarding of Unpromising Candidates

The concept of early elimination entails cutting short the training process of a candidate if it becomes evident that it won't stand out competitively. This strategic move is executed by scrutinizing empirical learning curves. By assessing learning curves that are only partially available, decisions regarding the significance of a specific candidate can be made.

Two distinct approaches can be discerned based on whether they embrace a horizontal or vertical model selection strategy. In a horizontal setting, a predetermined portfolio of candidates is established initially, and learning curves are concurrently developed for expanding anchor sizes, progressing horizontally from left to right. At each anchor point, a subset of candidates is discarded. This forms the crux of successive halving (K. Jamieson, 2016), and horizontal approaches are occasionally dubbed multi-fidelity optimizers. Conversely, in a vertical scenario, candidates are generated sequentially, and each undergoes evaluation on escalating anchor points until it can be forecasted that the candidate won't be competitive. This approach was initially explored for neural networks (T. Domhan, 2015) and more recently incorporated into the Learning Curve-based Cross-Validation scheme (LCCV) applicable to diverse learner types (F. Mohr, 2017). The findings suggest that even for non-iterative learners, the time required to assess a specific portfolio can be trimmed by over 20% on average compared to traditional cross-validation. For portfolios featuring iterative learners, it is envisaged that this enhancement would be even more pronounced.

In the spectrum between these extremes, hybrid methodologies come into play. For instance, Hyperband (L. Li et al., 2017) adheres to a horizontal approach but injects new candidates into the portfolio at each stage. Another strategy, Freeze-Thaw Optimization (K. Swersky et al., 2014) allows for the temporary suspension of training processes, with the option to resume them later if a candidate appears promising once again.

2.4.6 Energy Consumption as Part of the Objective Function

Another possibility is to integrate awareness of energy consumption directly into the AutoML search algorithm. For instance, one could modify Bayesian optimization (BO) for AutoML (C. Thornton et al., 2013; B. Komer et al., 2014; M. Feurer et al., 2021) by introducing a variant of expected improvement that takes into account the energy expended during the evaluation of the next solution candidate. Drawing parallels to the concept of expected improvement per second (J. Snoek et al., 2012), one approach to incorporate energy consumption into the optimization process is to

use expected improvement per kWh of consumed energy as an acquisition function, namely,

$$\mathrm{EIkWh(p)} = \frac{\mathrm{EI(p)}}{\mathrm{kWh(p)}} \tag{2.1}$$

A promising avenue for further exploration may involve the utilization of an acquisition function incorporating both the expected improvement denoted as EI(p) associated with pipeline p and the estimated energy consumption denoted as kW h(p) linked to the evaluation of pipeline p. This approach aims to strike a balance between the information gain of a solution candidate and its execution cost within the framework of Bayesian optimization (BO). However, akin to the methods discussed in Section 2.4.3, this concept relies on prior knowledge about the energy consumption of a specific pipeline before its execution. Initial efforts toward estimating such energy consumption exist (D. Stamoulis et al., 2018) propose a related acquisition function explicitly considering energy constraints on models, but their focus is solely on the energy needed for the inference of a trained network rather than the training energy, which is the subject of our interest. Moreover, one could explore the instantiation of Hyperband (L. Li et al., 2017) with energy as a budget. In principle, a combined approach integrating both concepts could be contemplated to devise a customized version of BOHB (S. Falkner et al., 2018) essentially constituting a hybrid of BO and Hyperband methodologies.

2.4.7 Exploiting Heterogeneous Hardware Resources

Exploring an alternative strategy, one might contemplate the development of AutoML systems that capitalize on the diversity of solution candidates concerning their energy consumption across different hardware configurations to enhance overall efficiency. In this context, leveraging a heterogeneous cluster (comprising various computational devices such as CPUs, GPUs, FPGAs, etc.) and strategically scheduling the evaluation of a solution candidate on the hardware best suited for the specific model in terms of energy efficiency could be considered. For instance, the evaluation of a neural network could be optimized on a GPU, while other types of learners might be more efficiently executed on CPUs. Although this approach may not necessarily expedite the AutoML search process in terms of time, it has the potential to yield improvements in terms of energy consumption. While this is recognized as a potentially intriguing avenue for research, it is important to note that, as of now, there is no awareness of any existing work in this particular direction.

2.4.8 Intelligent Stopping Criteria

Enhancements to AutoML systems can be achieved through the implementation of intelligent stopping criteria, which assess whether further improvement is likely within the remaining runtime. The core idea is to determine if the allocated runtime is genuinely necessary or if the search can be halted prematurely. These considerations share similarities with the principles of early stopping in machine learning (L. Prechelt et al., 2012) and the early stopping criteria commonly employed in the realm of metaheuristics (M. Gendreau et al., 2003).

An illustration of an intelligent criterion is provided by the concepts proposed in LeanML (Yves-Laurent Kom Samo, 2021). This criterion operates on the premise of estimating the highest attainable performance on a dataset, allowing for the premature termination of the AutoML search process when the probability of discovering a superior solution within the remaining time diminishes significantly.

A notable contribution in this direction, acknowledged with the Best Paper Award at the inaugural AutoML conference in 2022, was presented by Makarova et al. Their proposal advocates for concluding hyperparameter optimization when the validation performance is presumed to be in proximity to the achievable performance. This is determined by analyzing the estimated difference between the validation loss and the test loss, with the stopping point being set when it approximately aligns with the estimation error associated with this difference.

2.4.9 Use of Saved Resources

As previously discussed, several approaches can be employed to conserve runtime, including avoiding timeouts (refer to Section 2.4.3), early elimination of unpromising candidates (refer to Section 2.4.5), or implementing intelligent stopping criteria (refer to Section 2.4.8). Broadly speaking, there are two ways to utilize saved resources.

One approach involves early termination, directly impacting the environmental footprint. Alternatively, the search can be extended to consider more solution candidates, potentially minimizing the wastage of allocated resources. It's crucial to emphasize that merely reducing the amount of wasted resources does not necessarily translate into energy savings unless the overall search budget is proportionally reduced based on the saved search time. Nonetheless, such enhancements can be valuable even when the search time remains unchanged, as they contribute to an improved benefit to environmental cost ratio.

2.5 Model Benchmarking

The issue of benchmarking must be addressed since autoML falls within the realm of empirical research, experimentation is an integral component of research endeavors and lies at the core of nearly every scholarly publication. Given the intricate nature of AutoML systems, obtaining theoretical results is notably challenging and often entails making certain simplifications, consequently constraining the breadth of conclusions that can be drawn from the outcomes. Additionally, theoretical findings are typically complemented by experiments to validate their applicability in practical scenarios. In AutoML, experimentation extends beyond testing the newly proposed method and includes comparisons with rival methods serving as benchmarks, thereby showcasing the superiority of the novel approach over the current state-of-the-art. Hence benchmarks help the community to compare the energy footprint of different models or training techniques. The performance of AutoML systems, much like the selection and configuration of base algorithms by AutoML systems, complements each other rather than having a single superior AutoML system representing the state-of-the-art (SOTA). Instead, there exists a range of competitive methods. Consequently, comparisons involve a growing set of baselines, resulting in increased computational costs. For instance, Feurer et al. reported a computational cost of 11 CPU-years[1], while the experimental data from Mohr, Wever, and Hullermeier required 52 CPUy worth of computations. In another study, (M. Wever et al., 2021), the experimental investigation extended to 84 CPUy. In the sub-field of neural architecture search (NAS), the computational costs for experimentation can be even higher, with single AutoML runs utilizing the computational power of 450 GPUs for 7 days (E. Real et al., 2019) or 800 GPUs for 28 days [2]. It's important to note that the published figures typically consider the computational costs of the presented results and not those incurred during development for testing.

The recurrent execution of baselines in almost every study stems from the absence of a standardized experiment setup. The variation in datasets and specific configurations, such as hardware resources, timeouts, search spaces, and AutoML system configurations, from one study to another hinders not only comparability across publications but also reproducibility. This lack of a gold standard for the experiment setup leads to missing crucial details, impacting the ability to reproduce

[1]CPU years is a unit of measurement used to express the cumulative processing power of a central processing unit (CPU) over a certain period. It represents the equivalent amount of computational work that a CPU can perform in one year.

[2]Assuming a GTX 1080 Ti with a Thermal Design Power (TDP) of 250W. The power required by the GPU for each run is 134.4 megawatt-hours (MWh), equivalent to the annual power usage of about 30 households, each with a power demand of 4,250 kilowatt-hours (kWh). This calculation does not account for the power consumption of the rest of the system. On the current AWS GPU Nodes, the cost of a single run amounts to $483,840

the experiments accurately. In the paragraphs to come, the usefulness of benchmarking is analyzed. Several benchmarks have been proposed in order to ensure reproducibility and comparability. It is also useful in democratizing research, then clarifying the need to generate so-called performance curves.

Reproducibility and Comparability From a research standpoint, benchmarks serve the purpose of providing a shared platform for empirical studies. The primary objective is to establish or enhance comparability and reproducibility of results. In the field of automated machine learning, this involves defining specific variables such as the dataset selection, the target metric for optimization, time constraints for evaluating individual candidate solutions, and the overall runtime of the AutoML system. Additionally, factors like the hardware specifications, the definition of the search space, and others need to be considered.

For instance, the AutoML benchmark by OpenML [3] comprises 39 datasets. In this benchmark, each AutoML system is allotted a total of 4 hours to search for an appropriate pipeline. Each run is repeated ten times with different seeds. The benchmark recommends using Amazon AWS m5.2xlarge compute nodes, equipped with an Intel Xeon Platinum 8000 Series Skylake-SP processor featuring 8 CPU cores and 32GB memory. This choice aligns with the hardware specifications commonly found in AutoML publications, and it allows accessibility to anyone with such compute nodes. According to the benchmark specifications, evaluating a single AutoML system requires 12,480 CPU hours. Therefore, estimating the computational resources for an entire study involves multiplying the CPU hours by the number of considered AutoML systems or baselines. Fortunately, when utilizing the same computing infrastructure and exact experiment setup, there's no need to reevaluate already benchmarked AutoML systems, as the results should, in principle, be comparable.

It is crucial to note that meeting all the different criteria precisely is essential to ensure comparability. In the literature, results are at times borrowed directly from previous publications without considering changes in the experiment setup, based on which the newly proposed method is evaluated. While this approach minimizes energy usage, the results become incomparable, and valid conclusions are challenging to draw. Although the energy consumption is relatively low, energy efficiency is compromised, as the information obtained through energy investment is not as valuable as desired. Various benchmarks for AutoML systems have already identified numerous confounding factors (P. Gijsbers et al., 2019; A. Balaji et., 2018), which hinder the interpretation of results and insights derived from them.

[3] `https://openml.org/search?type=benchmark&sort=tasks_included&study_type=task&id=293`

Democratizing Research Diverging from the focus on reproducibility and comparability, benchmarks can serve an additional purpose enhancing the sustainability of experiments and broadening access to research in the realm of highly computationally intensive problems. For instance, various benchmarks have surfaced within the AutoML subfield of neural architecture search (NAS). In these benchmarks, every conceivable architecture within a predefined search space undergoes evaluation once, and the resultant performance metrics are cataloged in a reference table. When assessing novel strategies for optimizing neural architectures, one can conveniently refer to the performance of a proposed solution in this table, obviating the need to undertake the resource intensive processes of training and validating the specific architecture. Typically, these benchmark lookup tables encapsulate performance data from approximately 50,000 diverse architectures. Although the creation of such benchmarks entails notable costs, they represent a singular investment. Furthermore, since evaluating this extensive array of architectures necessitates substantial GPU computations, these benchmarks open avenues for researchers and practitioners without access to high end computational resources to actively participate in NAS research.

Given the limited number of possible architectures in the aforementioned benchmarks, Siems et al. advocate for a surrogate model to predict the performance of a neural architecture. This surrogate model is trained on the performance data of around 60,000 different architectures and exhibits robust generalization to new, previously unseen examples. The utilization of a surrogate model introduces greater flexibility, offering performance estimates for candidate solutions that haven't undergone direct evaluation. Hence, benchmarks revolving around such surrogate models have the potential for even greater sustainability. However, the construction of these models demands meticulous attention to prevent the propagation of misleading results in the exploration of new methods. Additionally, to attain the high-quality predictive capabilities of the surrogate model, Siems et al. must generate an extensive amount of training data, involving the evaluation of 60,000 architectures for this purpose. While this entails significant computational effort, it represents a one time cost that yields returns with each subsequent evaluation. Nevertheless, when employing a surrogate model, constraints on the use of the benchmark need to be imposed. Specifically, the benchmark specification mandates the surrogate model to be used exclusively in a query only mode, limiting any approach to requesting performance estimates for specific neural architectures. Consequently, this excludes methods that seek to exploit the surrogate model, such as analyzing its internal structure.

In summary, benchmarks emerge as potent tools for elevating the sustainability of AutoML research, shifting the focus from repetitive evaluations of candidate solu-

tions to energy conservation and broadening the participation of institutions lacking essential resources. Moreover, research acceleration becomes feasible, with evaluations of candidate solutions requiring mere milliseconds rather than minutes, hours, or days. Therefore, the advocacy and encouragement of benchmark use should be underscored, as it amalgamates several inherent advantages of benchmarks. However, within the research community, vigilance is necessary to avert the simultaneous development of multiple highly similar or potentially identical benchmarks, as this would unjustifiably inflate energy costs. Ideally, benchmark development should evolve as a collaborative endeavor communicated at an early stage, akin to the approach undertaken in the case of DACBench4, for example.

Performance curve When appraising the efficiency of an AutoML system, a mere juxtaposition of the ultimate performances upon the culmination of the entire run falls short. Different AutoML approaches manifest distinct performance trajectories, with some exhibiting robust anytime performance and others peaking in the final stages. Therefore, a thorough evaluation entails scrutinizing the performance curve of the system, elucidating the anticipated solution quality after expending a designated computational budget. Figure 2.3 furnishes a depiction of the ecological performance profiles of two competing AutoML systems, illustrating that the blue tool attains commendable performance well in advance of the orange tool.

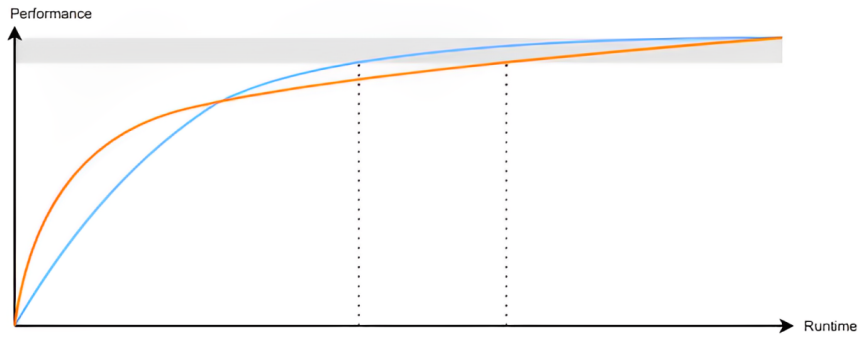

Figure 2.3: Illustrative depiction of two performance trajectories, wherein the azure tool attains commendable performance metrics (within the grey-shaded region) significantly ahead of the orange counterpart. Source: (T. Tornede et al., 2022).

Ideally, an exploration of an ecological performance profile would establish a correlation between predictive performance and the volume of CO2e emissions generated during computations. Nevertheless, as previously expounded in Section 2.3, opting to investigate CPU/GPU hours over the actual environmental impact proves more pragmatic. While it is assumable, for any judicious AutoML system, that performance profiles exhibit a monotonically increasing trend, signaling improved performances with a larger budget, the curves of individual approaches might inter-

sect. The ecological performance profile of an AutoML system may hinge on diverse factors, including runtime, the degree of parallelism, the adeptness in leveraging heterogeneous execution environments, and more. Assuming a consistent hardware setting for all contenders, runtime can indeed serve as a viable surrogate. Access to such performance profiles empowers users to opt for the AutoML system most harmonious with their budget and CO2e footprint. Relying on a solitary evaluation at a fixed point proves inadequate for making such a determination. However, it is imperative to acknowledge that these performance profiles are still contingent on the hardware employed to assess the performance of the approaches and derive their respective profiles. Performance curves tend to exhibit this nonlinear trend. In addition to the reflections made earlier about the possibility of identifying the best algorithm based on the available budget, other considerations can be made by analyzing the figure. To emphasize the necessity of detailing the computational

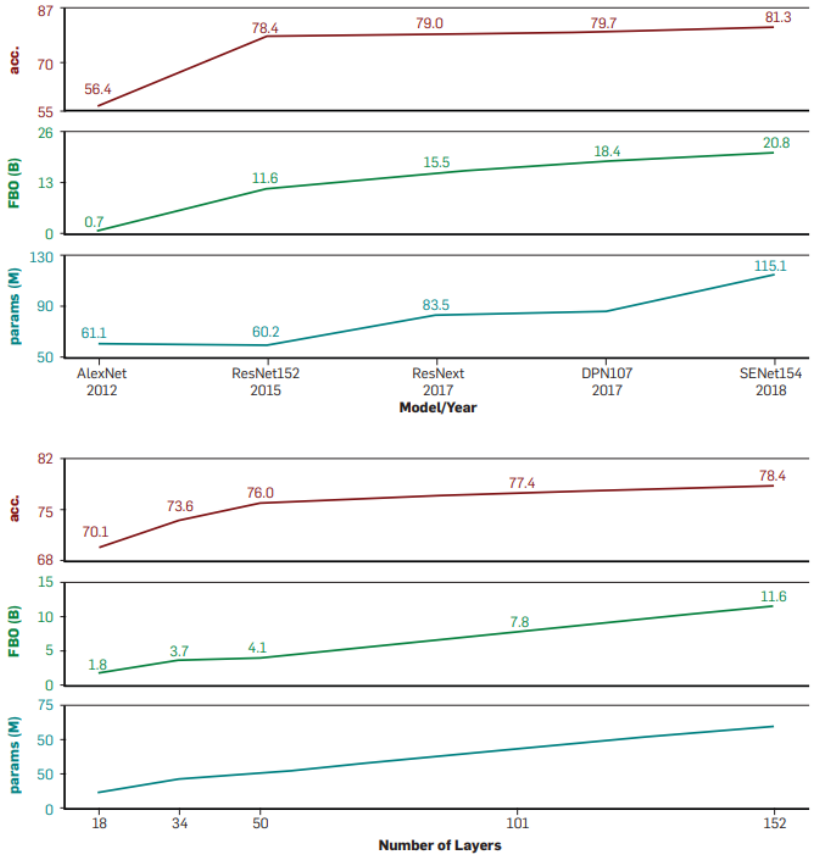

Figure 2.4: The escalation in Floating Point Operations (FPO) results in diminishing returns for top-1 accuracy in object detection. The plots (from bottom to top) display model parameters (in millions), FPO (in billions), and top-1 accuracy on ImageNet. In Figure 2.4(a), various prominent object recognition models, including AlexNet, ResNet, ResNext, DPN107, and SENet154, are showcased. Figure 2.4(b) offers a comparison of different sizes, measured by the number of layers, of the ResNet model.

complexities involved, insights into the FPO costs associated with various existing models are provided. In Figure 2.4(a), the number of parameters are illustrated and FPO for several notable object recognition models, along with their corresponding performance metrics on the ImageNet dataset. Several discernible patterns emerge from the dataset. First and foremost, aligning with prior discussion, models exhibit a growing computational burden over time; however, the upswing in FPO does not yield proportional enhancements in performance. For instance, an almost 35% surge in FPO between ResNet and ResNext (second and third data points on the graph) results in only a marginal 0.5% enhancement in top-1 accuracy. Similar trends surface when scrutinizing the impact of other increments in the computational workload of models. Secondly, relying solely on the number of model parameters proves insufficient for a comprehensive understanding: AlexNet (first data point on the graph) boasts more parameters than ResNet (second data point) but significantly fewer FPO and markedly lower accuracy.

In Figure 4(b), this analysis is extended to a specific object recognition model, ResNet, by comparing different versions with varying numbers of layers. This experimental setup ensures a controlled comparison, as the models share identical architectures but differ in size (and consequently, FPO costs). Once again, the overarching trend prevails: a substantial increase in FPO cost does not correspond to a proportionate increase in performance.

2.6 Transparency

Ensuring transparency regarding efficiency and environmental impact represents a crucial stride towards Green AutoML and, more broadly, a sustainable global perspective a practice that should be obligatory when presenting scholarly papers. While the typical acceptance criteria of a paper are centered around novelty and performance enhancement, factoring in the environmental impact becomes increasingly vital. Outright rejection of a paper based solely on environmental considerations, despite robust results, would be deemed unjustifiable since, ultimately, pollution cannot be undone, and pollution coupled with documented results is more preferable than pollution without. However, the emphasis should be on encouraging authors to minimize environmental impact proactively.

Consequently, as a preliminary measure, the expansion of existing checklists in major machine learning conferences with a set of additional questions is endorsed. These checklists must be completed before paper submission and are crafted to guide authors in contemplation of crucial aspects such as reproducibility, transparency, research ethics, and societal impact. Additionally, authors could be prompted to furnish details about the computational resources utilized in their paper and any

initiatives taken to quantify or mitigate resource consumption.

Transparency also involves divulging information about offsetting the ecological footprint and, preferably, detailing the offsetting method. For instance, if offsetting involves tree planting, those trees need to endure for a minimum of ten years to effectively neutralize the footprint. This heightened awareness regarding the considerable time required for consumed resources to be adequately compensated should encourage researchers to conduct experiments more judiciously.

Moreover, as a collective community effort, fervent advocacy for the publication of unsuccessful attempts and negative results is proposed to advance Green AutoML and sustainable scientific practices more broadly. Instances abound where, despite a promising concept, extensive effort, and experimentation, an idea failed to perform as anticipated in practice and, consequently, remained unpublished. This is regrettable not only from an environmental standpoint but also on a scientific level. Given the burgeoning scientific community, there's a high likelihood that someone else might work on the same idea and arrive at a similar conclusion, potentially averting redundant efforts. Furthermore, withholding negative results can stymie scientific progress, as even if one considers an idea thoroughly explored, another researcher might discover a way to turn a negative outcome positive with the right adjustments. In the context of the substantial computational resources required for AutoML research, there's an even more pressing need to devise a method for sharing unsuccessful attempts. An online compilation of journals focusing on negative results is available.

2.7 Emissions of popular offthe-shelf NLP models

Recent strides in the methodologies and hardware utilized for the training of deep neural networks have ushered in notable strides in accuracy across a spectrum of foundational Natural Language Processing (NLP) tasks (Bahdanau et al., 2015; Luong et al., 2015; Dozat and Manning, 2017; Vaswani et al., 2017). Notably, the most computationally intensive models have secured top performance scores (Peters et al., 2018; Devlin et al., 2019; Radford et al., 2019; So et al., 2019). Consequently, training a state-of-the-art model now necessitates substantial computational resources, incurring significant energy consumption and associated financial and environmental costs.

The continuous research and development of new models further amplify these costs by mandating frequent retraining to explore diverse model architectures and hyperparameters. A decade ago, most NLP models could be trained and developed on a standard laptop or server, but today, many require multiple instances of specialized hardware such as GPUs or TPUs, thereby limiting access to highly accurate

models based on financial constraints. Even when these costly computational resources are available, model training exacts a substantial environmental toll due to the energy required to power the hardware over weeks or months.

Despite the potential utilization of renewable or carbon credit-offset energy sources, the substantial energy demands of these models remain a concern. This is further exacerbated by the fact that, in many locations, energy is not currently sourced from carbon-neutral outlets. Moreover, even when renewable energy is accessible, it is constrained by the infrastructure for production and storage. In this context, the energy expended on training neural networks might be more effectively allocated to meeting the heating needs of households.

