Analyzing and Rating Greenness of Nature-Inspired Algorithms

Chander Jindal[a], Kanaishk Garg[a],, Shobhit Kumar[a], Shallu Juneja[a]*

{chanderjindal2, kanaishkgarg, ishobhit3, shallujuneja9}@gmail.com
[a] Department of Computer Science and Engineering, Maharaja Agrasen Institute of Technology, Delhi, India

Abstract: Nature-inspired optimization algorithms (NIOAs) are modelled after natural phenomena such as the behaviour of birds, bees, and other animals. These algorithms are designed to mimic the way that nature optimizes solutions to problems, and they can be used to find the optimal solution to a wide range of problems in fields such as machine learning, computer science, and engineering. This study aims to objectively evaluate energy use and the associated carbon footprint for a few well-known NIOAs. To measure the energy utilized by each algorithm, we used Microsoft Joulemeter. The associated carbon footprint is determined using the recommendations of the Central Electricity Authority of India. According to the study's findings, each algorithm uses varying amounts of energy to accomplish the same task. This study aims to enhance our knowledge regarding the energy consumption behaviour of different NIOAs and aid software designers in selecting greener NIOAs for their task implementation from the available options. The aim is to bring more focus toward greener software solutions to combat growing energy demands and climate change. Future studies might consider more NIOAs and their modifications for energy consumption analysis to identify the most environmentally friendly NIOAs. Additionally, further investigation into the potential effects of different CPU architectures on the efficiency and power usage of the NIOAs may be taken into account.

Keywords: nature-inspired optimization algorithms; energy consumption; carbon footprint; green software; environmental effect; microsoft joulemeter; central electricity authority of india;

1. Introduction

Traditionally, algorithm engineering has focused on the runtime of algorithms, and performance has been the primary metric for evaluating and optimizing algorithms. Today, analysing and minimizing the power consumption of algorithms is just as crucial as optimizing their runtime (Bayer & Nebel, 2009). Recent years have seen a significant increase in the use of the Internet of Things (IoT). Alongside this, the proliferation of smartphones, tablets, and other mobile devices through mobile computing has added to the strain on network and energy resources. Green Computing aims to address this by promoting energy-efficient and environmentally friendly practices. This is achieved through the implementation of energy-efficient techniques and procedures at both the hardware and software levels, aimed at reducing energy consumption, carbon dioxide emissions, and the overall impact on the environment. By doing so, existing devices, applications, and services can become more sustainable (Ahmad et al., 2019). An algorithm's effectiveness and efficiency must be assessed in the context of a particular application since how it is implemented will influence energy consumption and the environment. Software programs may become more eco- and energy-friendly as a result of enhanced algorithms and data structures.

One approach for assessing the ecological consequences of computers and other computing devices is through carbon footprint (Podder et al., 2020). By evaluating a program's degree of power efficiency related to carbon footprint and implementing it into ecologically friendly company operations or procedures, organizations may make the application a crucial component of their corporate social responsibility activities.

Machine learning is a powerful tool that enables the use of data to gain insights, make predictions, and automate decision-making. Machine learning model deployment has grown massively in recent years (Goodfellow et al., 2017). With this much utilization, it becomes crucial that algorithms used to solve these problems use as minimal resources as possible be it in compute time, power, or energy consumption to save ongoing expenses and deliver effective results faster. Thus, considerable issues have emerged regarding energy usage and expense related to developing ML models and training them (Schwartz et al., 2020). Therefore, it is crucial to consider an application's carbon footprint while planning, constructing, and deploying it.

Optimization is the process of modifying an existing system to improve the chances of achieving desired outcomes and reducing the risk of undesirable ones. This concept is widely applied in various fields, including engineering, business operations, industrial design, and many more. Optimization problems may have different objectives such as reducing energy consumption and increasing performance or efficiency. In machine learning, an optimizer is an algorithm or technique to adjust the various parameters of a model in order to minimize the loss function (in simpler terms, the error). The Nature Inspired Optimization Algorithms (NIOAs) are one such family of optimizers.

The majority of traditional optimization algorithms used to address real-world problems are highly non-linear, have several local optima, and involve complex nonlinear constraints. Nature Inspired Optimization Algorithms (NIOAs) are better at avoiding local optima in comparison to conventional optimization algorithms (H. Li et al., 2020). NIOAs are population-based metaheuristics that replicate a wide range of natural phenomena (X.-S. Yang, 2021). Therefore, they are utilized extensively across a wide range of sectors, including manufacturing, environmental engineering, finance, biology, data mining, and more, to tackle complex and nonlinear optimization problems.

More than a hundred NIO algorithms and their variations are now known and available in the literature (H. Li et al., 2020). The Bat Algorithm (BAT), Camel Algorithm (CAM), Cuckoo Search (CS), Firefly Algorithm (FIR), and Particle Swarm Optimization (PAR) are some of the regularly utilized NIO algorithms that will be the subject of this study's examination of energy consumption and associated carbon footprint. These algorithms were taken into account for this study due to the wide range of applications for them. This work aims to demonstrate how one may experimentally assess the energy consumption of different algorithms. Future research may concentrate on different NIO algorithms or methodologies to eliminate architectural biases.