Consumption	**CO_2e (lbs)**
Air travel, 1 person, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experiments	78,468
Transformer (big)	192
w/ neural arch. search	626,155

Table 2.2: Projected carbon dioxide emissions arising from the training of prevalent NLP models, juxtaposed against commonplace consumption.

The imperative to halve carbon emissions in the next decade to mitigate the surging frequency of natural disasters is underscored by estimates presented in Table 1, revealing that a substantial share of greenhouse gas emissions within the NLP research community is likely tied to the training and development of models.

In an effort to cultivate awareness and advocate for mindful practices and policies within the NLP community, the economic and environmental repercussions stemming from the training of neural networks pivotal to numerous state-of-the-art NLP models are delved into. This involves approximating the kilowatt-hours of energy needed for training various widely-used off-the-shelf NLP models, subsequently translating these figures into estimates of carbon emissions and associated electricity costs.

To assess the computational and environmental toll incurred by the training of deep neural network models in NLP, an examination of the energy consumption associated with the training process for various widely-used off-the-shelf NLP models is conducted. Additionally, a case study involving the comprehensive resource re-

quirements for the development of LISA, a state-of-the-art NLP model showcased at EMNLP 2018, encompassing all tuning and experimentation aspects, is undertaken.

Energy consumption measurements involve training the models described subsequently with default settings, utilizing a single NVIDIA Titan X GPU for all models except ELMo, which was trained on 3 NVIDIA GTX 1080 Ti GPUs. GPU and CPU power consumption during training are monitored, querying the NVIDIA System Management Interface[4] and using Intel's Running Average Power Limit interface[5], respectively. The training duration is capped at 1 day for each model. The total training time is estimated based on reported hardware and training times from the original papers.

To compute power consumption in kilowatt-hours (kWh), average power draw from CPU sockets (p_c), DRAM sockets (p_r), and GPUs (p_g), where g is the number of GPUs used, is considered. The total power consumption is estimated by combining GPU, CPU, and DRAM consumption and multiplying by the Power Usage Effectiveness (PUE) coefficient of 1.58, reflecting the 2018 global average for data centers (Ascierto, 2018). The formula for total power (pt) required during training is given by:

$$\mathrm{p_t} = \frac{1.58\mathrm{t}(\mathrm{p_c} + \mathrm{p_r} + \mathrm{gp_g})}{1000} \tag{2.2}$$

The U.S. Environmental Protection Agency (EPA) provides average CO2 produced (in pounds per kilowatt-hour)[6] for power consumed in the U.S. (EPA, 2018), which is used to convert power to estimated CO2 emissions:

$$\mathrm{CO_2e} = 0.954\mathrm{p_t} \tag{2.3}$$

This transformation considers the respective distribution of various energy sources (mainly including natural gas, coal, nuclear, and renewable sources) utilized for energy production in the United States as shown in the table.

In the following four models are assessed, providing an overview of their computational demands. The source code for all models is publicly available and was used as-is. For a comprehensive understanding of each model, please refer to the respective original papers.

Transformer: The Transformer model (T2T) by Vaswani et al. (2017) employs an encoder-decoder architecture with 6 stacked layers of multi-head self-attention in both the encoder and decoder. The $T2T_{base}$ model (65M parameters) was trained on 8 NVIDIA P100 GPUs for 12 hours, while the larger $T2T_{big}$ model (213M parameters) required 3.5 days (84 hours; 300k steps).

[4]nvidia-smi: `https://bit.ly/30sGEbi`

[5]RAPL power meter: `https://bit.ly/2LObQhV`

[6]Conversion factor to convert pounds (lbs) to kilograms (kg): 1 pound = 0.453592 kilograms

Consumer	Renew.	Gas	Coal	Nuc.
China	22%	3%	65%	4%
Germany	40%	7%	38%	13%
United States	17%	35%	27%	19%
Amazon-AWS	17%	24%	30%	26%
Google	56%	14%	15%	10%
Microsoft	32%	23%	31%	10%

Table 2.3: Percentages of energy derived from various sources, including renewables (such as hydro, solar, wind, natural gas, coal, and nuclear, were analyzed for the leading three cloud computing providers(Cook et al., 2017). The comparison extends to energy sourcing in the United States, China , and Germany(Burger, 2019).

ELMo: Based on stacked LSTMs, the ELMo model (Peters et al., 2018) produces context-rich word representations through pre-training on extensive data using a language modeling objective. ELMo was trained on 3 NVIDIA GTX 1080 GPUs for 2 weeks (336 hours).

BERT: Devlin et al.'s (2019) BERT model, utilizing a Transformer-based architecture, offers contextual representations akin to ELMo but with a different language modeling objective. The $BERT_{base}$ model (110M parameters) was trained on 16 TPU chips for 4 days (96 hours). NVIDIA reports training a BERT model in 3.3 days (79.2 hours) using 4 DGX-2H servers with 64 Tesla V100 GPUs.

GPT-2: OpenAI's GPT-2, the latest iteration of their general-purpose token encoder, relies on Transformer-style self-attention and is trained with a language modeling objective (Radford et al., 2019). The large model with 1542M parameters required 1 week (168 hours) of training on 32 TPUv3 chips for high zero-shot performance on benchmarks like question answering and language modeling.

Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$T2T_{base}$	P100x8	1415.78	12	27	26	$41–$140
$T2T_{big}$	P100x8	1515.43	84	201	192	$289–$981
ELMo	P100x3	517.66	336	275	262	$433–$1472
$BERT_{base}$	V100x64	12,041.51	79	1507	1438	$3751–$12,571
$BERT_{base}$	TPUv2x16	—	96	—	—	$2074–$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	$942,973–$3,201,722
NAS	TPUv2x1	—	32,623	—	—	$44,055–$146,848
GPT-2	TPUv3x32	—	168	—	—	$12,902–$43,008

Table 2.4: Estimated cost of training a model in terms of CO2 emissions (lbs) and cloud compute cost (USD). Power and carbon footprint are omitted for TPUs due to lack of public information on power draw for this hardware.

Table 2.4 provides an overview of the carbon dioxide emissions and estimated expenses associated with the training of the models. It is worth emphasizing that TPUs demonstrate higher cost efficiency compared to GPUs, especially for work-

loads optimized for TPU hardware, as exemplified by BERT. Furthermore, the table underscores the significant carbon footprint attributed to these models; the training of BERT on GPUs is tantamount to the emissions generated by a trans-American flight. Notably, the study by So et al. (2019) underscores that while Neural Architecture Search (NAS) achieves a marginal 0.1 BLEU score improvement, it incurs costs exceeding $150,000 in on-demand compute time and contributes to substantial carbon emissions.

Recognizing the imperative need to change course, the substantial resource consumption by state-of-the-art models like BERT and GPT poses a significant challenge. This awareness compels us to explore more sustainable and efficient solutions so that innovation in artificial intelligence can progress in harmony with the environmental and ethical needs of our society.

Chapter 3

Recent Advances in Optimization

AI systems are significantly complex, and achieving Green AI requires a collective effort that addresses all the different stages of an AI system's lifecycle (e.g., data collection, training, monitoring) and various components (e.g., data, model, pipeline, architecture, hardware), etc (M. Haakman et al., 2021). Given the heterogeneity of the field, it is also difficult to have a broad view of all the Green AI literature that has been published in the past years. To understand the existing research, Verdecchia et al. (2023) conduct a systematic literature review on Green AI. They provide an overview and characterization of the existing research in this field. The most popular topics revolve around monitoring, hyperparameter tuning, deployment, and model benchmarking.

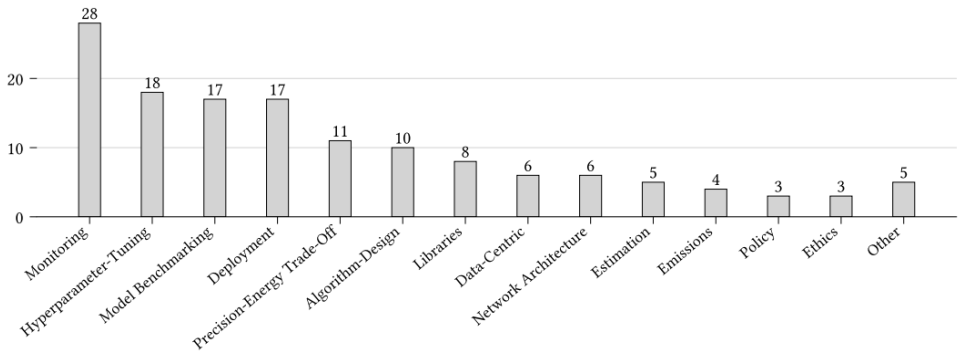

Figure 3.1: Identification of 13 main topics being addressed by the Green AI literature. Most studies consider monitoring AI model footprint, tuning hyperparameters to improve model sustainability, or benchmarking models. Source: (Verdecchia et al., 2023)

In 18 out of 98 papers, the issue of improving or assessing the impact on energy consumption by optimizing hyperparameters during AI model training is addressed. Many publications are motivated by the fact that tuning parameters leads to significant energy costs. It requires retraining a model multiple times in order to find the optimal set of hyperparameter values. Hence, most publications within this

topic focus on identifying alternative strategies that reduce the number of iterations required to tune hyperparameters (D. Stamoulis et al., 2018). On a different perspective, Chavannes et al. (2021) explore how hyperparameter tuning can help deliver more energy-efficient models by adding power consumption to the set of parameters being optimized. To make the more "greener", Bayesian Optimization gains prominence.

Bayesian Optimization (BO) stands out as a resourceful, sequential, model-driven approach to global optimization, ideally tailored for enhancing black-box, costly, and multi-extremal objective functions (Frazier, 2018; Archetti and Candelieri, 2019; Garnett, 2023). Its adeptness in handling samples positions BO at the core of the majority of existing AutoML solutions, spanning both open-source and commercial domains. Recent enhancements have propelled BO to grapple with diverse information sources, each incurring distinct querying costs (Ghoreishi and Allaire, 2019; Belakaria et al., 2020; Candelieri et al., 2021; Candelieri and Archetti, 2021b; Khatamsaz et al., 2020) and also to navigate multiple objectives (Hernández-Lobato et al., 2016; Paria et al., 2020).

The concept of multi-fidelity optimization, initially conceptualized in (Kennedy and O'Hagan, 2000) is a noteworthy scenario unfolds when information sources can be systematically arranged based on their fidelity. Whether dealing with multiple information sources or engaging in multi-fidelity optimization, the strategy for achieving energy efficiency involves astutely leveraging the more economical (and lower fidelity) sources, thereby maintaining a frugal cumulative query cost. This cumulative query cost serves as a stand-in for energy consumption and, consequently, for the emission of CO2. This multi-faceted optimization approach dovetails with the Gaussian Process-oriented Bayesian Optimization framework. This chapter also introduces an inventive method named Augmented Gaussian Process Multi-Information Source Optimization (AGP-MISO). The Augmented Gaussian Process memo, undergoes training exclusively with information identified as "reliable" among the available sources. Furthermore, a pioneering acquisition function is conceived based on the Augmented Gaussian Process.

Moreover in this chapter it is analyzed the multi-objective optimization, aimed at minimizing, simultaneously, misclassification error and some unfairness metric (Schmucker et al., 2020). Namely the aim is to search for machine learning models which offer equally Pareto efficient trade-offs. This is due to there is a consensus that focusing only on accuracy in searching for optimal machine learning models amplifies biases contained in the data, soleading to unfair predictions and decision supports. Finally, when referring to the approaches used in the case study in Chapter 4, approaches that predominantly leverage the Bayesian optimization method will be examined. These approaches have been employed with the overarching goal

of attaining outcomes that are both energy-efficient and fairness, by exploiting the notions of multi-fidelity and multi-objective optimization. The methods in question are: **autogluon-FairBO** (Schmucker et al., 2020), **BoTorch-MOMF** (Multi-Objective and Multi-Fidelity) (Irshad et al., 2021), **FanG-HPO** (Fair and Green Hyperparameter Optimization) (Candelieri et al., 2022).

3.1 Bayesian Optimization

Global optimization poses the challenge of identifying the optimum, specifically the global maximum or minimum of a given function f, often constrained by boundary conditions on the variables. In instances where f is analytically available and differentiable, traditional gradient-based optimization methods, fine tuned for seeking the global optimum, come into play (S. Boyd and L. Vandenberghe, 2004). However, real world scenarios frequently involve functions that deviate from these assumptions. In such cases, diverse derivative free approaches, such as genetic algorithms and the DiRect methods, offer viable alternatives. These techniques typically involve evaluating f at multiple points, making them well-suited for scenarios where assessing f does not entail significant resource consumption.

In practical applications, however, one often grapples with functions that demand substantial resources for evaluation, whether in terms of time, financial investment, or computational power. Jones et al. (1998) provide an illustration from geostatistics, where the aim is to pinpoint the region with the highest underground concentration of a valuable mineral based on a core sample. Owing to the costliness of multiple unsuccessful drilling attempts, a probabilistic method known as kriging, essentially akin to Bayesian optimization, has surfaced. Bayesian optimization, predominantly utilized in its standard formulation (J. Mockus, 1975), leverages all past function evaluations and a probabilistic model of f to navigate the search for the global optimum. As will be observed, this provides the advantage of significantly reducing the number of function evaluations compared to alternative methodologies.

A strategy steering the quest for the optimum hinges on striking a balance between **exploitation** and **exploration** concepts, favoring regions where, based on prior evaluations, the optimum is likely to reside, or alternatively, unexplored territories. To achieve this essential balance, there are various strategies based on the selection of an acquisition function, as elucidated in Section 3.1.2. Numerical simulations unfold on an illustrative example, accompanied by the analysis and visualization of these search policies. Additionally, connections are drawn between the numerical outcomes and the analytic expressions of the policies, thereby affirming their anticipated behavior.

Furthermore, Bayesian optimization has recently found application across a spec-

trum of contemporary techniques, including robotics, reinforcement learning, and the fine-tuning of machine learning hyperparameters (J.S. Bergstra et al., 2011). The latter application centers on pinpointing the set of hyperparameters conducive to the optimal performance of a model. Given that the objective function lacks an analytical expression and the model training is computationally demanding, Bayesian optimization emerges as a premier contender for addressing this challenge.

Basic idea of Bayesian optimization

Bayesian optimization represents a non-derivative approach to global optimization, drawing inspiration from the Bayesian concept of continually refining the model based on observed data. In mathematical terms, the focus is on solving the problem:

$$\max_{x \in X} f(x)$$

where $f(x) : X \to \mathbb{R}$ is called the *objective function*. Throughout this work, it is assumed that the domain of optimization X is a subset of $\mathbb{R}^d$, $d \in \mathbb{N}^+$ Bayesian optimization falls into the category of sequential optimization algorithms as it systematically seeks an improved optimum through iterative processes. The fundamental assumption is that a set of objective function values, along with their corresponding points in the domain X, is already known. The core concept involves establishing a probabilistic distribution over these values and leveraging it to infer the distribution of the function value at any new point within the optimization domain. This inferred distribution guides the formulation of a policy, determining the selection of a more favorable optimum. Following this selection, the set of observations is expanded to include the true value of the objective at the associated point, and the entire process is repeated.

This concept is formalized through two crucial components in Bayesian optimization: a *Gaussian process* serving as the underlying probabilistic model for the objective function, and the *acquisition function*, acting as a criterion for selecting a superior optimum under the assumption of this model. The key aspect of Bayesian optimization lies in the cost-effectiveness of computing the involved distributions, including the acquisition function, in contrast to the expensive evaluation of the objective.

In the subsequent sections 3.1.1 and 3.1.2, theoretical introductions and justifications for these two essential elements of Bayesian optimization will be provided. Finally, the overall concept will be encapsulated by presenting a general algorithm in section 3.1.3.

3.1.1 Gaussian Process

Formally, the probabilistic component of Bayesian optimization is the assumption that the objective function $f : \mathcal{X} \to \mathbb{R}$ It represents the manifestation of a fundamental Gaussian process, as defined in Definition (2.4, in Appendix). Moreover, it is essential to bear in mind that in the Bayesian optimization framework, this access is restricted to a finite set of observations. This implies a lack of knowledge of either the realization or the underlying process. Consequently, in practical terms, the only available option is to make an assumption about the values of the objective function. Specifically, they are considered as evaluations of the underlying Gaussian process at a given p $\mathbf{x} \in \mathcal{X}$ which, as per Definition (2.2, in Appendix), are random variables. In particular, according to Definition (2.4, in Appendix), any finite set of function values is presumed to follow a corresponding finite-dimensional joint Gaussian distribution (C. E. Rasmussen and C. K. I. Williams, 2006). As stated in 3.1, Supposing a sample of n observations $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^{\mathrm{T}} \subset \mathcal{X}^n$ and the associated objective function values $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^{\mathrm{T}} \subset \mathbb{R}^n$ is given. Following the Gaussian assumption, it is observed that

$$\mathbf{f} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}), \tag{3.1}$$

where $\mathbf{m}$ is an $n \times 1$ mean vector and $\mathbf{K}$ an n + n covariance matrix. Consider a scenario where a new, as-yet-unexplored sample is under consideration, $(n+1)$st point $\mathbf{x}_* \in \mathcal{X}$. Now, instead of evaluating the costly objective at this point, It is possible to predict the value of the objective function $f_* = f(x_*)$ under the Gaussian process assumption. In the Bayesian setting, this amounts to computing the posterior distribution of f_* given the previous set of observations $\mathbf{f}$. Hence, the finite collection

$$\begin{bmatrix} f(\mathbf{x}_*) \\ f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} f_* \\ \mathbf{f} \end{bmatrix}$$

follows a joint normal distribution, that is

$$\begin{bmatrix} f_* \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_* \\ \mathbf{m} \end{bmatrix}, \begin{bmatrix} k_{**} & \mathbf{k}_*^{\mathrm{T}} \\ \mathbf{k}_* & \mathbf{K} \end{bmatrix} \right), \tag{3.2}$$

where in this context, having divided both the mean vector and the covariance matrix to highlight the specific components that pertain to the preceding sample, those that relate to the new point, and those that apply to both. Therefore, there

is that k_* is an n $a \times b$ 1 vector of covariances between the previous sample and the new observation simply defined as

$$\mathbf{k}_* = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_*) \\ \vdots \\ k(\mathbf{x}_n, \mathbf{x}_*) \end{bmatrix}$$

and similarly that $m_* = m(\mathbf{x}_*)$ and $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$. Recall that the goal is to find the posterior distribution of f_* given $\mathbf{f}$. This is when the convenience of using a Gaussian process assumption becomes evident. It is demonstrated that this posterior distribution is also Gaussian, meaning that it is completely analytically tractable and simple to compute. To do this, proceeding to obtain the probability density function $p(f_* \mid \mathbf{f})$, which is by Theorem 1

$$p(f_* \mid \mathbf{f}) = \frac{p(\mathbf{f}, f_*)}{p(\mathbf{f})}, \tag{3.3}$$

where

$p(\mathbf{f}, f_*)$ is a joint probability density function of f_* and $\mathbf{f}$, and $p(\mathbf{f})$ the probability density function of $\mathbf{f}$. From (3.1) and (3.2), these densities are given as

$$p(\mathbf{f}) = \frac{1}{(2\pi)^{n/2}} |\mathbf{K}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{f}-\mathbf{m})^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f}-\mathbf{m})\right) \tag{3.4}$$

and

$$p(\mathbf{f}, f_*) = \frac{1}{(2\pi)^{(n+1)/2}} |\mathbf{K}|^{-1/2} \exp\left(-\frac{1}{2} \begin{bmatrix} f_* - m_* \\ \mathbf{f} - \mathbf{m} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} k_{**} & \mathbf{k}_*^{\mathrm{T}} \\ \mathbf{k}_* & \mathbf{K} \end{bmatrix}^{-1} \begin{bmatrix} f_* - m_* \\ \mathbf{f} - \mathbf{m} \end{bmatrix}\right) \tag{3.5}$$

It now remains to insert them into (3.3) and simplify the expression. However, initially, two results need to be established.

Lemma 1. For a square matrix $\mathbf{A}$ partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

it is observed that

$$|\mathbf{A}| = \left|\mathbf{A}_{22}\right| \left|\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\right|,$$

for $|\mathbf{A}_{22}| \neq 0$. Proof. See (R. A. Johnson and D. W. Wichern, 2007). Applying this lemma to the covariance matrix in (3.5) it is obtained.

$$\begin{vmatrix} k_{**} & \mathbf{k}_*{}^{\mathrm{T}} \\ \mathbf{k}_* & \mathbf{K} \end{vmatrix} = |\mathbf{K}| \left| k_{**} - \mathbf{k}_*{}^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}_* \right|.$$

The second required outcome involves expressing the exponent of a joint probability density function as the summation of terms attributable to contributions from both the conditional and marginal distributions. Specifically, the following relationship holds.

Lemma 2. For $q \times 1$ vectors $\mathbf{x}$ and $\boldsymbol{\mu}$ partitioned as $\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$ and $\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$, respectively, and a $q \times q$ symmetric matrix $\boldsymbol{\Sigma}$ partitioned as $\begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$ it is observed that

$$(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) = \alpha^{\mathrm{T}} \left(\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} \right)^{-1} \alpha + (\mathbf{x}_2 - \boldsymbol{\mu}_2)^{\mathrm{T}} \boldsymbol{\Sigma}_{22}^{-1} (\mathbf{x}_2 - \boldsymbol{\mu}_2),$$

where $\alpha = \mathbf{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} (\mathbf{x}_2 - \boldsymbol{\mu}_2)$. Proof. See (R. A. Johnson and D. W. Wichern, 2007).

Once again, within the context of the joint Gaussian probability density function (3.5), it is observed that, since: the covariance matrix is symmetric, that the Lemma 2 gives

$$\begin{bmatrix} f_* - m_* \\ \mathbf{f} - \mathbf{m} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} k_{**} & \mathbf{k}_*{}^{\mathrm{T}} \\ \mathbf{k}_* & \mathbf{K} \end{bmatrix}^{-1} \begin{bmatrix} f - m_* \\ \mathbf{f} - \mathbf{m} \end{bmatrix} = \alpha^{\mathrm{T}} \left(k_{**} - \mathbf{k}_*{}^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}_{\psi} \right)^{-1} \boldsymbol{\alpha} + (\mathbf{f} - \mathbf{m})^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f} - \mathbf{m}),$$

where now $\alpha = f_* - m_* - \mathbf{k}_*{}^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f} - \mathbf{m})$.

Returning now to (3.3) and using these results, it can be derived that the posterior probability density function is obtained as

$$\begin{aligned} p(f_* \mid \mathbf{f}) &= \frac{p(\mathbf{f}, f_*)}{p(\mathbf{f})} \\ &= \frac{(2\pi)^{n/2} |\mathbf{K}|^{1/2}}{(2\pi)^{(n+1)/2} |\mathbf{K}|^{1/2} |k_{**} - \mathbf{k}_*^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}_*|^{1/2}} \times \\ &\exp\left(-\frac{1}{2} \left(\boldsymbol{\alpha}^{\mathrm{T}} (k_{**} - \mathbf{k}_*^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}_*)^{-1} \boldsymbol{\alpha} + (\mathbf{f} - \mathbf{m})^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f} - \mathbf{m}) - (\mathbf{f} - \mathbf{m})^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f} - \mathbf{m}) \right) \right) \\ &= \frac{1}{(2\pi)^{1/2}} \left| k_{**} - \mathbf{k}_*^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}_* \right|^{-1/2} \exp\left(-\frac{1}{2} \boldsymbol{\alpha}^{\mathrm{T}} (k_{**} - \mathbf{k}_*^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{k}_*)^{-1} \boldsymbol{\alpha} \right) \end{aligned}$$

where $\boldsymbol{\alpha} = f_* - m_* - \mathbf{k}_*{}^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f} - \mathbf{m})$, as before. It is possible to recognize that the obtained probability density of the posterior distribution of f. given previous observations f has a form of the Gaussian probability density function with the mean:

$$\mu_{f,|\mathbf{r}}(\mathbf{x}_*) = m_* + \mathbf{k}_*{}^{\mathrm{T}}\mathbf{K}^{-1}(\mathbf{f} - \mathbf{m}) \tag{3.6}$$

and the variance:

$$\sigma^2_{f,|\mathbf{r}}(\mathbf{x}_*) = k_{**} - \mathbf{k}_*{}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{k}_\psi \tag{3.7}$$

The mean and variance obtained from the posterior distribution completely characterize the desired distribution. Consequently, these parameters will be employed in the acquisition function to determine the subsequent point for assessing the true objective.

Remember that a Gaussian process is fully defined by its mean function m and covariance function k. Therefore, the selection of these functions involves imposing specific properties on the objective function. Given that the underlying Gaussian process is not known, both its mean and covariance functions are also unknown. Consequently, it is the responsibility of the user to determine how to model them. In this subsection, a brief outline of some of the most common approaches will be provided.

Mean Function Typically, estimating the mean function involves performing regression on the existing data. The form of regression can vary, such as constant, linear, or polynomial, depending on the assumptions about the objective function. When there is no prior knowledge available, a common choice is constant regression, which involves calculating the sample average of the existing function evaluations. In this context, the mean m_* for an as-yet-unobserved point x_* is identical to the mean of the previous observations.

Covariance Function The covariance function often holds more significance as it captures the concept of similarity between function values, thereby embodying a prior understanding of the function's behavior. Several suggested covariance functions exist, including periodic ones, are available, but the preference is given to the simplest and commonly utilized one, known as the *squared exponential* function, defined as

$$k(\mathbf{x}, \mathbf{x}') = \alpha \exp\left(-\sum_{i=1}^{d} \frac{|x_i - x_i'|^2}{\theta_i}\right) \tag{3.8}$$

for any $\mathbf{x} = (x_1, \ldots, x_d), \mathbf{x}' = (x'_1, \ldots, x'_d)$, and $\alpha, \theta_1, \ldots, \theta_d > 0$. Ignoring the parameters for a moment gives a nice interpretation of this function, namely that it is close to 1 for nearby (similar) points, and that it tends to 0 for very distant (dissimilar) points. In this spirit, parameter α can be then interpreted as the magnitude of correlation and parameters $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_d\}$ as how the correlation changes in each of d dimensions.

Given the challenging nature of manually selecting these parameters, it is more common to derive them from the available data. Until now, the employment of a covariance matrix assumes fixed parameters. However, considering them as random variables, the process of estimating the optimal parameter set involves maximizing the probability of these parameters given the observed data.

$$p(\alpha, \boldsymbol{\theta} \mid \mathbf{f}) = \frac{p(\mathbf{f} \mid \alpha, \boldsymbol{\theta}) p(\alpha, \boldsymbol{\theta})}{p(\mathbf{f})}. \tag{3.9}$$

Since the denominator in the expression (3.9) does not depend on the parameters, it can be considered as a constant. Moreover, in the absence of a strong prior $p(\alpha, \boldsymbol{\theta})$ on the distribution of parameters [4], it is simply observed that

$$p(\alpha, \boldsymbol{\theta} \mid \mathbf{f}) \propto p(\mathbf{f} \mid \alpha, \boldsymbol{\theta}).$$

The term on the right-hand side of this proportion is referred to as the likelihood, denoted $\mathcal{L}(\alpha, \boldsymbol{\theta} \mid \mathbf{f})$, and it is interpreted as how well the data is described by the given parameters.

By the Gaussian process assumption, the likelihood $\mathcal{L}$ follows a joint normal distribution, which has in fact already been introduced as $p(f)$ in (3.4), but with the assumption of known parameters. Extending now the notation to $\mathbf{K}_{\alpha,\theta}$ to emphasize the dependence of the covariance matrix on the parameters, the likelihood is present

$$\mathcal{L}(\alpha, \boldsymbol{\theta} \mid \mathbf{f}) = \frac{1}{(2\pi)^{n/2}} \left|\mathbf{K}_{\alpha,\boldsymbol{\theta}}\right|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{f} - \mathbf{m})^{\mathrm{T}} \mathbf{K}_{\alpha,\boldsymbol{\theta}}^{-1} (\mathbf{f} - \mathbf{m})\right).$$

For a number of computational advantages one often maximizes the log-likelihood

$$\ln \mathcal{L} = -\frac{n}{2} \ln 2\pi - \frac{1}{2} \ln \left|\mathbf{K}_{\alpha,\theta}\right| - \frac{1}{2}(\mathbf{f} - \mathbf{m})^{\mathrm{T}} \mathbf{K}_{\alpha,\theta}^{-1} (\mathbf{f} - \mathbf{m}),$$

which is normally done using typical gradient-based methods. Finally, for the squared exponential to be a valid covariance function, the corresponding covariance matrix, by its definition, must be symmetric and positive semidefinite. It is easy to see that the symmetry condition is satisfied since $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$, but

showing positive semidefinitness is somewhat more complicated. The line of reasoning motivating this property is given in (C. M. Bishop, 2006). Hence, the squared exponential function does give rise to a symmetric and positive semidefinite matrix, and since every such matrix is a covariance matrix (A. N. Shiryayev, 1984), it is established that the squared exponential function is indeed a valid covariance function.

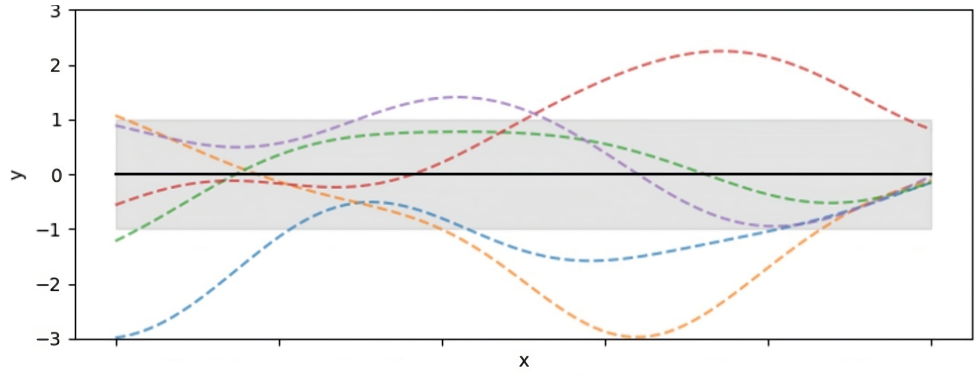

Figure 3.2: Prior with Radial basis function (RBF) kernel. Source: `https://scikit-learn.org/stable/modules/gaussian_process.html`

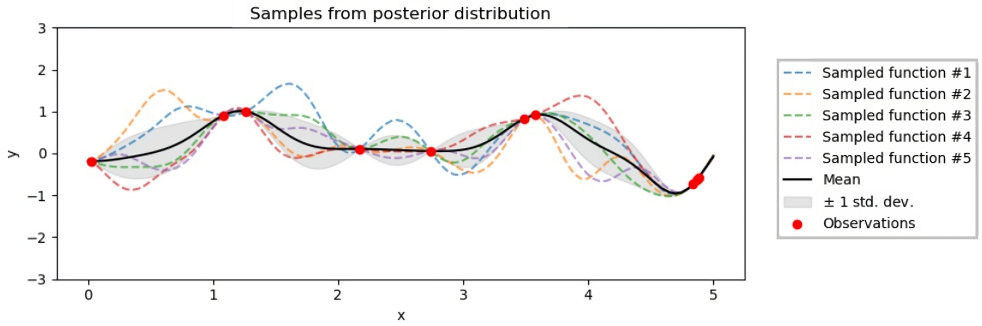

Figure 3.3: Posterior with Radial basis function (RBF) kernel. Source: `https://scikit-learn.org/stable/modules/gaussian_process.html`

In conclusion, Gaussian processes emerge as a powerful tool in modeling probabilistic distributions over functions. The incorporation of a prior in the context of a Gaussian process allows for the integration of prior knowledge about the structure of the objective function. Subsequently, through the application of Bayes' theorem, it is obtained an updated probability distribution, the posterior, reflecting our knowledge combined with new observed data. It is evident from the figures 3.2 and 3.3 that, by utilizing the obtained results, the potential trajectories precisely pass through those points. Furthermore, leveraging the concept of similarity, there is reduced uncertainty in the vicinity of those points This flexibility in handling uncertainties and updating predictions makes Gaussian processes a valuable tool in various applications, ranging from regression to Bayesian optimization.

3.1.2 Acquisition function

A function $a : \mathcal{X} \to \mathbb{R}$ used to determine which point in the set $\mathcal{X}$ is the next one to evaluate the true objective function at is called the acquisition function. The intuition behind it is that the high values of acquisition function correspond to high values of the true objective. Mathematically, the formulation of this is posed as an optimization problem subject to boundary constraints on the variables, namely

$$\mathbf{x}_{\text{next}} = \arg\max_{x \in \mathcal{X}} a(\mathbf{x}).$$

The essence of Bayesian optimization is to utilize the fact that it is relatively easy to optimize $a(\mathbf{x})$ since its evaluation at any point depends only on the probabilistic model which is cheaper to compute than evaluating the costly objective function.

In this probabilistic model, the posterior distribution derived in (3.1.1) is used as a surrogate for the value of the objective at a certain point. Throughout this section, it is assumed that $f(\mathbf{x})$ for a given $\mathbf{x} \in \mathcal{X}$ is a normal random variable having the mean $\mu(\mathbf{x})$ and the variance $\sigma^2(\mathbf{x})$ as defined in equations 3.6 and 3.7 , respectively. Furthermore, the maximum value of the true objective function observed so far will be utilized, denoted as$f(\mathbf{x}^+)$.

There are three most common types of acquisition function used in the literature: **Probability of Improvement, Expected improvement, Upper or Lower confidence bound**. The key property that an acquisition function has to satisfy is a trade-off between exploitation and exploration, where exploitation concerns to target the area providing more chance to improve namely preferring the points with higher posterior mean, while exploration means to move toward less explored regions of the search space namely preferring points with higher posterior variance. In this dissertation it is only analyzed the UCB.

Upper Confidence Bound or Lower Confidence Bound. Upper and Lower Confidence Bound (UCB and LCB) are used, respectively for maximization and minimization problems is an acquisition function that manages exploration-exploitation by being optimistic in the face of uncertainty, in the sense of considering the best-case scenario for a given probability value (Auer, 2002).
For the case of minimization, LCB is given by:

$$\text{LCB}(x) = \mu(x) - \xi\sigma(x)$$

where $\xi \geq 0$ is the parameter in charge of managing the trade-off between exploration and exploitation ($\xi = 0$ is for pure exploitation; on the contrary, higher values of ξ emphasize exploration by inflating the model uncertainty).

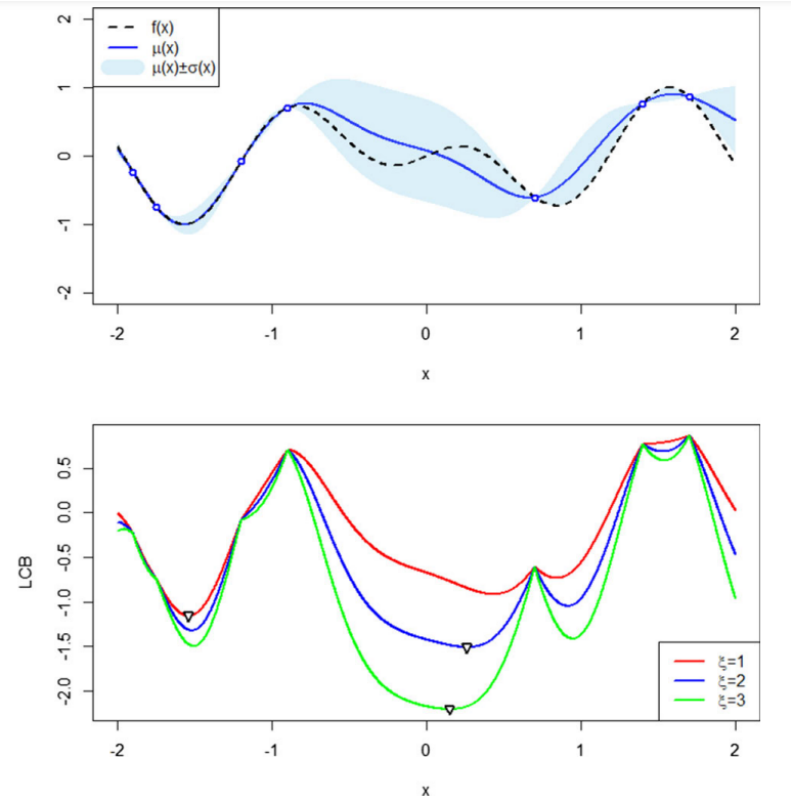

Figure 3.4: GP trained depending on seven observations (top), LCB with respect to different values of ξ and min values corresponding to the next point to evaluate (bottom). Source: (A. Candelieri et al., 2021)

Figure 3.4 shows how the selected points change depending on ξ.

Finally, the next point to evaluate is chosen according to $x^{(n+1)} = \underset{x \in X}{\text{argmin}} LCB(x)$, in the case of a minimization problem, or $x^{(n+1)} = \underset{x \in X}{\text{argmax}} UCB(x)$ in the case of a maximization problem.

From the perspective of BO a particularly interesting bandit problem is the kernelized continuum armed banditproblem (Srinivas et al. 2010). Here, f is assumed to be in the closure of functions on X expressible as a linear combination of a feature embedding parametrized by a kernel k.

The optimization of the acquisition function leads to the next location to be queried, $x^{(n+1)}$, and, consequently, to a sequence of locations generated $\{x^{(1)}, \ldots, x^{(N)}\}$ over the BO process, with N the overall number of function evaluations at the end of the process.

Expected Improvement (EI) PI considers only the probability of improving the current best estimate, but it does not factor in the magnitude of the improvement. This is where the expected improvement acquisition function is different. Instead of looking at the improvement $I(x)$, which is a random variable, it is possible to

calculate the "Expected Improvement", which is the expected value of $I(x)$:

$$\mathrm{EI}(x) \equiv \mathbb{E}[I(x)] = \int_{-\infty}^{\infty} I(x)\varphi(z)\mathrm{d}z$$

Where $\varphi(z)$ is the probability density function of the normal distribution $\mathcal{N}(0,1)$, i.e., $\varphi(z) = \frac{1}{\sqrt{2x}} \exp\left(-z^2/2\right)$.

$$\mathrm{EI}(x) = \int_{-\infty}^{\infty} I(x)\varphi(z)\mathrm{d}z = \int_{-\infty}^{\infty} \underbrace{\max\left(f(x) - f\left(x^{\star}\right), 0\right)}_{I(x)} \varphi(z)\mathrm{d}z$$

To calculate this integral the max operator needs to be eliminated. To achieve this, the integral will be broken up into two components, one where $f(x) - f\left(x^{*}\right)$ is positive and one where it is negative. The point where the switch happens is given by:

$$f(x) = f\left(x^{\star}\right) \Rightarrow \mu + \sigma z = f\left(x^{\star}\right) \Rightarrow z = \frac{f\left(x^{\star}\right) - \mu}{\sigma}$$

Let's call this point $z_0 = \frac{f(x^*) - \mu}{\sigma}$, and break up the integral as:

$$\mathrm{EI}(x) = \underbrace{\int_{-\infty}^{z_0} I(x)\varphi(z)\mathrm{d}z}_{\text{Zero stace } I(z)-0} + \int_{z_0}^{\infty} I(x)\varphi(z)\mathrm{d}z$$

In this way there is:

$$\begin{aligned}
\mathrm{EI}(x) &= \int_{z_0}^{\infty} \max\left(f(x) - f\left(x^{\star}\right), 0\right)\varphi(z)\mathrm{d}z = \int_{z_0}^{\infty} \left(\mu + \sigma z - f\left(x^{\star}\right)\right)\varphi(z)\mathrm{d}z \\
&= \int_{z_0}^{\infty} \left(\mu - f\left(x^{\star}\right)\right)\varphi(z)\mathrm{d}z + \int_{z_0}^{\infty} \sigma z \frac{1}{\sqrt{2\pi}} e^{-z^2/2}\,\mathrm{d}z \\
&= \left(\mu - f\left(x^{\star}\right)\right) \underbrace{\int_{z_0}^{\infty} \varphi(z)\mathrm{d}z}_{1-\Phi(z_0)=1-\mathrm{CDF}(z_0)} + \frac{\sigma}{\sqrt{2\pi}} \int_{z_0}^{\infty} z e^{-z^2/2}\,\mathrm{d}z \\
&= \left(\mu - f\left(x^{\star}\right)\right)\left(1 - \Phi\left(z_0\right)\right) - \frac{\sigma}{\sqrt{2\pi}} \int_{z_0}^{\infty} \left(e^{-z^2/2}\right)'\mathrm{d}z \\
&= \left(\mu - f\left(x^{\star}\right)\right)\left(1 - \Phi\left(z_0\right)\right) - \frac{\sigma}{\sqrt{2\pi}} \left[e^{-z^2/2}\right]_{z_0}^{\infty} \\
&= \left(\mu - f\left(x^{\star}\right)\right) \underbrace{\sigma}_{\frac{\Phi(-z_0)}{(1-\Phi(z_0))} + \sigma\varphi(z_0)}) + \sigma\varphi\left(\frac{\mu - f\left(x^{\star}\right)}{\sigma}\right)
\end{aligned}$$

At the last point, the fact that the Probability Density Function of a normal distribution is symmetric was utilized, therefore $\phi\left(z_0\right) = \phi\left(-z_0\right)$. Alright, so this equation might seem intimidating, but it's really not. $\mathrm{EI}(x)$ take high values When $\mu > f\left(x^{\star}\right)$. L.e, then mean value of the Gaussian Process is high at x. Expected

improvement is also increased when there's lots of uncertainty, therefore when $\sigma > 1$. By the way, the formula above works for $\sigma(x) > 0$, otherwise, if $\sigma(x) = 0$ (as it happens at the observed data points), it holds that $\mathrm{EI}(x) = 0$.

At the end by injecting a (hyper)parameter ξ into the formula for $\mathrm{EI}(x)$, it is possible to fine tune how much exploitation vs. how much exploration the BO algorithm will do. So, the full formula is:

$$\mathrm{EI}(x;\xi) = (\mu - f(x^\star) - \xi)\,\Phi\left(\frac{\mu - f(x^\star) - \xi}{\sigma}\right) + \sigma\varphi\left(\frac{\mu - f(x^\star) - \xi}{\sigma}\right)$$

For $\xi = 0$, the outcome aligns with the previous formula. Nonetheless, in the case of large values of ξ one may consider it as pretending to possess a larger current best value than it actually is! This, in turn, guides the Bayesian Optimization algorithm toward increased exploration.

3.1.3 Bayesian optimization algorithm

Now, all the essential elements are in place to formally present a comprehensive Bayesian optimization algorithm. In summary, the overarching concept is to transform the challenge of optimizing a costly and potentially non-analytically available objective function into a series of optimizations of the acquisition function. This acquisition function is less expensive to evaluate and is often available in a closed-form. The most effective approach to outline the general Bayesian optimization algorithm is through the following pseudocode.

Algorithm 1 Bayesian optimization algorithm

1: Generate an initial sample of points $\mathbf{X}_0$ and associated objective function evaluations $\mathbf{f}_0$.
2: **while** some stopping criterion is not fulfilled **do**
3: Maximize the acquistion function $a(\mathbf{x})$ using the posterior mean and variance under the Gaussian process assumption:

$$\mathbf{x}_{i+1} = \arg\max_{x \in \mathcal{X}} a(\mathbf{x}).$$

4: Evaluate the objective function at this point and put $f_{i+1} = f(\mathbf{x}_{i+1})$. ▷ The costly step.
5: Augment the observations as

$$\mathbf{X}_{i+1} = \mathbf{X}_i \cup \{\mathbf{x}_{i+1}\} \;\;,\;\; \mathbf{f}_{i+1} = \mathbf{f}_i \cup \{\mathbf{f}_{i+1}\}.$$

6: **end while**
7: Choose the maximum $f_{\max}$ to be the maximum value in the final $\mathbf{f}$.

3.1.4 Grid, Random or Bayesian Search?

Previously, it has been stated that when dealing with algorithms that involve numerous parameters, attempting all conceivable combinations to identify the optimal set

becomes a challenging task. Consequently, there is a desire to conduct hyperparameter tuning in an efficient and manageable manner. The intent of this paragraph is to highlight the benefits of Bayesian optimization compared to grid search or random search.

Grid Search Grid search is a technique used in machine learning to find the best hyperparameters for a model. It involves defining a grid of values for each hyperparameter and testing all possible combinations of values. Grid that is identified beforehand from the analyst and does not change during the iterations.

Random Search In contrast to Grid Search, Randomized Search explores only a subset of parameter values. These values are randomly sampled from a specified list or distribution.When the parameters are provided as a list, sampling without replacement occurs (similar to grid search). However, if the parameter is specified as a distribution, sampling with replacement is recommended.

Bayesian Search The primary distinction between Bayesian search and other methods lies in the fact that the tuning algorithm refines its parameter selection in each round based on the score from the previous round. Instead of selecting the next set of parameters randomly, the algorithm optimizes the choice, potentially reaching the optimal parameter set more quickly than the previous two methods. Essentially, this method focuses solely on the pertinent search space, discarding ranges that are less likely to yield the best solution. As a result, it proves advantageous in scenarios with substantial amounts of data, slow learning processes, and a desire to minimize tuning time.

From the analysis conducted on the learning rate hyperparameter of an XGBoost (Gorodetski.,2021) algorithm, emerges the effectiveness of Bayesian optimization. The best learning rate parameter in this comparison is 0.008 (found by the bayesian search).

It is clear that the bayesian search outperforms the other methods by a little. This effect is much more noticeable in larger datasets and more complex models. There is no doubt that is the smartest way to find the best combination of hyperparameters compliance with the computazional efficiency.

3.2 Multiple information source optimization

Continuing the pursuit of efficient function optimization, accounting for both computational budget and time (an excellent proxy for CO2 consumption), the scenario

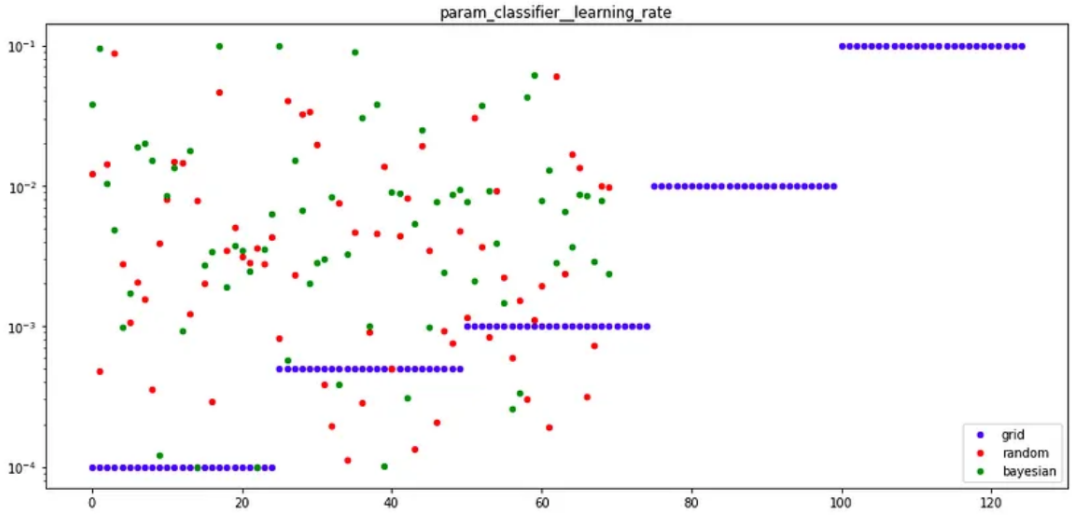

Figure 3.5: Visualization of parameter search, learning rate. Source: (Gorodetski., 2021)

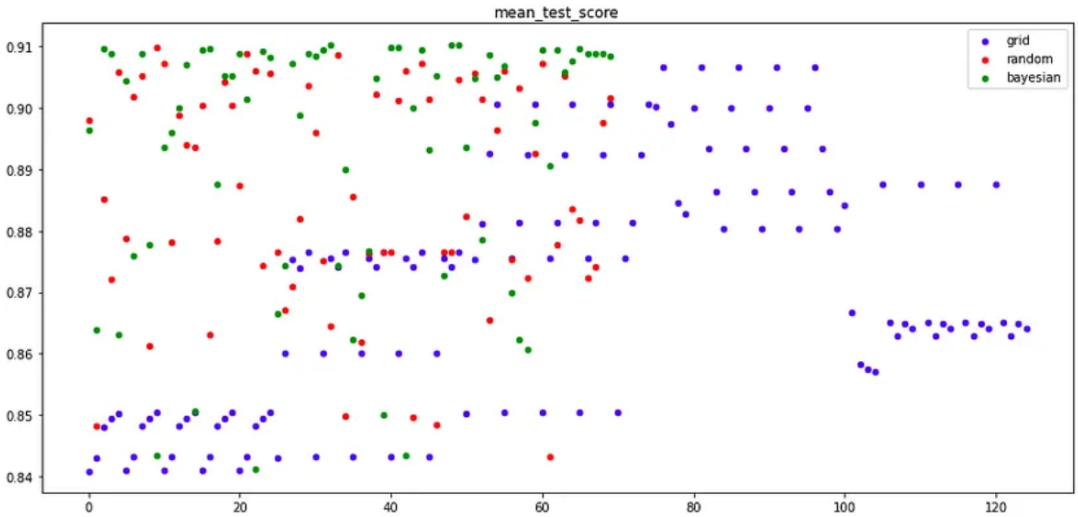

Figure 3.6: Visualization of the mean score for each iteration. Source: (Gorodetski, 2021)

outlined in Section 3.1 is expanded by introducing the concept of multiple information sources. Multiple Information Source Optimization (MISO) is designed to explore the global optimum of a costly, multi-peaked function, known as the ground-truth, which is a black-box function. MISO allows the inclusion of less expensive information sources that serve as approximations to the ground-truth. The primary objective is to identify an optimal solution for the ground-truth, meeting constraints on the accumulated query cost during the search process. MISO is tailored for single-objective problems, where the subscript is employed to specify a particular

information source, where $f_1(\mathbf{x})$ is the groundtruth and $f_s(\mathbf{x})$, with $s \in \{2, \ldots, S\}$, are the cheap information sources. The MISO problem can be formulated as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} f_1(\mathbf{x}) \tag{3.10}$$

$$\text{subject to: } \sum_{(s,\mathbf{x})\in Z^{1:n}} c_s \leq C_{\max} \tag{3.11}$$

where $Z^{1:n} = \left\{\left(s^{(i)}, \mathbf{x}^{(i)}\right)\right\}_{i=1:n}$ denotes the set of source-location pairs sequentially queried, c_s is the cost for querying $f_s(\mathbf{x})$, and $C_{\max}$ is the maximum query cost that can be cumulated along the optimization process. Bayesian Optimization (BO) has been effectively expanded to address Multiple Information Source Optimization (MISO) challenges. In this context, each information source is independently represented by a probabilistic surrogate model, typically a Gaussian Process (GP), constructed based on the queries conducted on that specific source. Subsequently, these individual models are amalgamated into a unified model, guiding the selection of the next promising source-location pair to query. This approach has been utilized in works such as those by Ghoreishi and Allaire (2019) and Candelieri and Archetti (2021b). MISO has been gaining increasing attention in the last years.

In the literature, various approaches have been explored to leverage information from different information sources:

- Hyperparameter Optimization (): The use of small datasets to quickly optimize the hyperparameters of machine learning models on larger datasets. The knowledge gained from previous optimizations is transferred to new tasks to expedite k-fold cross-validation.

- MF-GP-UCB (Multi-Fidelity Gaussian Process Upper Confidence Bound): A multifidelity bandit optimization that utilizes Gaussian Process approximations of all sources. It explores the search space using lower fidelity sources first and then higher fidelity sources in successively smaller regions.

- FABOLAS (FAst Bayesian Optimization on LArge dataSets): An approach for HPO on large datasets that selects hyperparameter configurations and dataset sizes to optimize hyperparameters for the entire dataset. It often provides good solutions faster than traditional BO-based HPO on the full dataset.

- GP with Kernel on Combined Space: The use of a Gaussian Process with a kernel operating on a space consisting of both the search space (hyperparameters

to optimize) and information sources.

- IBO (Importance-based Bayesian Optimization): Modeling a distribution over the location of the optimal hyperparameter configuration and allocating experimental budget based on cost-adjusted expected reduction in entropy. Higher fidelity observations provide a larger reduction in entropy, albeit at a higher evaluation cost.

- MISO-AGP (multi information source optimization augmented GP:the idea is constructing a Gaussian Process (GP) by training it exclusively on a subset of "reliable" function evaluations gathered from all the information sources available. This GP is denoted as the Augmented Gaussian Process (AGP), leading to the nomenclature MISO-AGP for our approach. The term "augmented" emphasizes that the set of function evaluations used to train the AGP originates from those conducted on the most resource-intensive source and is then expanded by incorporating evaluations from additional sources.

In this dissertation, only this latter approach is analyzed

3.2.1 Augmented Gaussian process (MISO-AGP)

It is introduce, immediately some helpful notations.

Let $D_s = \left\{ \left(x^{(i)}, y_s^{(i)} \right) \right\}_{i=1,\dots,n_s}$ denote the n_s function evaluations performed so far on the source $\int$. For each source s a specific GP, $\mathcal{G}_s$, is trained on the current D_s. Let us introduce a model discrepancy measure, $\eta\left(x, \mathcal{G}, \mathcal{G}'\right)$, between two GPs. Differently from other papers, such as Poloczek et al. (2017) and Ghoreishi et al. (2019), It is computed simply as:

$$\eta\left(x, \mathcal{G}, \mathcal{G}'\right) = |\mu(x)| D_s - \mu'(x)\,|D_{s'}| \tag{3.12}$$

with $\mu(x)$ and $\mu'(x)$ the conditioned mean functions of the two GPs. It is also important to note that $\eta\left(x, \mathcal{G}, \mathcal{G}'\right)$ depends on x. Indeed, in the context of Multiple Information Sources Optimization (MISO), the fidelity of each source is not known a priori, and it may not be constant over $\mathcal{X}$.

Assume that $f(x)$ can be queried at the highest cost, that is $f(x) = f_1(x)$. Thus, the set of evaluations to train the AGP consists of D_1 "augmented" by:

$$\tilde{D} = \{(\tilde{x}, \tilde{y}) : \exists \ddagger : (\tilde{x}, \tilde{y}) \in D_{\ddagger} \eta\left(x, \mathcal{G}_1, \mathcal{G}\right) < m\sigma_1(x)\} \tag{3.13}$$

with m a technical parameter of the MISO-AGP algorithm. It was employed with $m = 1$ (i.e., around 68% of observations normally distributed are in the interval mean $\pm$ standard deviation). Thus, function evaluations on cheaper sources, having a discrepancy lower than the threshold given in (3.13), are considered "reliable" to be merged with those collected on the most expensive source. Let $\widehat{D}$ denote the augmented set of function evaluations, such that $\widehat{D} = D_1 \cup \tilde{\mathrm{D}}$, the AGP $\widehat{\mathcal{G}}$ is trained on $\widehat{D}$, leading to $\widehat{\mu}(x)$ and $\widehat{\sigma}(x)$. An example is reported in Fig. 3.7.

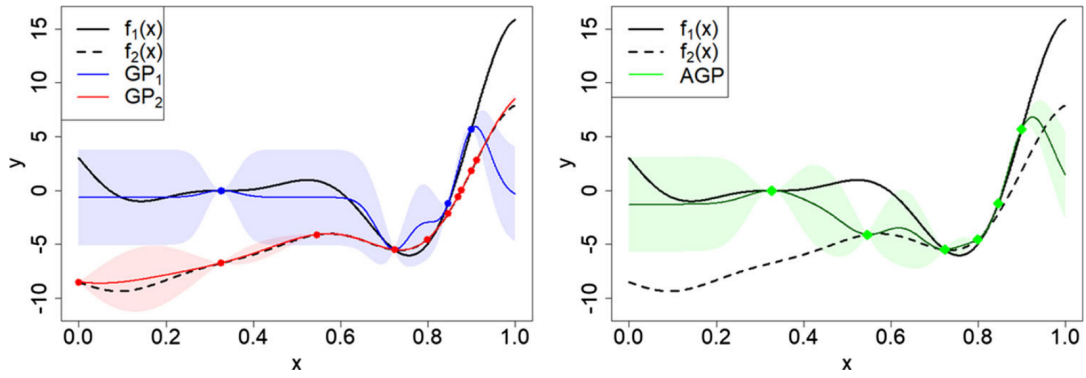

Figure 3.7: A demonstration of the Augmented Gaussian Process (AGP) is illustrated in the context of a one-dimensional MISO minimization problem involving two information sources. On the left side, there are two separate Gaussian Processes trained on each source, while on the right side, the AGP is depicted. In the AGP scenario, only three evaluations from the more cost-effective source (located around x = 0.5, x = 0.7, and x = 0.8) are strategically chosen to "augment" the evaluations obtained from the expensive source. This simultaneous reduction in uncertainty near the global minimum of f_1 is achieved alongside a decrease in the number of evaluations required to train the AGP (specifically, six out of the 14 overall evaluations). Source: (A. Candelieri et al., 2021).

Acquisition function in MISO-AGP algorithm

Following the training of the AGP, an acquisition function must be used to choose the next pair source-location to query, that is (s', x'). Considering the framework of U/ LCB:

$$(s', x') = \mathrm{tar}^{\underset{s=1,\dots,S}{\mathrm{argmax}}}_{x \in X \subset \mathbb{R}^d} \left\{ \frac{y^+ - \left(\widehat{\mu}(x) - \sqrt{\beta^{(n)}} \widehat{\sigma}(x) \right)}{c_s \left(1 + \eta \left(x, \widehat{\mathcal{G}}, \mathcal{G}_s \right) \right)} \right\}$$

where n is the number of function evaluations into $\widehat{D}$ and $y^+ = \min_{(x,y) \in \widehat{D}} \{y\}$ is the best observed value into $\widehat{D}$. The numerator is the most optimistic improvement with respect to the AGP's LCB, penalized by the cost of the source $\int$ and the model discrepancy between the AGP $\widehat{\mathcal{G}}$ and $\mathcal{G}_s$, at the location x.

There is the chance that x' could be too close to some previous function evalua-

tions on s'. This behaviour arises when BO is converging to a (local/global) optimum and leads to a well-known instability issue in GP training, which is ill-conditioning in the inversion of the matrix $[\mathbf{K} + \lambda^2\mathbf{I}]$. This instability issue occurs even more frequently and quickly in the noise free setting (i.e., $\lambda = 0$). To mitigate the undesired behavior that might lead to the wasteful use of evaluations without achieving improvement and/or the risk of encountering instability issues, the following correction is introduced.

$$\begin{aligned} &\text{Given} \quad (s', x') \quad \text{from} \\ &\exists \left(x^{(i)}, y^{(i)}\right) \in D_{s'} x' - x^{(i)2} < \delta \\ &s' \leftarrow 1 \text{ and } x' = \underset{x \in \mathcal{X} \subset \mathbb{R}^d}{\text{argmax}} \sigma_1(x) \end{aligned} \tag{3.14}$$

if with $\delta > 0$ the second MISO-AGP's technical parameter. In essence, the acceptable level of approximation, δ, is defined for locating the optimizer. In situations where x' is closer than δ to another evaluation on s', a preference is given to "allocating the budget" towards reducing uncertainty on the most expensive source.

The MISO-AGP algorithm is summarized in the following.

In conclusion, the experiment conducted using Support Vector Machine (Figure 3.8) (Candelieri et al.) indicates that the optimization achieved through the Augmented Gaussian Process outperforms both in terms of performance and computational efficiency compared to the one obtained using a single source. MISO-AGP has intelligently exploited the cheaper information source, thanks to the proposed AGP, leading to an energy-efficient and green HPO task.

3.3 Multi-objective optimization

In recent research, the focus has shifted towards multi-objective optimization (MO) in the context of Fairness-aware AutoML. This approach aims to minimize both misclassification error and a defined unfairness metric simultaneously (Schmucker et al., 2020). Fairness-aware AutoML, as termed by Weerts et al. (2023), involves the automation of the design of a Machine Learning (ML) pipeline, with Hyperparameter Optimization () of ML algorithms being a specific task within this framework (Hutter et al., 2019; He et al., 2021).

Fairness-constrained HPO faces challenges in real-life settings due to the difficulty in establishing a suitable threshold on fairness a-priori. In response to this consideration and recent trends in the research field (Nguyen et al., 2023), a decision has been made to concentrate on the multi-objective HPO approach.

Algorithm: MISO-AGP

Input:

$f_1(x), \ldots, f_S(x)$; $c_1, \ldots, c_S$; $\mathcal{X}$; max cumulated cost $\bar{C}$; max iterations N;
set m and δ (MISO-AGP's technical parameters)

Initialization:

$D_s = \left\{\left(x^{(i)}, y_s^{(i)}\right)\right\}_{i=1,\ldots,n_s}$ $\forall\, s = 1, \ldots, S$ and with n_s initial evaluations on locations randomly sampled in $\mathcal{X}$

Main:

$c \leftarrow 0$; $n \leftarrow 0$;
while ($c < \bar{C}$ AND $n < N$) **do**
  train $\mathcal{G}_s$ on D_s $\forall\, s = 1, \ldots, S \Rightarrow \mu_s(x), \sigma_s(x)$
  build $\widehat{D} = D_1 \cup \widetilde{D}$ with $\widetilde{D}$ defined in (9)
  train the AGP $\hat{\mathcal{G}}$ on $\widehat{D} \Rightarrow \hat{\mu}(x), \hat{\sigma}(x)$
  choose (s', x') according to (10)
  if $\exists\, x^{(i)}: \left(x^{(i)}, y^{(i)}\right) \in D_{s'} \wedge \left\|x' - x^{(i)}\right\|^2 < \delta$ **then**
    (s', x') according to (11)
  endif
  query source s' at location x' and observe $y_{s'}$
  update $D_{s'} \leftarrow D_{s'} \cup \{(x', y_{s'})\}$
  $c \leftarrow c + c_{s'}$; $n \leftarrow n + 1$
endwhile

Output:

build $\widehat{D} = D_1 \cup \widetilde{D}$ with $\widetilde{D}$ defined in (9)
return $(x^+, y^+) \in \widehat{D}$: $y^+ = \min_{(x,y)\in\widehat{D}} \{y\}$

Multi-objective optimization involves solving problems with more than one objective function to be optimized simultaneously. That is:

$$\min_{\mathbf{x}\in\Omega} \mathbf{f}(\mathbf{x}) \tag{3.15}$$

In multi-objective optimization (MO), the search space Ω is typically box-bounded in $\mathbb{R}^d$, and $\mathbf{f} : \Omega \to \mathbb{R}^M$ represents the vector-valued function of the multiple objectives. Due to the conflicting nature of these objectives in MO, there isn't a single solution $\mathbf{x}^* \in \Omega$ to problem 3.15. Instead, the primary objective is to identify a set of equally efficient trade-offs among the objectives. This set of efficient trade-offs is illustrated within the space defined by the M conflicting objectives, giving rise to the Pareto front (also known as the Pareto frontier or boundary). The corresponding set of solutions within the search space Ω is referred to as the Pareto set. An example is provided in Figure 3.9.

Formally, the Pareto set comprises only dominant (or non-dominated) solutions, where a solution $\mathbf{x}$ is considered dominant over another solution $\mathbf{x}'$ if their respective

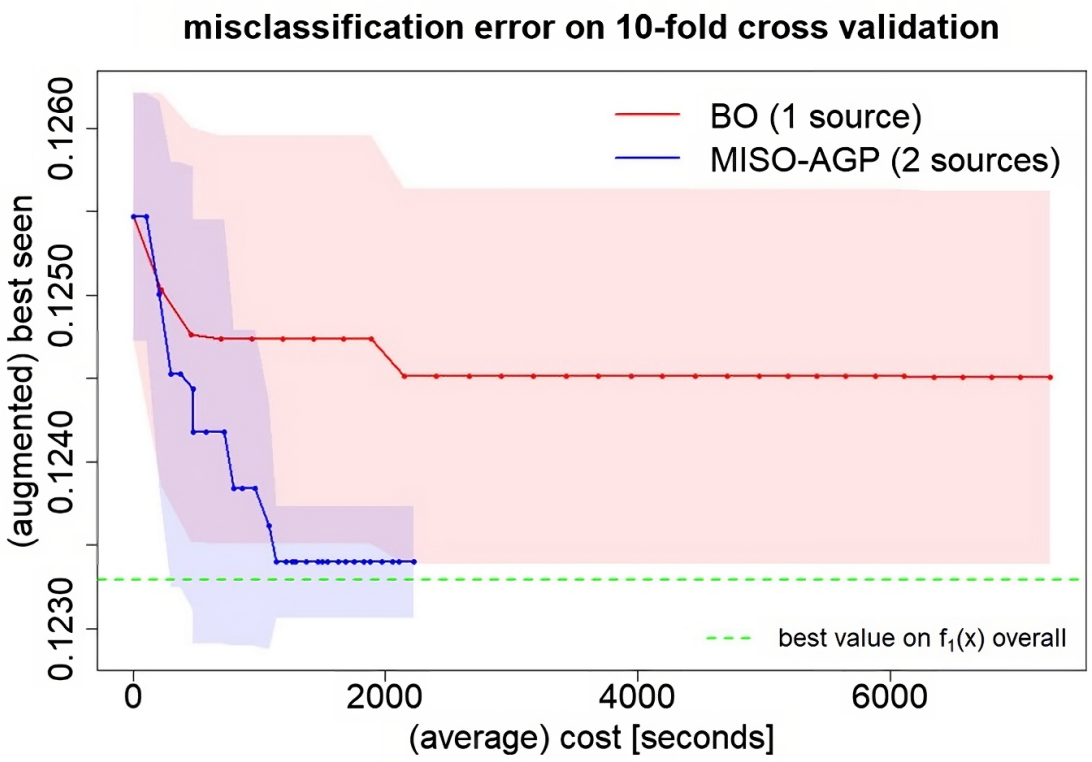

Figure 3.8: HPO of C-SVC on the MAGIC dataset. Comparison between traditional BO-based HPO and MISO-AGP on two information sources. Results refer to ten independent runs. Source: (Candelieri et al., 2021)

objectives, denoted as $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))$ and $\mathbf{f}(\mathbf{x}') = (f_1(\mathbf{x}'), \ldots, f_M(\mathbf{x}'))$, satisfy the following two conditions:

$$\begin{aligned} &f_m(\mathbf{x}) \leq f_m(\mathbf{x}') \, \forall m \in \{1, \ldots, M\} \\ &\exists j \in \{1, \ldots, M\} : f_j(\mathbf{x}) < f_j(\mathbf{x}') \end{aligned} \tag{3.16}$$

Equation 3.16 means that $\mathbf{x}$ is not worse than $\mathbf{x}'$ in all the objectives, and equation 3.17 means that $\mathbf{x}$ is strictly better than $\mathbf{x}'$ in at least an objective. The Pareto dominance symbol, $\prec$, is used to synthesize (3.16,3.17): $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}')$.

If the objectives are black-box, their values can only be known point-wise by querying $\mathbf{f}(\mathbf{x})$ at specific locations. Given all the queries performed so far, the set of non-dominated solutions (respectively, outcomes) is the current approximation of the Pareto set (respectively, front). If the objectives are also expensive to evaluate, in terms of time or resources, then (3.15) must be solved efficiently, meaning that a good Pareto front/set approximation has to be found within a limited number of queries. Thus, sample-efficiency of BO was the driver of its successful extension to the MO setting (i.e., MOBO), mainly along three different strategies:

- **Scalarization** which maps the vector of all objectives into a scalar parametrized

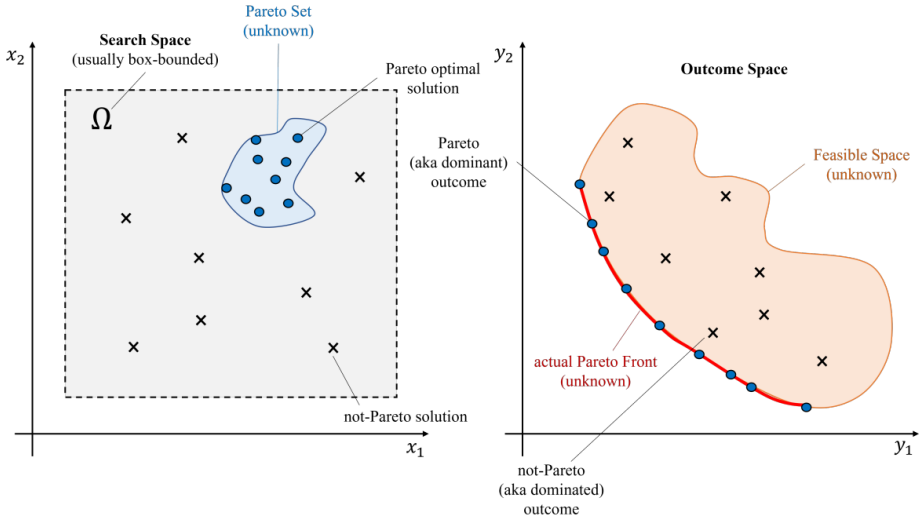

Figure 3.9: On the left: box-bounded Search Space Ω, (unknown) Pareto Set, Pareto and not Pareto solutions. On the right: (unknown) Feasible Space within the Outcome Space (i.e., consisting of all the outcomes associated to solutions in the Search Space), Pareto (aka dominant) outcomes, not-Pareto (aka dominated outcomes), and actual (unknown) Pareto Front.

function whose optimizer, computed by a single objective method, employed by Paria et al. (2020) and Zhang and Golovin (2020), allows the exploration of the entire Pareto set as the parameters vary. However, a drawback is its failure to consider the geometry of the Pareto front approximation.

- Another strategy involves maximizing an index related to the quality of the Pareto front approximation, with the **dominated hypervolume indicator** (HV) being a common choice. The HV measures the volume of the region dominated by a Pareto front approximation.

- An information theoretic approach focuses on reducing uncertainty/entropy about the Pareto front, as seen in the works of Belakaria and Deshwal (2019), Suzuki et al. (2020), and Belakaria et al. (2020), the latter also considering the multi-fidelity setting.

It is crucial to note that, unlike "vanilla" BO, where a probabilistic surrogate model is used to approximate the black-box objective function, most multi-objective Bayesian optimization (MOBO) approaches in the literature adopt a probabilistic surrogate model for each objective, assuming independence. This assumption is reasonable, given that in MO, objectives are expected to be competing and uncorrelated. Similar to BO, every new query contributes to a better approximation of the vector-valued objective function through the update of the probabilistic surrogate model. The acquisition function, addressing the exploration-exploitation dilemma, remains a key component of MOBO, with "vanilla" BO acquisition functions ap-

plicable to scalarization. However, Expected Hypervolume Improvement (EHVI) is a specific acquisition function designed for vector-valued MOBO, extending the concept of Expected Improvement (EI) to the multi-objective setting.

The primary focus of this thesis is on Hyperparameter Optimization (HPO) for a classification model that encompasses two minimization objectives: misclassification error (MCE) and the fairness metric known as Differential Statistical Parity (DSP). The calculation of these objectives involves stratified 10-fold cross-validation (10FCV), characterizing them as black-box, resource-intensive, multi-extremal, and potentially influenced by noise, contingent on the particular machine learning algorithm or cross-validation procedure.

3.3.1 Misclassification error (MCE)

In machine learning, the misclassification rate is a metric that indicates the percentage of observations incorrectly predicted by a classification model. It is calculated as the number of incorrect predictions divided by the total number of predictions.

$$\text{Misclassification Rate} = \frac{\#\text{ Incorrect Predictions}}{\#\text{ Total Predictions}} \tag{3.17}$$

In a binary context where the variable can take only two values (0 or 1), the misclassification rate ranges from 0 to 1. A value of 0 implies a model with zero incorrect predictions, while 1 indicates a model with entirely incorrect predictions. Accuracy, the complement of the misclassification rate, is calculated as

$$\text{Accuracy} = 1 - \text{Misclassification Rate} \tag{3.18}$$

Pros of the misclassification rate include its simplicity and ease of interpretation and calculation. However, a notable drawback is its failure to consider data distribution. For instance, a model predicting that no players get drafted when 90% of players do not get drafted would have a seemingly low misclassification rate of 10%, but it fails to correctly predict any drafted players.

3.3.2 Differential Statistical Parity (DSP)

The issue of fairness metrics, is a sensitive topic. The integration of machine learning techniques and algorithms in various systems and applications significantly impacts our daily lives, ranging from common algorithms used in search, recommendation, and social media to specialized algorithms involved in critical decision-making areas like medicine, criminal justice, and finance. However, the extensive influence of machine learning raises concerns about unintended consequences, particularly in the context of biased decision-making.

To illustrate, a case management and decision support tool, such as the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS), used in the U.S. courts to assess the likelihood of a defendant recommitting a crime, has been found to exhibit bias against African-American defendants. Similarly, facial recognition tools, like those in digital cameras, may overpredict Asians as blinking, indicating unintentional biases in machine learning algorithms. Numerous studies have uncovered examples of artificial intelligence systems introducing biases and systematic discrimination in various applications, including AI chatbots, loan applications, dating apps, flight routing, immigration algorithms, and hiring processes.

To address these issues, fairness in machine learning has become a prominent research field, focusing on the causes of bias and discrimination, mitigation methods, and the evaluation of unfairness. One key aspect of unfairness is statistical parity, which is frequently associated with demographic discrepancies and societal problems. Discrepancies can arise at different stages of the machine learning pipeline, including the training data, learning algorithm, and predictive evaluation metrics.

Statistical bias in the data, for instance, may result from non-representative sampling and measurement errors, categorized into statistical bias and societal bias. Statistical bias involves a systematic mismatch between the sample used for model training and the current world, leading to fairness implications that are often unmeasured by fairness definitions. Societal bias, on the other hand, may persist even with accurate demographic representation in the training data, introducing objectionable social structure data points.

Considering philosophical and psychological points of view, the fight against discrimination has a long history, and in recent years, this issue has also been addressed in machine learning. However, to achieve fairness and eliminate discrimination in decision-making, it is essential to first define the concept of fairness and evaluate it based on well-defined measurements.

In the context of machine learning, an approach based on two families of fairness concepts has been proposed: individual fairness and group fairness. Individual fairness focuses on ensuring fairness at an individual level, while group fairness requires a statistic of a classifier to be equally valid across certain defined protected subgroups. These subgroups are defined by sensitive features, such as gender, ethnicity, age, and political affiliation, and involve distinguishing between privileged and unprivileged groups based on whether the subgroup is favored for positive classification. Building upon the previously introduced concept, an individual instance i in a dataset D consists of a d-dimensional feature vector represented as x_i, its corresponding label y_i, and a feature S representing a sensitive feature defining the subgroup to which an individual belongs. In this context, various fairness definitions have been introduced (S. Verma and J. Rubin, 2018), with a specific focus

on statistical fairness definitions in the binary classification setting. The following presents well-known group and individual fairness definitions, each explained with an example. Throughout these definitions, Y represents the model prediction, Y the true label, and S the sensitive attribute.

Demographic Parity, sometimes referred to as statistical parity, requires that positive predictions are unaffected by the value of the sensitive feature, independently of the actual label:

$$P\left(Y^* = 1 \mid S = 0\right) = P\left(Y^* = 1 \mid S = 1\right)$$

Furthermore, a measure of the violation of this condition, known as Disparate Statistical Parity (DSP), is introduced. The DSP is the absolute value of the difference between these two probabilities:

$$DSP = |P(\hat{Y} = 1 \mid S = 0) - P(\hat{Y} = 1 \mid S = 1)|$$

This concept of DSP provides a quantitative measure of the injustice associated with the lack of statistical parity.

Overall, achieving statistical parity and addressing biases in machine learning is crucial to ensuring fair decision-making that does not discriminate against certain groups based on characteristics such as age, race, gender, or political affiliation.

3.4 Fairness-aware AutoML algorithms

In this paragraph, the features of three algorithms employed to conduct the experiment discussed in Chapter 4 are highlighted. These algorithms belong to the AutoML (HPO) category, where hyperparameter optimization leverages the capabilities of Bayesian optimization.

3.4.1 AutoGluon-FairBO

AutoGluon-FairBO stands as a noteworthy extension of the influential AutoGluon library [1], introduced by Schmucker et al. in 2020, emphasizes hyperparameter optimization in a particular context: multi-objective optimization addressed as a bi-objective task.

At its core, AutoGluon-FairBO relies on Hyperband, a well-recognized algorithm for its efficiency in hyperparameter optimization. Hyperband stands out for its ability to combine random search with successive halving, optimizing the resource

[1] `https://auto.gluon.ai/stable/index.html`

allocation process. AutoGluon-FairBO inherits this powerful foundation but goes further by integrating an "early discarding of unpromising candidates" approach.

This method, also known as "early discarding of unpromising candidates," aims to enhance optimization efficiency by early elimination of less promising hyperparameter configurations. Such an approach is crucial in scenarios with limited resources, thereby reducing energy consumption and CO2 emissions.

The heart of AutoGluon-FairBO's innovation lies in its ability to address multi-objective optimization purposefully. Specifically, optimization is conducted simultaneously to minimize both classification error and unfairness during 10-fold cross-validation. This advanced approach enables not only top-notch performance in terms of accuracy but also ensures fair and just treatment of different data classes or groups.

In summary, AutoGluon-FairBO represents a significant step forward in the field of AutoML, providing efficient hyperparameter optimization that is environmentally conscious and sensitive to fairness in data. This powerful library allows users to train state-of-the-art machine learning models for image classification, object detection, text classification, and tabular data prediction with just a few lines of code, making the model development experience accessible to everyone, regardless of their prior experience in machine learning.

3.4.2 BoTorch-MOMF

BoTorch-MOMF (Multi-Objective and Multi-Fidelity) and FanG-HPO both fall under the umbrella of "multi-fidelity performance measurements" methods, yet they exhibit significant methodological differences in their approaches.

Starting with BoTorch-MOMF, it employs a multi-output Gaussian Process (GP) to model three objectives. These objectives include not only the 10-fold cross-validation (10FCV) Misclassification Error (MCE) and 10FCV Differential Statistical Parity (DSP) but also the query costs associated with different information sources. Additionally, BoTorch-MOMF treats fidelity as an additional decision variable, treating it akin to a hyperparameter of the machine learning model under optimization. Notably, the acquisition function in BoTorch-MOMF is based on Expected Hypervolume Improvement (EHVI), which penalizes values based on the cost of the source, making higher fidelity sources more costly.

In contrast, FanG-HPO utilizes independent Augmented Gaussian Processes (AGP) for each objective. These AGPs are fitted by merging observations from various information sources. FanG-HPO adopts a different approach in terms of acquisition function, still utilizing EHVI. However, it employs a two-step mechanism explained better in Section 3.4.3, which differs from the cost-based penalization used

in BoTorch-MOMF.

Specifically, BoTorch-MOMF is a open source library[2], introduced by Irshad et al. in 2021, implements a generic multi-objective and multi-fidelity Bayesian Optimization (BO) framework. It has been employed to address Hyperparameter Optimization (HPO) as a bi-objective task, aiming to enhance energy efficiency. This is achieved by selectively utilizing the entire dataset (considered a high-fidelity source) or a portion of it (considered a low-fidelity source) to compute the misclassification error and unfairness metric for each hyperparameter configuration. BoTorch-MOMF is a recent addition to the BoTorch suite, demonstrating its applicability in tackling complex optimization problems with multiple objectives and fidelities.

3.4.3 FanG-HPO

FanG-HPO is a methodology that distinguishes itself by utilizing subsets of a large dataset, referred to as information sources. This approach involves obtaining cost-effective approximations of both accuracy and fairness for the Hyperparameter Optimization (HPO) task. Unlike some other methods, such as BoTorch-MOMF, FanG-HPO explicitly exploits two information sources, leveraging diverse data subsets to inform the optimization process.

The method employs an independent Augmented Gaussian Process (AGP) for each objective, fitting them by merging observations from various information sources. The optimization process is driven by multi-objective Bayesian Optimization, allowing FanG-HPO to efficiently identify Pareto-efficient machine learning models. The use of subsets of the large dataset enables the exploration of trade-offs between accuracy and fairness, making it a distinctive approach in the HPO landscape.

The proposed eco-conscious Hyperparameter Optimization (HPO) approach aims to tackle the problem (3.10-3.11), albeit with the scalar objective function substituted by a vector-valued counterpart:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} \mathbf{f_1}(\mathbf{x})$$
$$\text{subject to:} \sum_{(s,\mathbf{x})\in Z^{1:n}} c_s \leq C_{\max}$$

Here, $\mathbf{f_1}$ represents the vector-valued ground-truth, while all additional cost-effective information sources are denoted by $\mathbf{f_s}$, where $s \in \{2, \ldots, S\}$. This adjustment underscores the focus on creating an environmentally conscious optimization framework, where objectives extend beyond a single scalar measure. The objective functions encapsulate various considerations, with $\mathbf{f_1}$ serving as the overarching ground-truth, and $\mathbf{f_s}$ representing other economical information sources. The optimization task

[2] `https://botorch.org/`

adheres to the constraint that the cumulative cost associated with these sources, expressed as $\sum_{(s,\mathbf{x})\in Z^{1:n}} c_s$, does not surpass the predetermined maximum cost limit $C_{\max}$.

In FanG-HPO, both objectives and information sources are modelled independently via GP regression (Williams and Rasmussen, 2006; Gramacy, 2020). A GP is a probabilistic regression model whose predictive mean, $\mu(\mathbf{x})$, and uncertainty, $\sigma(\mathbf{x})$, are conditioned on previous observations. A brief introduction to GP regression is provided in the Paragraph 3.1.

Thus, at a generic iteration, FanG-HPO learns $S \times M$ GP models, leading to the following set of predictive means and uncertainty functions:

$$\{\mu_{sm}(\mathbf{x}), \sigma_{sm}(\mathbf{x})\}_{s=1:S,m=1:M}$$

Subsequently, an individual GP model is constructed for each objective by amalgamating the GPs that independently capture that particular objective across all information sources. In the context of FanG-HPO, this process is executed through the application of the Augmented Gaussian Process (AGP) methodology, as recently introduced by Candelieri and Archetti. To elaborate, a set of indices is calculated to identify "reliable" observations from less expensive sources for each source-objective pair. Here, "reliable" implies that these observations do not exhibit significant discrepancies compared to the ground-truth:

$$\mathcal{I}_{sm} = \left\{ i : \left|\mu_{1m}(\mathbf{x}) - \mu_{sm}\left(\mathbf{x}^{(i)}\right)\right| \leq \alpha\sigma_{1m}\left(\mathbf{x}^{(i)}\right), \mathbf{x}^{(i)} \in \mathbf{X}_s \right\}, \\ \forall s \neq 1, \forall m \in \{1, \ldots, M\}$$

where α is a technical parameter to tune reliability of the observations from cheap information sources. In (Candelieri and Archetti, 2021b) the suggested value is $\alpha = 1$.

Then, the observations on the ground-truth are "augmented" with those identified by $\mathcal{I}_{sm}$, separately for each objective $m \in \{1, \ldots, M\}$:

$$\widehat{\mathbf{X}}_m \leftarrow \mathbf{X}_1 \cup \left\{\mathbf{x}^{(i)} \in \mathbf{X}_s : i \in \mathcal{I}_{sm}, \forall s \neq 1\right\}$$
$$\widehat{\mathbf{Y}}_m \leftarrow \mathbf{Y}_{1[m]} \cup \left\{y^{(i)} \in \mathbf{Y}_{s[m]} : i \in \mathcal{I}_{sm}, \forall s \neq 1\right\}$$

where $\mathbf{X}_s$ are the locations queried on source s and $\widehat{\mathbf{Y}}_{[m]}$ are the values observed for the objective m and associated to the set $\widehat{\mathbf{X}}_m$ (i.e., the symbol $[m]$ is the operator selecting only the column m of the $n_s \times M$ matrix $\mathbf{Y}_{sm}$, with n_s the number of queries performed on the source s).

Finally, FanG-HPO fits M independent AGPs, with predictive means and uncertainty respectively denoted with $\widehat{\mu}_m(\mathbf{x})$ and $\widehat{\sigma}_m(\mathbf{x})$, and both conditioned to

$\left\{\widehat{\mathbf{X}}_m, \widehat{\mathbf{Y}}_m\right\}$.

The subsequent determination of the source-location pair to be queried, denoted as $(s', \mathbf{x}')$, is accomplished by addressing a multi-objective problem. The objectives in question are approximated through the AGPs, as detailed earlier. The adoption of M independent AGPs aligns with recent findings in the literature. As outlined in the study by Zhan et al., employing separate GPs to model each objective independently facilitates the implementation of multi-objective optimization methodologies. Conversely, utilizing dependent GP models, such as multi-output GPs, does not yield significant advantages over independent GPs.

More precisely, $(s', \mathbf{x}')$ is obtained according to the following two-steps procedure:

1. Selecting $\mathbf{x}'$. First, the location $\mathbf{x}'$ is selected depending on the well-known Expected Hypervolume Improvement (EHVI). Hypervolume Improvement (HVI) is defined as the relative increase in the hypervolume indicator, when an outcome $\mathbf{y}$, associated to a solution $\mathbf{x}$, is added to the current Pareto front approximation. In BO, the HVI is a random variable because $\mathbf{y}$ is a (set of) random variable itself, and this leads to the EHVI.

$$\mathbf{x}' = \arg\max_{\mathbf{x}\in\Omega} \mathrm{EHVI}(\mathbf{x}, \mathcal{P}, \mathbf{r})$$

   where $\mathcal{P}$ represents the presently estimated Pareto front, and $\mathbf{r}$ serves as the designated reference point. Within the scope of this research, $\mathbf{r}$ signifies the most unfavorable point, characterized by both and DSP values equating to 1. Despite the existence of a closed formula for , its computational cost is substantial. To address this challenge, the FanG-HPO approach employs a swift computation method proposed by Zhao et al.. This method, an extension to the EHVI computation, is based on the Walking Fish Group (WFG) technique, renowned as one of the fastest algorithms for calculating the hypervolume of a Pareto front approximation.

2. Selecting s'. Then, the information source s' is selected according to both its query cost and its discrepancy with respect to the ground-truth at $\mathbf{x}'$, with respect to all the objectives, that is:

$$s' = \arg\min_{s\in\{1\ldots,S\}} c_s \cdot \sum_{m=1}^{M} \left|\mu_{1m}\left(\mathbf{x}'\right) - \mu_{sm}\left(\mathbf{x}'\right)\right| \tag{3.19}$$

In contrast to recent methodologies that advocate frequent querying of the ground truth, as demonstrated in the work of Khatamsaz et al., FanG-HPO adopts

an adaptive strategy during each iteration. This strategy involves the dynamic selection from all available sources, including the ground truth. However, to ensure a satisfactory quality of the approximation facilitated by the Augmented Gaussian Processes (AGPs), FanG-HPO takes a precautionary step before addressing equation (3.19). Specifically, the algorithm assesses whether the count of additional observations from less expensive sources surpasses those obtained from the ground truth. In such instances, $s' = 1$ is chosen instead of proceeding with the resolution of equation (3.19).

Chapter 4

AutoML Experimental Cases

In the realm of AutoML, theoretical results are challenging to obtain as the problem is complex and not easily amenable to theoretical analysis. As a result, most research contributions are empirical in nature: the proposal of a new technique or approach is accompanied by extensive experimental research demonstrating the benefits of the proposed techniques. Since the focus is on techniques that perform well in general across a wide range of problems, a large number of datasets are required for evaluation. Furthermore, as there is no single AutoML system that represents the state of the art (SOTA) and dominates others, multiple competitors need evaluation. To achieve this, typically each competitor is re-evaluated on the same datasets and ideally under the same conditions, such as the same search space and hardware constraints. Coupled with the long evaluation times of a single candidate solution, which can extend to several hours or even days, as in the case of Neural Architecture Search (NAS), all this translates into a field that demands significant resources and therefore generates immense carbon emissions, many of which could be mitigated, as discussed in Section 2.

Therefore, the goal of this chapter is to make an empirical comparison between two state-of-the-art Bayesian Optimization tools addressing multi-objective and energy-aware optimization and FanG-HPO, a novel approach introduced by A. Candelieri et al..

Additionally experiments involve one benchmark dataset focusing on fairness and accuracy (the theme of Pareto efficiency is addressed) and utilize two renowned Machine Learning algorithms (XGBoost, MLP). In this way the issue of energy-efficiency of HPO is discussed.

4.1 Experimental Setup and Methodology

4.1.1 Architectures and Datasets

To establish or enhance the comparability and reproducibility of results, as well-discussed in Section 2.3, 39 datasets for AutoML have been identified by OpenML. For each of these datasets, an AutoML system is allocated a total of 4 hours to search for a suitable pipeline.

For this analysis, a typical dataset has been chosen to address the fairML theme. Specifically, a binary classification dataset. The dataset in question is ADULT, taken from the R package fairml [1], which also provides implementations of a set of fairness-aware algorithms. The same dataset is also available in the well-known UCI Repository [2], but the version in the R package might differ slightly due to some basic preprocessing operations. It's important to clarify that these operations are related to common data preprocessing (e.g., identifier removal) and not fairness-oriented preprocessing techniques. Below, relevant details for the experiment are provided. The goal of the associated classification task is to predict whether personal income exceeds $50,000 per year, using U.S. 1994 Census data. The dataset consists of 30,162 instances and 14 features, including two sensitive features: "gender" and "race."

Before performing Hyperparameter Optimization (HPO), the dataset is (further) pre-processed by applying one-hot encoding to all nominal features, increasing the final number of features (including the target feature).

4.1.2 Machine Learning Algorithms

Regarding the Hyperparameter Optimization (HPO) of the machine learning algorithms, two distinct ML algorithms were chosen: Multi-Layer Perceptron (MLP) and eXtreme Gradient Boosting (XGB).

Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a type of artificial neural network that consists of an input layer, one or more hidden layers, and an output layer. Each node in a layer is connected to all nodes in the next layer, forming a densely connected network structure. During training, MLPs learn the weights associated with these connections to model complex relationships in the data. The addition of hidden layers allows MLPs to capture more intricate representations of input data, making

[1] https://www.rdocumentation.org/packages/fairml/versions/0.6.1/topics/fairml-package

[2] https://archive.ics.uci.edu/ml/index.php

them suitable for a variety of machine learning tasks, including classification and regression. The number of hyperparameters to optimize is 10 for MLP. The aim is to find the better configurations of hyperparamters in order to achieve the energy-efficiency considering the Pareto efficiency. The search spaces (Table 4.1) utilized for MLP align with those specified in (Schmucker et al., 2020).

HYERPARAMETER	TYPE	DOMAIN	SCALING
N_LAYERS	INTEGER	$\{1,2,3,4\}$	LINEAR
LAYER_1	INTEGER	$\{2,\ldots,32\}$	LINEAR
LAYER_2	INTEGER	$\{2,\ldots,32\}$	LINEAR
LAYER_3	INTEGER	$\{2,\ldots,32\}$	LINEAR
LAYER_4	INTEGER	$\{2,\ldots,32\}$	LINEAR
ALPHA	REAL	$[10^{-6},10^{-1}]$	LOG_{10}
LEARNING_RATE_INIT	REAL	$[10^{-6},10^{-1}]$	LOG_{10}
BETA_1	REAL	$[0.001,0.99]$	LOG_{10}
BETA_2	REAL	$[0.001,0.99]$	LOG_{10}
TOL	REAL	$[10^{-5},10^{-2}]$	LOG_{10}

Table 4.1: sklearn MLP's search space.

XGBoost

XGBoost, short for eXtreme Gradient Boosting, is a powerful and scalable machine learning algorithm designed for supervised learning tasks like classification and regression. It constructs a sequence of decision trees, each correcting errors made by the previous one, using a gradient boosting approach. XGBoost excels in efficiency, incorporating regularization to prevent overfitting, parallelization for speed, and handling missing values during training. It is flexible, accommodating both sparse and dense data, and allows customization of loss functions for different problem types. With features like tree pruning to control depth, XGBoost is widely used for its performance and versatility in various applications. The number of hyperparameters to optimize is 7 for XGBoost. The aim is to find the better configurations of hyperparamters in order to achieve the energy-efficiency considering the Pareto efficiency. The search spaces (Table 4.2) utilized for MLP align with those specified in (Schmucker et al., 2020).

HYERPARAMETER	TYPE	DOMAIN	SCALING
N_ESTIMATORS	INTEGER	$\{1,\ldots,256\}$	LINEAR
LEARNING_RATE	REAL	$[0.01,1.0]$	LOG_{10}
GAMMA	REAL	$[0.0,0.1]$	LINEAR
REG_ALPHA	REAL	$[10^{-3},10^{3}]$	LOG_{10}
REG_ALPHA	REAL	$[10^{-3},10^{3}]$	LOG_{10}
SUBSAMPLE	REAL	$[0.01,1.0]$	LINEAR
MAX_DEPTH	INTEGER	$\{1,2,\ldots,16\}$	LINEAR

Table 4.2: sklearn XGBoost's search space.

4.1.3 Efficiency Metrics

Objectives

The dual-objective ground truth involves the misclassification error (MCE) and Differential Statistical Parity (DSP), calculated through a stratified 10-fold cross-validation process using the complete datasets. To streamline DSP computation for each sensitive feature, the decision was made for DSP $= \max_{i \in F_S} \{DSP_i\}$, where F_S represents the set of sensitive features, and DSP_i denotes the feature-specific DSP value. The process of querying the cost-effective information sources entails calculating the same metrics but utilizing only half of the original dataset.

Sources

In the context of managing multi-fidelity, based on initial assessments using random hyperparameter configurations, the cost of querying the ground truth is approximately double that of querying the cost-effective information sources across all dataset and ML algorithm pairs. Consequently, it was established $c_1 = 1$ and $c_2 = 0,5$. It is important to remark that nominal query cost is not a direct proxy of energy consumption and CO2 emissions, but it drives energy-efficient choices in the two approaches. This will be done for FanG-HPO and BoTorch-MOMF

Instead built upon Hyperband, autogluon-FairBO utilizes successive halving on the validation folds. In essence, if a hyperparameter configuration proves unpromising based on the iteratively collected results from the folds, it is discarded, and the 10-fold cross-validation procedure is prematurely halted. Consequently, the cost of evaluating a hyperparameter configuration in autogluon-FairBO cannot be predefined. Therefore, information sources and their query costs cannot be determined a priori. If the 10-fold cross-validation procedure concludes successfully, the query is considered to have been performed on the ground truth, utilizing $c_1 = 1$. Al-

ternatively, if the procedure is unsuccessful, $c_2 = nf/10$ is considered, with nf representing the number of folds analyzed before halving.

Concerning energy consumption and CO2 emissions, runtime (i.e., query time) is deemed a suitable proxy, as detailed and justified in Section 2.3.

4.1.4 AutoML Methods

The comparison was conducted between a novel approach, FanG-HPO, and two state-of-the-art BO suites capable of multi-objective and energy-efficient optimization: autogluon-FairBO and BoTorch-MOMF. Bi-objective optimization, involving the simultaneous minimization of 10FCV MCE and 10FCV DSP, is handled through scalarization in autogluon-FairBO and via EHVI maximization in both FanG-HPO and BoTorch-MOMF.

Another notable distinction lies in the strategies employed to address energy efficiency. According to the taxonomy outlined in (Tornede et al., 2023), **autogluon-FairBO** falls under the category of "early discarding of unpromising candidates" methods, utilizing Hyperband as its foundation. **BoTorch-MOMF** and **FanG-HPO**, on the other hand, are categorized under "multi-fidelity performance measurements" methods, albeit with a significant methodological difference.

BoTorch-MOMF utilizes a multi-output Gaussian Process (GP) to model three objectives: 10FCV MCE, 10FCV DSP, and the query costs associated with the sources. Additionally, fidelity is treated as an additional decision variable, akin to a hyperparameter of the ML to be optimized.

In contrast, FanG-HPO employs independent Augmented Gaussian Processes (AGP) for each objective, formed by merging observations from different information sources. This distinction is particularly crucial concerning the acquisition function: while both BoTorch-MOMF and FanG-HPO use EHVI, the former penalizes the associated value based on the source's cost (higher fidelity correlates with higher cost), while the latter adopts the two-step mechanism elucidated in Section 3.4.3.

4.1.5 Trial phases

In order to ensure fairness between the methods, it was decided to proceed in the following manner.

1. Running autogluon-FairBO with a query limit of 200[3];

[3]To ensure fairness it is necessary to start from Autogluon-FairBO due to does not provide the option to specify a maximum cumulative query cost as a termination criterion. The only available option is to set a maximum number of queries, indicating the upper limit on the number of hyperparameter configurations to be evaluated.

2. Calculating the cost of each query as nf/10, where nf is the number of folds considered before halving (if any);

3. Computing the total query cost accumulated by autogluon-FairBO during that run (referred to as the budget);

4. Selecting the initial 2(d + 1) hyperparameter configurations queried by autogluon-FairBO and randomly dividing them into two sets, each of size d+1, where d is the number of hyperparameters to optimize;

5. Executing BoTorch-MOMF by initializing the multi-output Gaussian Processes with the two aforementioned sets (noting that fidelity is treated as an additional hyperparameter to be optimized, and its associated query cost is part of the acquisition function). The budget computed earlier serves as a threshold for the cumulative query cost (i.e., termination criterion);

6. Running FanG-HPO by initializing the two Augmented Gaussian Processes (AGPs) with the two sets of observations mentioned above. The termination criterion for FanG-HPO is the same as that used for BoTorch-MOMF;

To reduce the impact of the random initialization in autogluon-FairBO, Ten independent runs were conducted for each combination of ML algorithm and dataset, and the experimental protocol was consistently applied in each of these separate runs.

4.2 Experiment Results

In this section, experimental results are presented with the methodology defined in the Section 4.1.

4.2.1 Accuracy in terms of Runtime and Nominal Query Cost

When assessing AutoML systems, it's not appropriate to solely focus on their ultimate performances at the conclusion of their execution. Rather, one should examine the performance curves (as illustrated in Chapter 2), typically depicted by the best observed metric value concerning the number of queries conducted or the cumulative query cost.

In Chapter 3, it was demonstrated how the transition from Misclassification error to Accuracy and vice versa can be achieved. This is the way:

$$\text{Accuracy} = 1 - \text{Misclassification Rate} \tag{4.1}$$

In conducting the experiment, the objective is to minimize MCE (namely one of the two objective), corresponding to the maximization of ACC. The graphs presented in this chapter analyze how accuracy evolves with changes in *Nominal Query Time* and *Runtime*, one curve for every BO-based approach and run.

Nominal Query Cost

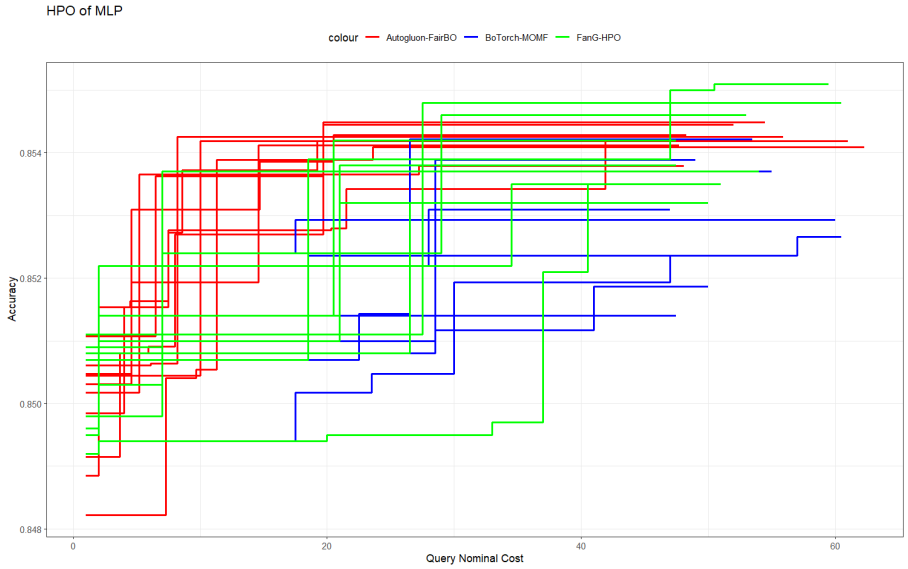

Figure 4.1: Cost-effectiveness of the three BO-based approaches for bi-objective HPO of a MLP classifier over 10 independent runs.

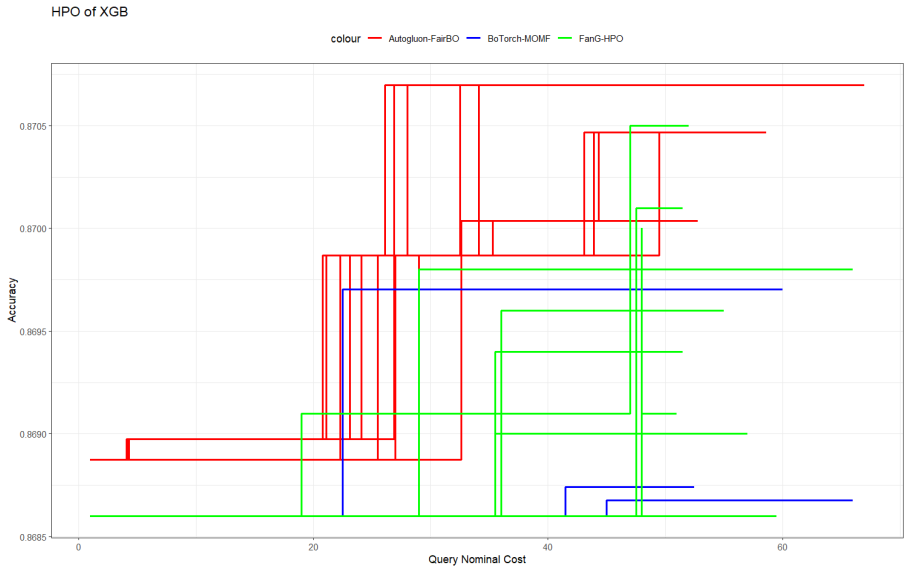

Figure 4.2: Cost-effectiveness of the three BO-based approaches for bi-objective HPO of a XGB classifier over 10 independent runs.

Consideration 1. In terms of accuracy, all three approaches seem to behave similarly, as depicted in the figure 4.1. Autogluon-FairBO, due to its different initialization at the beginning, performs slightly better initially but is later equaled by the others. Depending on the random initialization, FanG-HPO, on the other hand, appears to be slightly better. Another observation can be made about Autogluon-FairBO, which seems to have less dispersion, meaning that the initialization has little influence on the final result, bringing all 10 initializations to roughly the same level of accuracy.

Consideration 2. In terms of accuracy, all three approaches exhibit similar behaviors, as illustrated in Figure 4.2. Autogluon-FairBO, due to its distinct initialization at the beginning, demonstrates slightly better performance initially. FanG-HPO's outcome, however, appears to be more variable based on the initialization. Another observation pertains to BoTorch-MOMF, where the results from the 10 initializations largely coincide, albeit with slight variations. It is worth noting that the initializations of FanG-HPO and BoTorch-MOMF are identical, and they exhibit the same behavior up to the nominal cost of 19.

Runtime

Evaluating cost-effectiveness based on cumulative nominal query cost does not directly quantify the energy consumption or carbon footprint of the three BO-based approaches. As discussed in (Tornede et al., 2023), although runtime is not a precise measure of energy efficiency due to hardware dependency, it is easily measurable on most hardware, unlike other metrics. Additionally, it serves as a practical proxy for environmental impact when detailed information on hardware energy consumption (per time unit) and energy mix composition is unavailable.

Runtime is influenced by factors like parallelism and execution environment heterogeneity, but in scenarios where the same hardware is used for competing approaches, such as in this study, it provides a reasonable approximation of each competitor's CO2 footprint at the specific location and time of execution.

Considering these factors, the decision was made to represent the previous cost-effectiveness curves in terms of cumulative query time (runtime) rather than cumulative nominal query cost. It's essential to note that, to ensure a fair comparison, only the runtime required to evaluate hyperparameter configurations (query time) was considered, excluding the computational time of the approaches themselves, which is negligible compared to query time. When using runtime instead of cumulative nominal query cost, these curves are referred to as ecological performance profiles (Tornede et al., 2023) and are presented in Figures 4.3, 4.4, one for each ML algorithm.

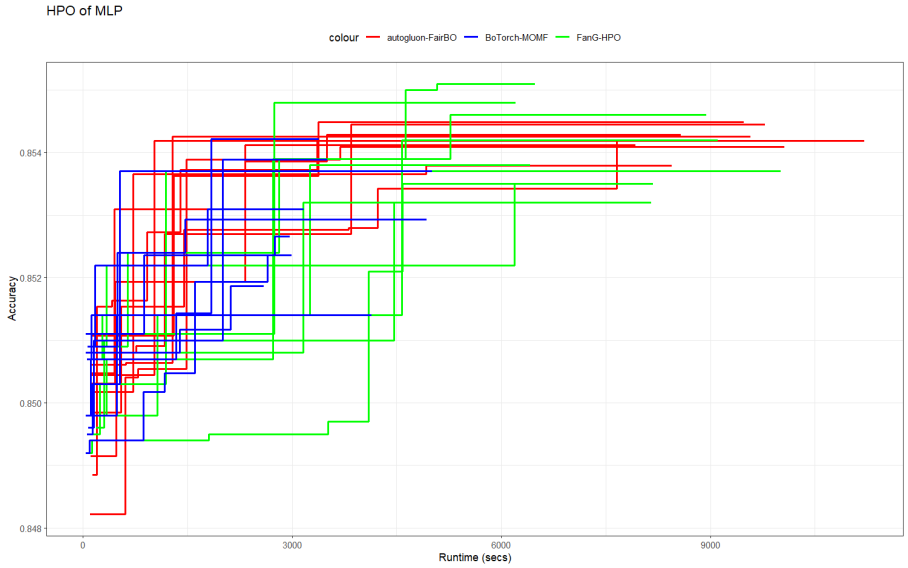

Figure 4.3: Greenness Curve of the three BO-based approaches for bi-objective HPO of a MLP classifier over 10 independent runs.

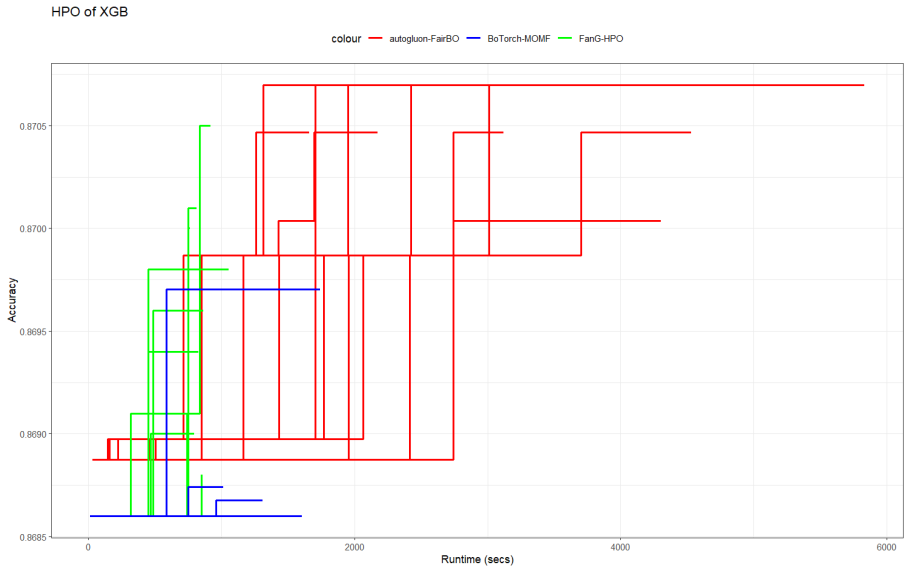

Figure 4.4: Greenness Curve of the three BO-based approaches for bi-objective HPO of a XGB classifier over 10 independent runs.

Consideration 3. By examining the nominal query cost, Autogluon-FairBO appeared as a competitive approach. However, when considering runtime (used as a proxy for CO2 emissions), the efficiency of multi-fidelity and multiple information source BO approaches becomes undeniable. They ensure approximately the same accuracy levels but with a shorter time, indicating lower CO2 consumption. FanG-HPO presents a suitable compromise, delivering better results than Autogluon-FairBO while spending a bit more than BoTorch-MOMF (due to more frequent queries to the true source), but significantly less than Autogluon-FairBO.

Consideration 4. In the case of XGB, Autogluon-FairBO ensures accuracy results that are slightly superior (considering the different initialization, as mentioned). However, a necessary clarification is required regarding FanG-HPO's performance, as it achieves two valuable outcomes. FanG-HPO proves to be efficient both in terms of maximizing the objective function and in terms of energy consumption, making it the optimal choice. It is important to note that in BoTorch-MOMF, the 10 runs appear to overlap.

4.2.2 Fairness in terms of Runtime and Nominal Query Cost

As depicted in Section 3.3.2 it is widely agreed that exclusively prioritizing accuracy when searching for optimal machine learning models can exacerbate biases present in the data, resulting in unfair predictions and decision support. In this section, only the results obtained by minimizing the objective function related to unfairness (aka DSP) are analyzed. However, the presented graphs depict *1-DSP*, so the analysis is framed in a maximization logic.

Consideration 5. As evident from the graph 4.5, Autogluon-FairBO never reaches a DSP level of 0. It consistently maintains the previously observed trend, wherein autogluon-FairBO performs better at lower cost. However, it is quickly surpassed. Both FanG-HPO and BoTorch-MOMF significantly outperform it. The former is slightly superior to the latter at the same cumulative nominal cost.

Consideration 6. The pattern observed for MLP holds true for XGB as well. Although the approach changes, Autogluon-FairBO is still not efficient in terms of maximizing the target function at the same nominal cost. The superiority of FanG-HPO becomes more pronounced in this case as well. Once again, Autogluon-FairBO fails to achieve a DSP level of 0.

Query Nominal Cost

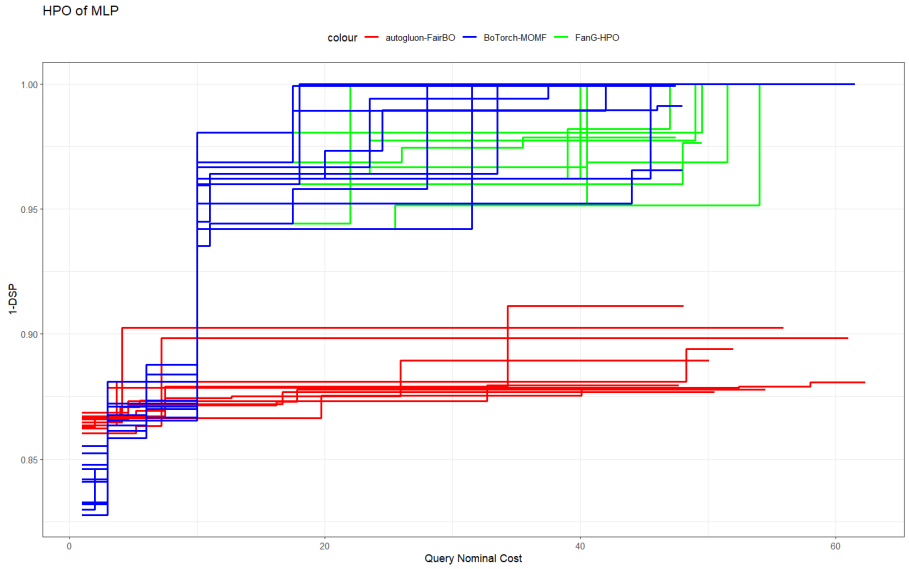

Figure 4.5: Cost-effectiveness of the three BO-based approaches for bi-objective HPO of a MLP classifier over 10 independent runs.

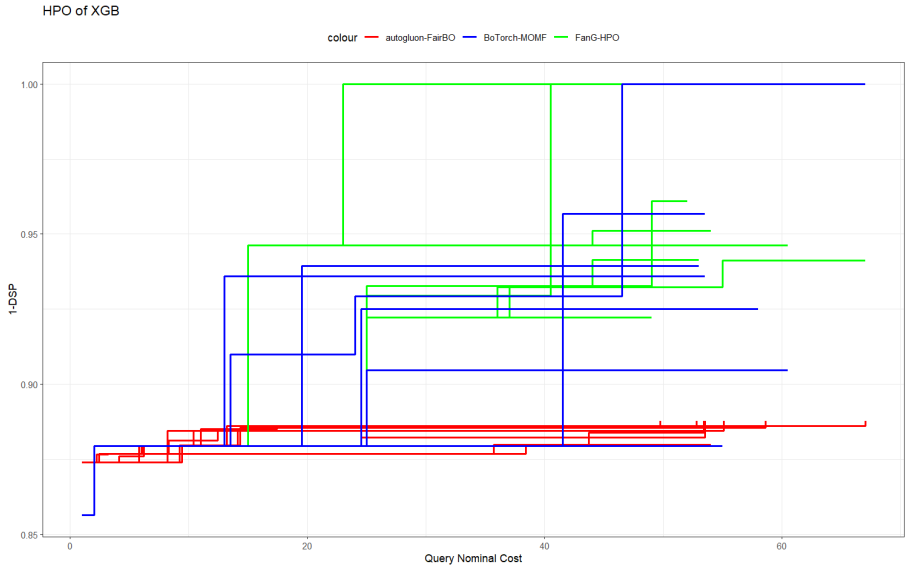

Figure 4.6: Cost-effectiveness of the three BO-based approaches for bi-objective HPO of a XGB classifier over 10 independent runs.

Runtime

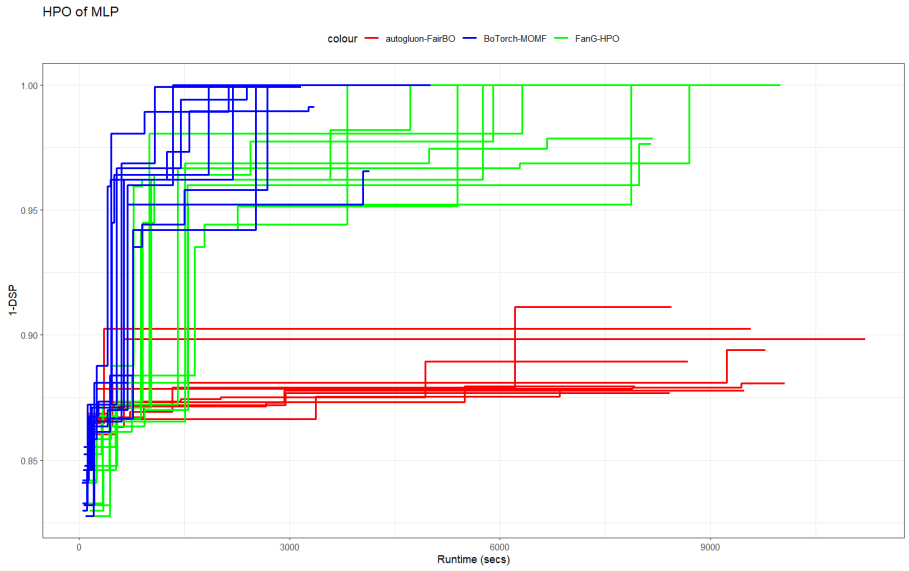

Figure 4.7: Greenness Curve of the three BO-based approaches for bi-objective HPO of a MLP classifier over 10 independent runs.

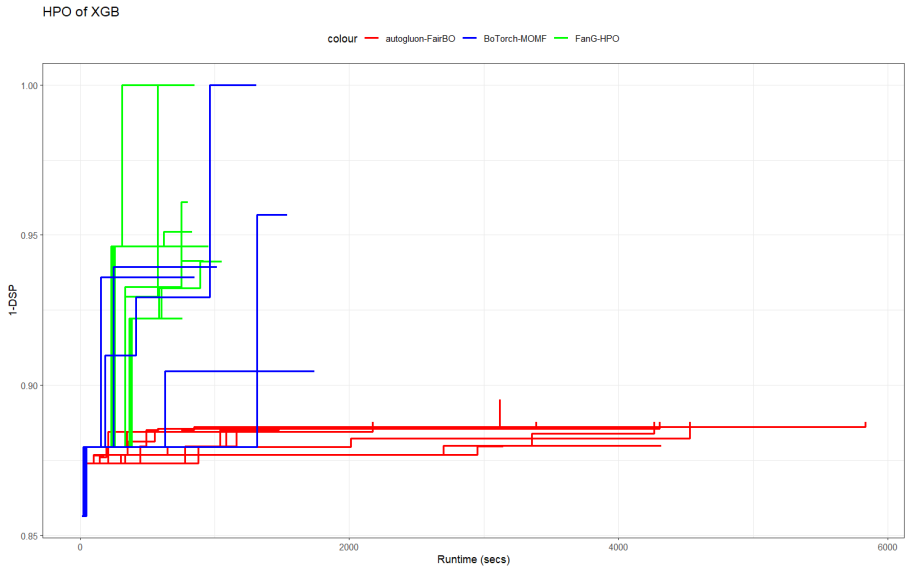

Figure 4.8: Greenness Curve of the three BO-based approaches for bi-objective HPO of a XGB classifier over 10 independent runs.

Consideration 7. Attempting optimization using the MLP algorithm tends to confirm similar results. autogluon-FairBO continues to underperform, while BoTorch-MOMF outperforms FanG-HPO in terms of energy consumption. Notably, out of 10 random initializations, FanG-HPO achieves a DSP level of 0 in 8 instances, whereas BoTorch-MOMF achieves this in 7 cases.

Consideration 8. If the parameters of XGB are optimized, FanG-HPO outperforms both in maximizing the objective function and minimizing energy consumption, significantly surpassing autogluon-FairBO and BoTorch-MOMF. Nonetheless, it can be confirmed that, essentially, these results align with those previously obtained with MLP. The multi-fidelity and multiple information source BO approaches, namely BoTorch-MOMF and FanG-HPO, prove to be more cost-effective compared to the successive halving mechanism of Autogluon-FairBO.

4.2.3 Hyper Volume in terms of Runtime and Nominal Query Cost

As mentioned in Section 3.3, in order to avoid amplifying the bias present in the data, the goal is to minimize a fairness metric as well, and for this reason in recent times, there has been a suggestion to utilize multi-objective hyperparameter optimization for exploring machine learning models that provide balanced Pareto-efficient compromises between accuracy and fairness. While these methods have demonstrated greater flexibility compared to algorithms focused on fairness in machine learning, where accuracy is optimized within certain fairness constraints, they have the potential to significantly elevate energy consumption, particularly in scenarios involving large datasets.

In this section, the curves representing the optimal Hypervolume (of the approximated Pareto front) in relation to the cumulative query cost are presented. Each curve corresponds to a specific BO-based approach and run, and is categorized separately for each ML algorithm-dataset pair. These graphical representations facilitate the assessment and comparison of the cost-effectiveness of the three BO-based approaches. In this analysis, the focus lies on evaluating the Hypervolume (HV) of the approximated Pareto front, which includes dominant hyperparameter configurations. These configurations are exclusively assessed on the entire dataset, involving queries solely on the ground truth. The assessment is conducted as the cumulative query cost increases. It is important to note that, in this subsection, the reference is specifically made to the nominal query costs associated with the two different sources. The aim is to maximize the Hypervolume, where the worst point is located at (0,0) with coordinates corresponding to *accuracy* and *1 - DSP*.

Query Nominal Cost

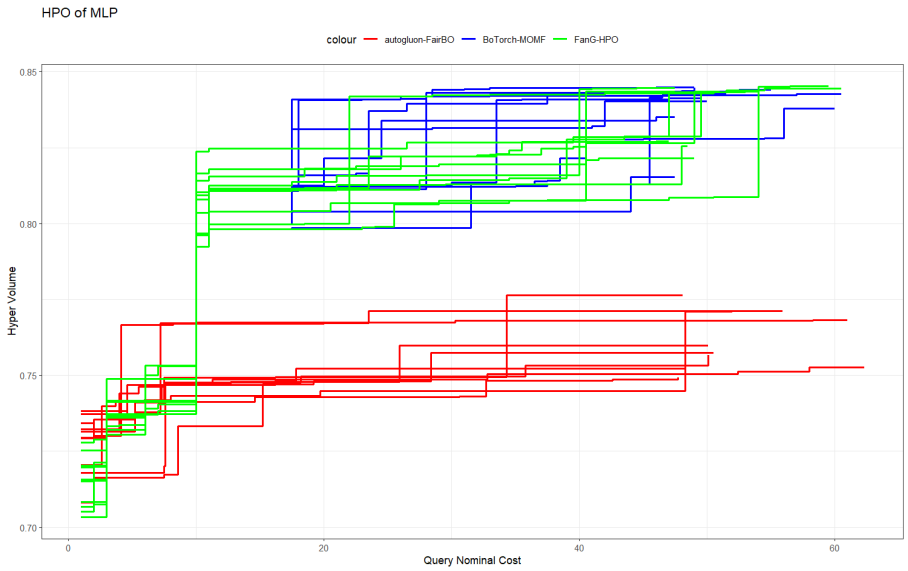

Figure 4.9: Cost-effectiveness of the three BO-based approaches for bi-objective HPO of a MLP classifier over 10 independent runs.

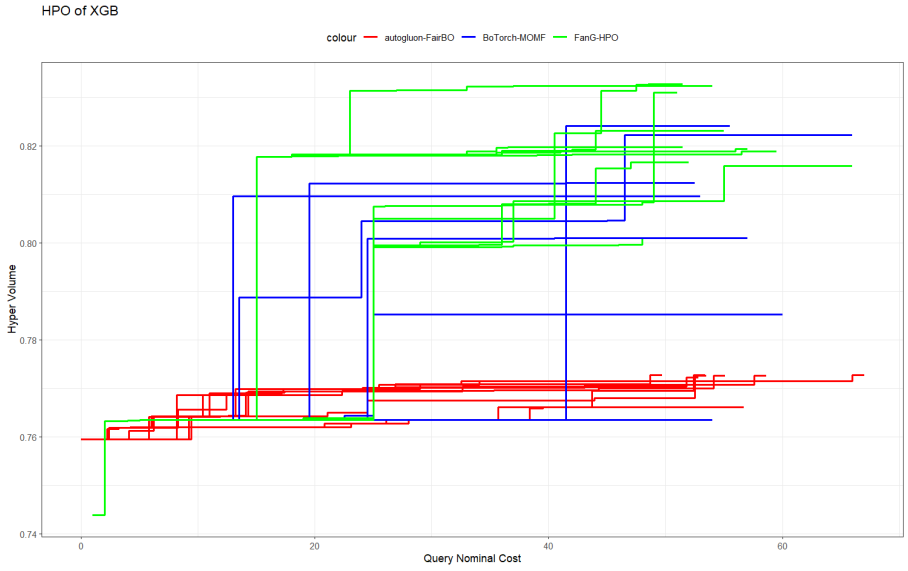

Figure 4.10: Cost-effectiveness of the three BO-based approaches for bi-objective HPO of a XGB classifier over 10 independent runs.

Consideration 9. In general, the successive halving approach (autogluon-FairBO) demonstrated lower "ecological" efficiency compared to multi-fidelity and multiple information source bi-objective hyperparameter optimization (BoTorch-MOMF and FanG-HPO, respectively). As evident from Figures 4.9 and 4.10, the cumulative query time of BoTorch-MOMF and FanG-HPO is notably lower than that of autogluon-FairBO.

Runtime

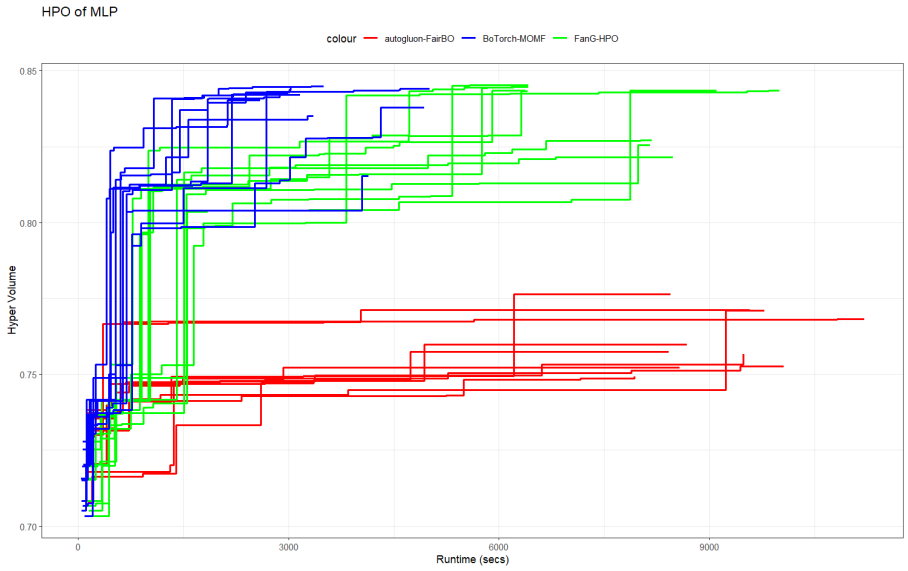

Figure 4.11: Greenness Curve of the three BO-based approaches for bi-objective HPO of a MLP classifier over 10 independent runs.

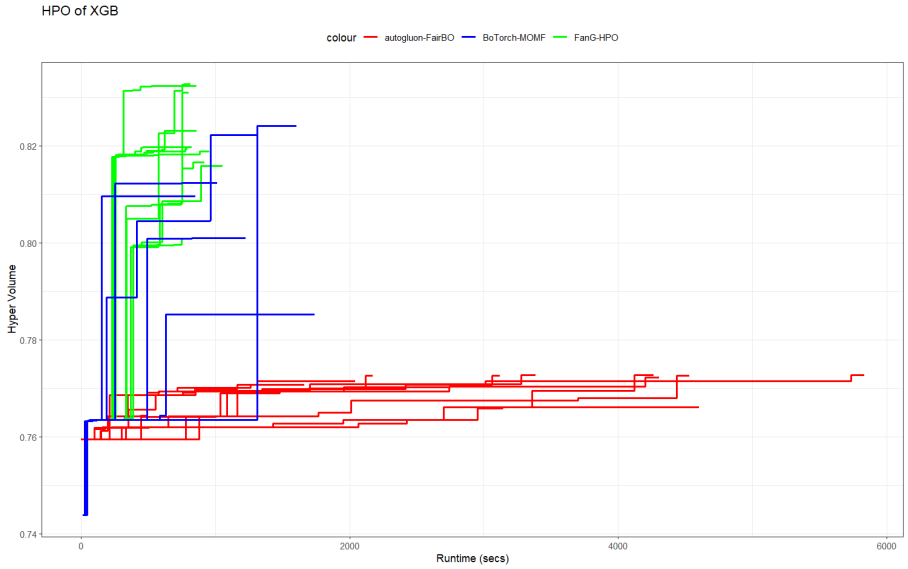

Figure 4.12: Greenness Curve of the three BO-based approaches for bi-objective HPO of a XGB classifier over 10 independent runs.

Consideration 10. XGB is the ML algorithm on which performing bi-objective HPO yields the best results. FanG-HPO is the most effective and green method for HPO.

As commonly demonstrated, XGB typically ranks among the top-performing ML algorithms across various datasets. Furthermore, conducting bi-objective Hyperparameter Optimization (HPO) on it, considering both accuracy and fairness, yields the most extensive collection of Pareto-optimal models. Lastly, when engag-

ing in HPO for XGB using FanG-HPO, it produces the most favorable ecological performance profile.

FanG-HPO, which stands for Fair and Green Hyperparameter Optimization, is designed to search for accurate and fair machine learning models while utilizing smaller portions of large datasets. This approach aims to reduce computational time for training and validation, subsequently lowering energy consumption and, potentially, associated CO2 emissions.

Preliminary results indicate that FanG-HPO does not underperform multi-objective Hyperparameter Optimization (HPO), considering both accuracy and fairness, built on BoTorch-MOMF. Moreover, the ability to handle multiple information sources, such as small portions of the largest dataset, allows FanG-HPO to significantly enhance efficiency in terms of accumulated query costs compared to BoTorch-MOMF. This implies that an approximate Pareto front with a larger hypervolume can be identified with a lower accumulated computational time required for training and validating ML models. This observation was particularly evident in the case of HPO for MLP and XGB on ADULT dataset. Finally, depending on the ML algorithm to be optimized, both FanG-HPO and BoTorch-MOMF that belong to the family of the "multi-fidelity performance measurements" methods HPO can identify more efficient models in Pareto terms than autogluon-FairBO that belongs to the family of "early discarding of unpromising candidates".

4.2.4 Comparison of Average Hypervolume

This section provides a concise summary of the key findings in the study. Figures 4.13, 4.14 present a comparison between FanG-HPO, BoTorch-MOMF and autogluon-FairBO focusing on the evolution of the Hypervolume (HV) associated with the current approximation of the Pareto front concerning the accumulated runtime throughout the HPO process. It is crucial to emphasize that the computation of the approximate Pareto front and associated HV relies solely on observations from the ground truth at each iteration.

Consideration 11. Regarding the HPO of MLP, FanG-HPO surpasses BoTorch-MOMF. Specifically, FanG-HPO identifies MLP models that achieve a superior balance between (MCE) and (DSP) on average. The HV, computed as the mean across 10 runs, is slightly higher for FanG-HPO respect to BoTorch-MOMF, but with a runtime almost double, namely a double Co2 consum. In average, the initial effectiveness of autogluon-FairBO is confirmed, but it is quickly surpassed by other approaches. So, if one aims to consume less energy, even at the cost of sacrificing a small amount of accuracy, BoTorch-MOMF can be chosen.

Consideration 12. Regarding the HPO of XGB, in this case, FanG-HPO not only maximizes the two objective functions better on average compared to the two approaches, but also at significantly lower costs, thus, in terms of CO2 consumption due to runtime is a proxy of Co2 emitted. In average, the initial effectiveness of autogluon-FairBO is confirmed, but it is quickly surpassed by other approaches. FanG-HPO, in this case, results the best choice.

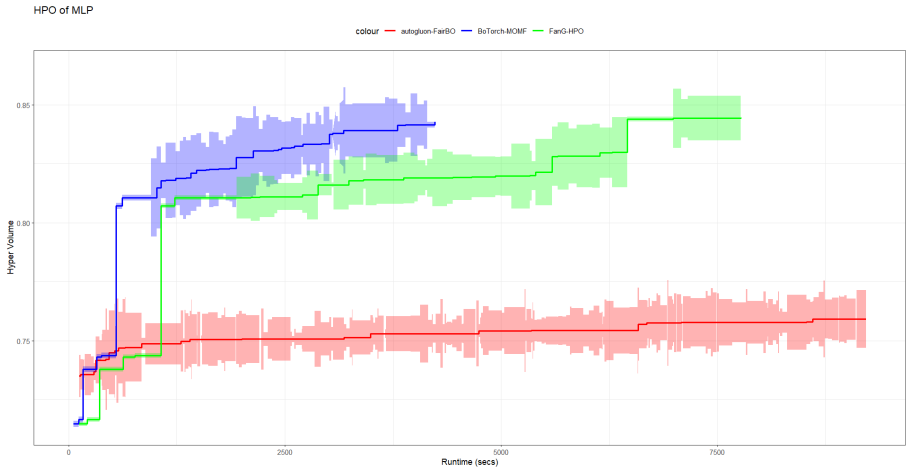

Figure 4.13: Hypervolume (HV) with respect to runtime (secs) over the HPO of MLP process (mean ± standard deviation over the 10 experiments. Comparison between FanG-HPO, BoTorc-MOMF and autogluon-FairBO.

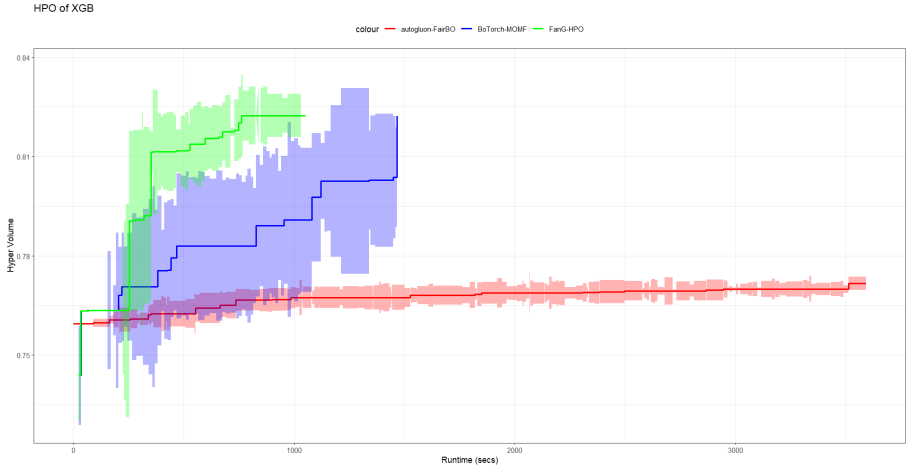

Figure 4.14: Hypervolume (HV) with respect to runtime (secs) over the HPO of XGB process (mean ± standard deviation over the 10 experiments. Comparison between FanG-HPO, BoTorc-MOMF and autogluon-FairBO.

4.3 Interactive visualization on Shiny for R

In this chapter, data and graphs have been presented considering the runtime expressed in seconds. It is worth noting, as highlighted in Section 2.3, that while

runtime does not take into account factors like memory consumption, it exhibits a strong correlation with the energy consumption of the corresponding experiment.

The objective of this section is, therefore, to effectively highlight the consumption of these approaches in terms of CO2 based on:

-Power consumption of the computer: The energy consumption of a computer depends on various factors, including the type of computer, its power, usage, and the energy efficiency of its components. Typically, desktop computers consume between 200 and 800 watts, while laptops consume much less, generally between 20 and 100 watts.

To calculate energy consumption in kilowatt-hours (kWh), you can use the following formula:

$$\text{Power consumption (kWh)} = \left(\frac{\text{Computer's power (Watt)}}{1000}\right) \times \text{Usage time (hours)} \quad (4.2)$$

For example, if a desktop computer has a power of 500 watts and is used for 1 hour per day:

$$\text{Energy consumption (kWh)} = \left(\frac{500}{1000}\right) \times 1 = 0.5\,\text{kWh} \quad (4.3)$$

These are approximate values and may vary based on the specifications of your computer and how you use it. For a more accurate estimate, you can refer to the energy specifications of your computer or use an energy meter.

-CO2 emissions per 1 kWh of production: The CO2 emissions per kilowatt-hour (kWh) vary depending on the source of electricity generation. Different energy sources produce different amounts of carbon emissions during their production. Here are some approximate estimates of CO2 emissions per kWh for common sources:

Coal: Coal-fired power plants can produce an average of about 800-1000 grams of CO2 per kWh.

Natural Gas: Natural gas power plants may have CO2 emissions of around 400-600 grams per kWh.

Oil: Oil-fired power plants can produce an average of about 650-900 grams of CO2 per kWh.

Nuclear: Nuclear power plants generate electricity with low CO2 emissions, generally less than 20 grams per kWh.

Renewable Energies: Sources such as solar, wind, and hydropower can have very low or zero CO2 emissions during electricity production.

It is important to note that these are approximate estimates, and actual emissions can vary based on various factors, including power plant efficiency, technologies used, and the origin of materials. Whenever possible, using low-carbon energy sources can

help reduce the overall environmental impact.

Commonly in a residence, the usage to produce 1 kWh of electricity, is equivalent of approximately 2.56 kWh of fossil fuels is burned, resulting in the emission of approximately 0.53 kg of carbon dioxide into the air. Therefore, it can be said that each kWh produced by the photovoltaic system avoids the emission of 0.53 kg of carbon dioxide.

-Energy mix: The term "energy mix" refers to the composition or distribution of various energy sources used to meet the energy demand in a specific area, region, or country. This composition may include various energy sources such as fossil fuels (coal, oil, natural gas), nuclear, and renewable (solar, wind, hydroelectric, biomass).

Examining the energy mix is important for understanding the environmental impact of an area and for developing strategies for a transition to more sustainable and low-carbon energy sources. Governments and regulatory bodies often monitor the energy mix to assess progress toward environmental goals and to guide policy decisions and investments in the energy sector.

The equation to calculate the amount of emitted CO2 in kilograms is as follows:

$$\text{Kg.Co2} = \text{Kg.Co2/Kwh} \times \text{Kwh} \times \text{h} \times \text{Energy Mix} \tag{4.4}$$

where h is the time taken by each approach to reach a specific hyper volume level.

To do all this, R Shiny [4] . An intuitive and extensible reactive programming model which makes it easy to transform existing R code into a "live app" where outputs automatically react to new user input.

The purpose of this interactive window is to make the quantity of CO2 emissions produced by these optimization algorithms understandable to anyone. To achieve this, it is compared to the distance that would be covered by a gasoline-powered car emitting the same amount of CO2.

Observing Figure 4.15, it is noted that, as revealed in section 4.2.4, for an hyper volume equal to 0.728, autogluon-FairBO consumes less compared to the other two methods. It would emit an amount of CO2 equivalent to what a gasoline car would consume while covering approximately 0.10 km. The calculation of CO2 emissions is done using equation 4.4. The repetition of the same experiment 100 times, or the performance of 100 similar experiments in terms of consumption comparable to our experiment, would consume approximately the same amount of CO2 as a gasoline-powered car produces in about 10 kilometers of travel. A kWh value of 0.5 kWh was used, equivalent to what any laptop would consume in an hour of usage. It is emphasized that this variable depends on the specific computer specifications (aka

[4]Here is the complete repository used to conduct this experimental analysis: `https://github.com/dpiccarreta/energy_efficiency`

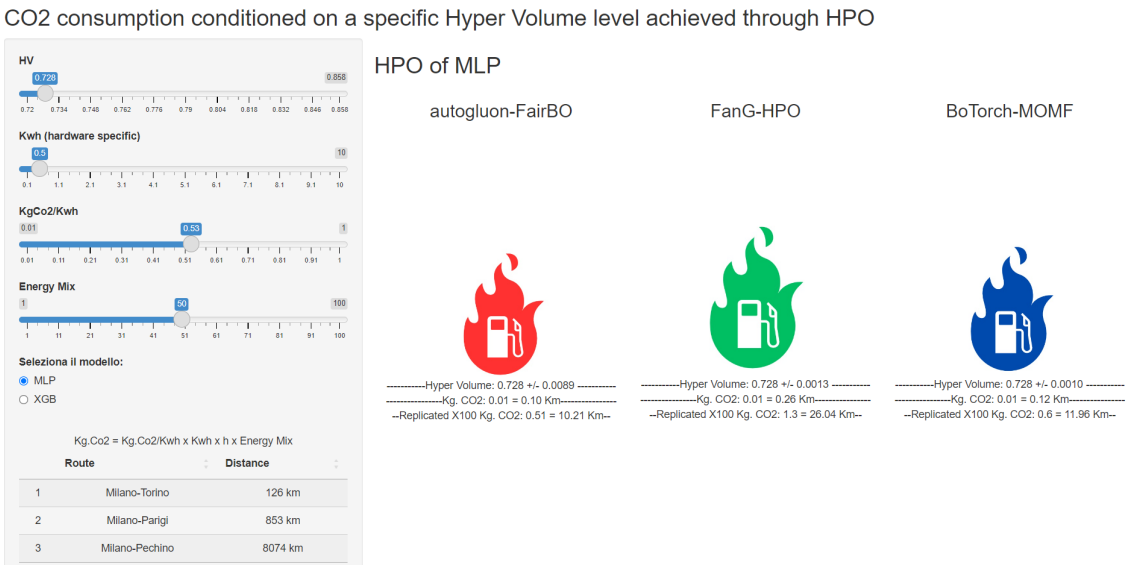

Figure 4.15: Co2 emitted through FanG-HPO, BoTorch-MOMF and autogluon-FairBO varying the level of hyper volume. In this case equal 0.728.

hardware specifics). A ratio of 0.53 kgCO2/kWh is employed, which is the usual ratio for a common household. An energy mix of 50 is used, meaning that 50% of the electricity comes from renewable sources and therefore does not produce CO2 emissions. While for the conversion to kilometers, it is assumed that a car produces 0.05 kg of CO2 per kilometer traveled.

It is highlighted that in figures 4.15 and 4.16, the case of hyperparameter optimization for MLP is displayed. However, the user can choose to visualize the case of XGB.

In Figure 4.16, on the other hand, it is observed that autogluon-FairBO is unable to reach, even after several iterations, a hyper volume level of 0.838, hence the label "not available." Whereas the other two approaches, under the same conditions as before (ceteris paribus), to achieve an HV level of 0.838, would consume the equivalent of CO2 emitted by a gasoline car covering approximately 4.75 km for FanG-HPO and about 2.34 km for BoTorch-MOMF. The repetition of the same experiment 100 times or the performance of 100 experiments similar in terms of consumption to our experiment would consume approximately the same amount of CO2 as a gasoline-powered car produces in about 475 kilometers of travel for FanG-HPO and 234 kilometers for BoTorch-MOMF.

This emphasizes how the theme of transparency in sharing research results among researchers is important to prevent the repetition of the same experiment multiple times.

Therefore, in conducting this hyperparameter optimization, with relatively few hyperparameters, while trying to maximize the two objective functions, energy consumption is not negligible. For about a day of work with these algorithms under

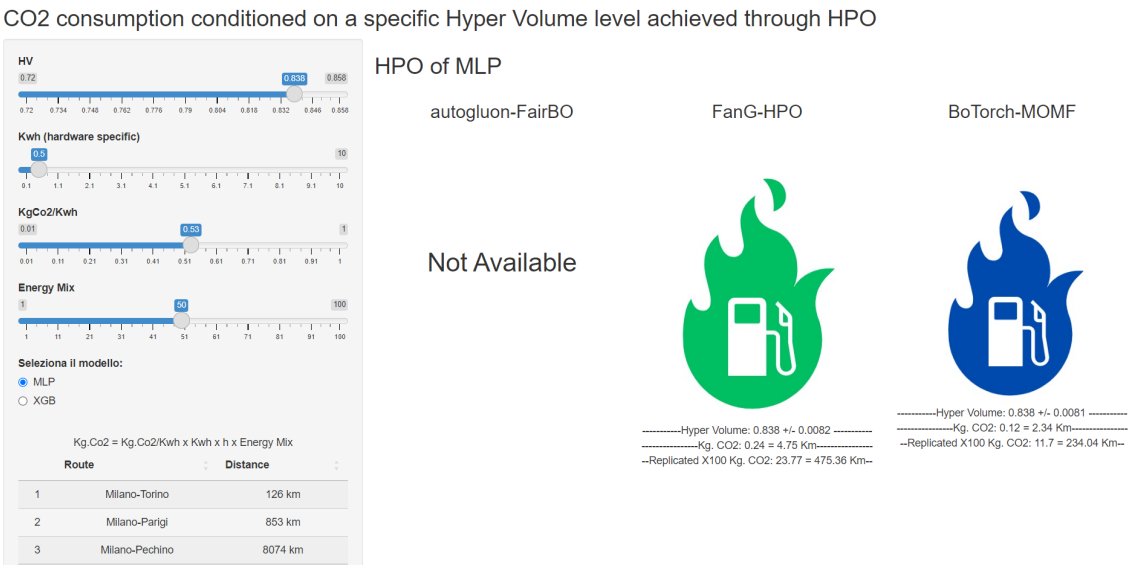

Figure 4.16: Co2 emitted through FanG-HPO, BoTorch-MOMF and autogluon-FairBO varying the level of hyper volume. In this case equal 0.838.

common conditions (using a common laptop connected to a household socket), a quantity of CO2 is emitted equivalent to what a gasoline car would emit to complete the Milan-Turin route, which is quite remarkable.

Chapter 5

Conclusions

5.1 Achievements

This dissertation introduces the concept of Green AutoML, aiming to enhance the efficiency and eco-friendliness of AutoML processes. It explores *(i)* various methods to quantify carbon emissions, suggests *(ii)* different approaches for integration into AutoML practices to improve efficiency, and discusses *(iii)* strategies for reducing benchmark resource requirements.

(iv) A study is conducted to analyze the behavior of diverse AutoML systems under varying CO2e budgets. The empirical findings demonstrate the superior effectiveness of Fairness-aware AutoML approaches, specifically bi-objective Hyperparameter Optimization (HPO). Two methods within the multi-fidelity performance measurement family, BoTorch-MOMF and FanG-HPO, exhibit better ecological performance compared to successive halving (autogluon-FairBO). Specifically, adopting the information source definition in BoTorch-MOMF and FanG-HPO, as opposed to successive halving, enhances the learning ability from different sources, proving advantageous in the sequential optimization process. This aligns with recent results in Candelieri et al., 2021, emphasizing the effectiveness of single-objective HPO based on multiple information sources over FABOLAS (Klein et al., 2017), which incorporates dataset size as an additional hyperparameter for multi-fidelity optimization. Although BoTorch-MOMF and FanG-HPO belong to the same family, their underlying methodological backgrounds differ significantly, as clarified in Section 4.1. This dissimilarity forms the basis for the generally better performance of FanG-HPO. Currently, one of the key practical advantages is that ML developers have the flexibility to choose between BoTorch-MOMF and FanG-HPO based on their preferences and requirements.

5.2 Limitations

Unfortunately, the switch to a greener transition in aspects related to machine learning is fraught with pitfalls. Firstly, the disparity in access and availability of computational resources poses a significant challenge. The conducted experiments revealed that many studied AutoML models require substantial computational resources, creating a gap between academic and industrial researchers. Limiting this research to industrial labs could not only restrict the diversity of ideas but also perpetuate a funding cycle favoring already well-funded groups, potentially leading to a loss of innovation.

Another critical limitation emerged regarding the need for standardization in measuring training time and sensitivity to parameters. The absence of standardized metrics has made it challenging to compare the studied models, complicating the decision-making process for end-users who wish to assess whether the computational resources required are suitable for their specific context.

Lastly, while acknowledging the advantages of cloud computing services in terms of flexibility and, at times, environmental sustainability. The construction of centrally funded computing resources by the government could provide a fairer solution to ensure homogeneous access for all researchers, regardless of their financial resources.

The future of research on Green AutoML is closely tied to the incentives created to drive efforts towards sustainability. Scientific competitions, such as the AutoML competition from 2015-2018, have demonstrated the potential of such incentives in promoting efficiency. The imposition of tight deadlines has forced methodologies to adhere to specific time limits, creating an environment that has pushed for the development of extremely efficient approaches, as highlighted by the PoSH methodology, which later formed the basis for auto-sklearn 2.0. The evolution of these challenges over time, recently focusing on deep learning and few-shot learning, provides valuable foundations to catalyze efficient approaches throughout the research field. For this transition to take place practically, a mental shift is essential, requiring commitment and dedication from every citizen. The observation made by Miguel Luengo-Oroz during his participation in both events in December 2019 highlighted a lack of direct collaboration between AI experts and climate scientists. This was evident between globally renowned conferences, such as NeurIPS in the field of Artificial Intelligence, and the United Nations Conference on Climate Change (COP25), focused on climate science. This phenomenon indicates a limitation in the current effort to effectively integrate AI into the fight against climate change. Overcoming this limitation will require a concerted effort to break down existing barriers and promote greater synergy between the two disciplines.

5.3 Perspectives

If one aims to act in a greener manner, it is not only necessary to create algorithms and approaches that can optimize hyperparameter configuration more cost-effectively than existing state-of-the-art methods but also imperative to initiate a series of actions such as funding, to be transparency and to insert AutoML as much as possible. This section outlines everything that can be done in the immediate future to reduce environmental impact.

5.3.1 Extension of AutoML's Influence

Future research should delve into the automation of manual configuration of machine learning algorithms. The automation of manual configuration of machine learning algorithms can address the issue of slowness and inefficiency associated with human intuition. Human capacity to handle manageable problems is often a valid heuristic but proves inadequate when navigating a high-dimensional space of potential solutions. AutoML, through the efficient traversal of this search space, reduces the energy consumption required to evaluate non-informative candidates, providing more effective solutions for high-dimensional problems (Tornede et al., 2023).

The dual role of AutoML is emphasized. Among its advantages, there is one related to optimization. Broader access to machine learning technology through AutoML can optimize a wide range of processes and systems. This requires a detailed investigation of AutoML applications, such as computational simulation, exploring ways it can reduce energy consumption in specific sectors. Studies by Candelieri et al. (2021) demonstrate that AutoML can enhance predictive maintenance, optimizing cyclical planning of structure maintenance and contributing to ecological balance. Moreover AutoML can be used to develop predictive models on the availability of renewable energy in the energy distribution network. This not only makes AutoML benchmarking more sustainable but also helps balance energy distribution. Works like those by Bergstra et al. (2011) provide insights into predicting energy availability, indicating how AutoML can contribute to environmental efficiency in this context.

5.3.2 Efficient Resource Management

An important research direction is represented by efficiency in model development, which goes beyond the sole consideration of the training data size. Currently, much attention is given to data efficiency, with a particular emphasis on pretraining models on raw data and subsequent fine-tuning on specific tasks. However, efficiency in the pretraining phase is equally crucial. Research by Klein et al. (2017) highlights

the importance of reporting performance with different amounts of training data to optimize model development. To handle better the resources further research should explore how resources are managed throughout the entire AI development cycle. The discussion on the importance of carefully considering costs and benefits associated with each experiment is supported by works like those by Snoek et al. (2012), providing guidelines to promote transparency and optimize the development process. Otherwise implementing intelligent strategies during the development process can improve overall efficiency. Early stopping and other strategies to optimize computational resources deserve further investigation. Research such as that by Bergstra and Bengio (2012) emphasizes the importance of hardware-independent standards for measuring training time and sensitivity to data and hyperparameters.

5.3.3 Incentivizing Research and Sustainability

Promoting sustainability in the field of machine learning should be a moral imperative not only for researchers but also for funding entities such as companies and public institutions. The crucial role of funding agencies is vital in directing research towards sustainability. Initiatives by the German Research Foundation (DFG), as highlighted in works like those by Tornede et al. (2023), indicate how a focus on sustainability can be implemented through special committees and carbon offset mechanisms. Active participation in conferences and journals is crucial to promote sustainability in the scientific community. Organizing special sessions, creating dedicated tracks, and awarding specific prizes for sustainability work highlight the importance of these efforts. Initiatives like the Sustainability AI of 2021 and sustainability checklists (in Appendix .2 there is an example by Tornede et al.) in conferences like AutoML 2022 indicate growing awareness and commitment to sustainability. The growing need to engage funders and research organizations actively in supporting sustainable AutoML research is highlighted in works like those by Bergstra et al. (2011). Establishing specific programs and incentives for green research can catalyze further progress, promoting increased environmental awareness in the scientific community.

5.3.4 Future Research and Commitments

Future research should be based on empirical studies regarding the environmental footprint of AutoML systems. Exploring the behavior of different AutoML systems with varying CO2e budgets, as suggested in works like those by Candelieri et al. (2021), can provide significant insights.

Proposals for creating dedicated sections and tracks in journals and conferences for Green AutoML research. Establishing initiatives similar to those highlighted

in works like those by Bergstra et al. (2011) can further solidify sustainability as an essential component of AutoML research. The idea is to create a unified online space where anyone can share their results in order to highlight different aspects of efficiency and sustainability. Specifically, considering the environmental impact of an article as a criterion in the review process is deemed important. Conversely, arguing for the publication of unsuccessful attempts and negative results is crucial to avoid redundant work on identical ideas. In general, appropriate research incentives can drive the scientific community towards Green AutoML, through the establishment of special sections or dedicated tracks in journals and conferences.

Bibliography

[1] Dhar P., The carbon impact of artificial intelligence, Aug 2020

[2] Schwartz R. et al., Creating efficiency in AI research will decrease its carbon footprint, Dec 2020

[3] Tornede T. et al., Towards Green Automated Machine Learning: Status Quo and Future Directions, Oct 2022

[4] Candelieri A. et al., Fair and Green Hyperparameter Optimization via Multi-objective and Multiple Information Source Bayesian Optimization, May 2022

[5] Strubell E. et al., Energy and Policy Considerations for Deep Learning in NLP, Aug 2019

[6] Candelieri A. et al., Green machine learning via augmented Gaussian processes and multi-information source optimization, 2021

[7] Verdecchia R. et al., A Systematic Review of Green AI, May 2023

[8] Gorodetski M., Hyperparameter Tuning Methods - Grid, Random or Bayesian Search?, Aug 2021

[9] Zach, Misclassification Rate in Machine Learning: Definition & Example, Mar 25 2022

[10] Vinuesa R. et al., The role of artificial intelligence in achieving the sustainable development goals, Nature Communications, 2020

[11] Strubell A. et al., Energy and policy considerations for deep learning in NLP, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019

[12] Schwartz R. et al., Green AI, Association for Computing Machinery, Nov 2020

[13] UNESCO, Preliminary report on the first draft of the Recommendation on the Ethics of Artificial Intelligence. United Nations Educational, Scientific and Cultural Organization, 2021

[14] Thompson, et al., The Computational Limits of Deep Learning, 2020

[15] Martin E. G. et al., Estimation of energy consumption in machine learning, Parallel and Distributed Computing, Elsevier, 2019

[16] Mair J. et al., Quantifying the energy efficiency challenges of achieving exascale computing, 15th IEEE/ACM International Symposium on Cluster Cloud and Grid Computing, 2015

[17] Kloh V. et al., Use of Machine Learning for Improvements in Performance and Energy Consumption in HPC Systems, 2020

[18] Feurer M. et al., Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning, 2021

[19] Vanschoren J., Meta-Learning: A Survey, 2018

[20] Xian Y. et al., Zero-Shot Learning - The Good, the Bad and the Ugly, 2017

[21] Nguyen P. et al., A meta-mining infrastructure to support KD workflow optimization, 2011

[22] Drori I. et al., AutoML using Metadata Language Embeddings, 2019

[23] Singh N. et al., Privileged Zero-Shot AutoML, 2021

[24] Mellor J. et al., Neural Architecture Search without Training, Proceedings of the 38th International Conference on Machine Learning, 2021

[25] Lin M. et al., Zen-NAS: A Zero-Shot NAS for High-Performance Deep Image Recognition, 2021

[26] Rivolli A. et al., Characterizing classification datasets: a study of meta-features for meta-learning, 2018

[27] Torrey L., Shavlik J., Transfer learning, Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, 2010

[28] Lake B. M. et al., One shot learning of simple visual concepts, Proceedings of the 33th Annual Meeting of the Cognitive Science Society, 2011

[29] Wang Y. et al., Generalizing from a Few Examples: A Survey on Few-shot Learning, ACM Comput. Surv., 2020

[30] Mohr F. et al., Predicting Machine Learning Pipeline Runtimes in the Context of Automated Machine Learning, IEEE Trans. Pattern Anal. Mach. Intell., 2021

[31] Yang C. et al., OBOE: Collaborative Filtering for AutoML Model Selection, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, 2019

[32] Hutter F. et al., Algorithm runtime prediction: Methods evaluation, Artif. Intell., 2014

[33] Tornede A. et al., Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis, Proceedings of The 12th Asian Conference on Machine Learning, 2020

[34] Huang L. et al., Predicting Execution Time of Computer Programs Using Sparse Polynomial Regression, Advances in Neural Information Processing Systems 23, 2010

[35] Smith-Miles K., van Hemert J. I., Discovering the suitability of optimisation algorithms by learning from evolved instances, Ann. Math. Artif. Intell., 2011

[36] Eggensperger K. et al., Neural Model-based Optimization with Right-Censored Observations, 2020

[37] Hutter F. et al., ParamILS: An Automatic Algorithm Configuration Framework, J. Artif. Intell. Res., 2009

[38] Ansotegui C. et al., A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms, Principles and Practice of Constraint Programming - CP 2009, 2009

[39] Petrak J., Fast subsampling performance estimates for classification algorithm selection, Proceedings of the ECML-00 Workshop on MetaLearning: Building Automatic Advice Strategies for Model Selection and Method Combination, 2000

[40] Kandasamy K. et al., Multi-fidelity Bayesian Optimisation with Continuous Approximations, PMLR, 2017

[41] Klein A. et al., Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017

[42] Wu J. et al., Practical Multi-fidelity Bayesian Optimization for Hyperparameter Tuning, Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, 2019

[43] Awad N. H. et al., DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization, Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, 2021

[44] Jamieson K. G. et al., Non-stochastic Best Arm Identification and Hyperparameter Optimization, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, 2016

[45] Domhan T. et al., Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, 2015

[46] Swersky K. et al., Freeze-Thaw Bayesian Optimization, CoRR abs/1406.3896, 2014

[47] Thornton C. et al., Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms, Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, 2013

[48] Komer B. et al., Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn, ICML workshop on AutoML, 2014

[49] Stamoulis D. et al., HyperPower: Power- and memory-constrained hyperparameter optimization for neural networks, 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, 2018

[50] Wolff L. F. A. et al., Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models, CoRR abs/2007.03051, 2020

[51] Falkner S. et al., BOHB: Robust and Efficient Hyperparameter Optimization at Scale, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, 2018

[52] Real E. et al., Regularized Evolution for Image Classifier Architecture Search, The ThirtyThird AAAI Conference on Artificial Intelligence, AAAI 2019, 2019

[53] Wever M. et al., AutoML for Multi-Label Classification: Overview and Empirical Evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 43.9, 2021

[54] Boyd S., Vandenberghe L., Convex optimization, Cambridge University Press, 2004

[55] Bergstra J. S. et al., Algorithms for hyper-parameter optimization, Advances in Neural Information Processing Systems 24, 2011

[56] Papoulis A., Probability, random variables, and stochastic processes, McGraw-Hill, 1991

[57] Rasmussen C. E., Williams C. K. I., Gaussian processes for machine learning, MIT Press, 2006

[58] Johnson R. A., Wichern D. W., Applied multivariate statistical analysis, Pearson Prentice Hall, 2007

[59] Bishop C. M., Pattern recognition and machine learning, Springer, 2006

[60] Kandasamy K. et al., Multi-fidelity Bayesian Optimisation with Continuous Approximations, PMLR, 2017

[61] Klein A. et al., Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017

[62] Wu J. et al., Practical Multi-fidelity Bayesian Optimization for Hyperparameter Tuning, Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, 2019

Appendix

.1 Preliminaries in probability theory

Definition 2.1 (Gaussian random vector). A random vector $\mathbf{x} = [X_1, \ldots, X_n]^{\mathrm{T}}$ is said to have joint Gaussian distribution if for any $(\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$,

$$\sum_{i=1}^{n} \alpha_i X_i$$

is a one dimensional Gaussian random variable. A joint probability density function of a Gaussian random vector is given by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}|^{1/2}} \exp\left((\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

where

$$\boldsymbol{\mu} = [\mu_1, \ldots, \mu_n]^{\mathrm{T}} = [\mathbb{E}[X_1], \ldots, \mathbb{E}[X_n]]^{\mathrm{T}}$$

is the mean vector and

$$\mathbf{C} = \begin{bmatrix} \operatorname{cov}(X_1, X_1) & \ldots & \operatorname{cov}(X_1, X_n) \\ \vdots & \ddots & \vdots \\ \operatorname{cov}(X_n, X_1) & \ldots & \operatorname{cov}(X_n, X_n) \end{bmatrix},$$

where $\operatorname{cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$, is the covariance matrix, symmetric and positive semideinite. This is denoted as $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$.

Definition 2.2 (Stochastic process). Suppose that $(\Omega, \mathcal{F}, \mathbb{P})$[1] is a probability space and that $\mathcal{X}$ is some index set. Suppose further that for each $\mathbf{x} \in \mathcal{X}$, there is a random variable $f_{\mathbf{x}} : \Omega \to \mathbb{R}$.[2] The function $f : \mathcal{X} \times \Omega \to \mathbb{R}$ defined as $f(\mathbf{x}, \omega) = f_{\mathbf{x}}$ is called a stochastic process, and is denoted $f = (f(\mathbf{x}) : \mathbf{x} \in \mathcal{X})$. Remark: For a given $\mathbf{x} \in \mathcal{X}$, the function $f(\mathbf{x}, \cdot) : \Omega \to \mathbb{R}$ is simply a random variable, while for a given $\omega \in \Omega$, the function $f(\cdot, \omega) : \mathcal{X} \to \mathbb{R}$ is a deterministic function of $\mathbf{x}$, also referred to as a sample path or a realization of the stochastic process.

[1] Where Ω is a sample space, $\mathcal{F}$ a set of all events and $\mathbb{P}$ a probability measure.

[2] For notational convenience, denoting $f(\mathbf{x}) = f_{\mathbf{x}}$.

Definition 2.3 (Finite-dimensional distributions of a stochastic process). Let $(f(\mathbf{x}) : \mathbf{x} \in \mathcal{X})$ be a stochastic process. The distributions of the finite-dimensional random vectors of the form $[f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^{\mathrm{T}}, \mathbf{x}_i \in \mathcal{X}$, for $i = 1, \ldots, n$ and any finite n are called the finite-dimensional distributions of the process.

Definition 2.4 (Gaussian process). A stochastic process $(f(\mathbf{x}) : \mathbf{x} \in \mathcal{X})$ is called a Gaussian process if all of its finite-dimensional distributions are joint Gaussian distributions.

Specifically, it is stated that $(f(\mathbf{x}) : \mathbf{x} \in \mathcal{X})$ is a Gaussian process with the mean function $m(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ defined as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

and the covariance function $k(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined as

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))\left(f(\mathbf{x}') - m(\mathbf{x}')\right)\right],$$

for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, if for any finite collection of elements $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, It is observed that the corresponding random vector $[f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^{\mathrm{T}}$ follows a joint normal distribution, i.e.

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_n) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}\right).$$

Theorem 1 (Bayes' theorem). Given the probability density function $p(\mathbf{x})$ of random vector $\mathbf{x}$ and the probability density function $p(\mathbf{y})$ of a random vector $\mathbf{y}$, the conditional probability density function $p(\mathbf{x} \mid \mathbf{y})$ is defined as

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})},$$

where $p(\mathbf{x}, \mathbf{y})$ is a joint probability density function of $\mathbf{x}$ and $\mathbf{y}$. Remark: Conditional density $p(\mathbf{x} \mid \mathbf{y})$ is often referred to as the posterior probability density. Proof. See (A. Papoulis, 1991).

.2 Checklist

In the commitment to promote sustainable AutoML, Tornede et al. propose the adoption of a sustainability checklist to be appended to each submitted paper. This checklist should provide a transparent overview of the efforts made by the authors to make their work more environmentally responsible. Below are the key points to

include in the checklist:

- **What key aspects does your approach design include to be efficient?** State if and how your approach is built with efficiency in mind. For example: Do you use warmstarting? Do you apply multi-fidelity evaluations? Does it avoid timeouting evaluations? Is your approach aware of its environmental impact, like expected improvement per kWh or per g/CO2e?

- **What steps did you consider during the development to reduce the footprint of the development process?** State whether and how you attempted to avoid wasting resources.

- **Does the evaluation consider a metric related to environmental footprint?** State whether and how the evaluation does not only consider performance, but also other metrics, e.g., related to improvement per invested CO2e tons.

- **Does the work yield an improvement over SOTA in terms of efficiency or environmental impact?** State whether and how the presented method improves compared to the SOTA in terms of efficiency or environmental impact.

- **Did you add your approach to an existing benchmark?** State whether you added your approach to an appropriate existing benchmark.

- **Did you make the generated data publicly available?** State whether you created a publicly available repository of the data generated during the creation of the work such as pipeline evaluations on datasets.

- **What resources have you used for the final evaluation?** State the type of CPU/GPU hours and the kind of parallelization that has been used.

- **How many CPU/GPU hours have been used for the final evaluation?** State the amount of CPU hours that have been used for the evaluation presented in the paper.

- **What is the used energy mix?** State if and to what degree your experiments were run using renewable energy. To this end, reaching out to your compute center provider will most likely be necessary.

- **What is your footprint?** State how many tons of CO2e you produced during the creation of the paper.

- **Did you compensate the carbon emissions?** Especially if nonrenewable energy has been used for the creation of your paper, state to what degree the corresponding amount of carbon emissions have been compensated, e.g., through supporting a carbon offset project.