The objectives of this study are as follows:

- Conducting an analytic review of existing literature on
    - Global Energy usage by Information and Communications Technology (ICT)
    - Effects of Information and Communications Technology (ICT) on the environment,
    - Green or energy-efficient programs,
    - Effects of software's power usage on hardware,
    - Analysis of software's consumption of electricity in algorithm implementations,
    - Current state of Energy-efficient algorithms
    - Nature-inspired optimization algorithms
    - Bayesian Optimization
- Implement the NIO Algorithms in Python programming language using NatureInspiredSearchCV provided by the sklearn_nature_inspired_algorithms library and NiaPy for nature-inspired algorithms.
- Performing Hyper-parameter tuning on NIOAs using Bayesian Optimization.
- Determine the Energy each algorithm consumes using Microsoft Joulemeter.
- Based on how much energy each algorithm consumes, calculate its equivalent carbon footprint.

Previous works have compared the energy usage of various programming languages (Georgiou et al., 2017; Pereira et al., 2017, 2021) and implementations of sorting algorithms (Deepthi T & Birunda, 2018; Rashid et al., 2015; Sade Ayodele & Oluwade, 2019; Verma & Chowdhary, 2018). Only one other team has evaluated the NIO algorithms' energy usage and greenness. They assessed the effectiveness of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), and Artificial Bee Colony (ABC) (Green et al., 2022). According to their results, while Differential Evolution is the most efficient, the result cannot be directly compared to this study due to the use of different methodologies to test the algorithm suite.

The energy consumption behaviour of NIOAs has not been well studied when applied to real-world problems. The novel contribution of this study leans more towards finding out how the NIOAs may compare to each other when applied to real-world optimization problems. Python is a more widely utilized language for machine learning tasks. So, analysing the algorithms' performance, and subsequent consumptions will provide more information regarding how they may be applied in industries for optimization tasks. While this approach has its limitations, it is hoped that the results provide an idea about how the NIOAs perform if they were to be applied today to optimize a model. This study aims to compare the energy consumption of different NIOAs when solving a commonly encountered optimization problem. The issue of conducting hyper-parameter tuning is addressed by using Bayesian optimization to select the parameter values and then applying the resultant model to optimize a classification model to see the effects of NIOAs. By understanding the energy consumption of these algorithms, more informed decisions can be made on which algorithm to use in energy-constrained systems, which can contribute to the development of energy-efficient algorithms for technological devices, appliances, and systems. This may have a significant impact on their design and implementation and help to reduce their environmental impact.

The results of this study (as well as other studies in this field) can help in many ways:

- Choosing Energy efficient options: By understanding the energy consumption of these algorithms, one can make more informed decisions on which algorithm to use in energy-constrained systems, such as mobile and embedded devices, which will lead to the development of more energy-efficient algorithms.
- Cost reduction: More efficient algorithms will consume less energy which will also help in reducing the operating costs of mobile and embedded devices, benefiting both individuals and Organizations that utilize these devices.
- Improved performance: By selecting the most appropriate algorithm for a given task, the performance of mobile and embedded systems can be improved, which will have a positive impact on various fields, such as machine learning, computer vision, and control systems.
- Environmental Impact: By reducing the energy consumption and thus in turn carbon footprint of mobile and embedded devices, this study can help lower environmental impact due to the growth of the given technologies.

The rest of this paper is divided into the following sections:

- Section 2 presents a review of the literature on global energy usage by ICT, the effects of ICT on the environment, green or energy-efficient software, the influence of hardware energy consumption on software, the current state of energy-efficient algorithms, and related works on the analysis of energy consumption in algorithm implementations as well as on nature-inspired optimization algorithms and Bayesian Optimization.
- Section 3 covers both macro and micro methodology as well as the planning and design of experiments.
- Section 4 presents the findings and commentary, which goes through the ethical concerns and difficulties of this study as well as the energy usage and associated carbon footprint of each algorithm.
- Section 5 provides a summary of the topic and suggestions for additional research.

2. Literature Review

2.1. Global Energy Usage by Information and Communication Technology (ICT)

Due to the rapidly increasing demand for Communication Technology devices and the establishment of the Internet of Things (IoT), most devices are interconnected nowadays (Albrecht & Michael, 2013). Be it cloud computing (Berl et al., 2010), thin clients like smartphones (Maga et al., 2013), or highspeed network access (Ajmone Marsan & Meo, 2011), they all have had a disruptive impact on the ICT sector. While efficiency improvements have both been made in computations (Kudtarkar et al., 2010) and mass-storage operations (Baliga et al., 2011) but still the growth of electricity usage has outweighed these improvements (Neves & Krajewski, 2012). As this trend continues (A. S. G. Andrae & Edler, 2015) and more and more applications utilizing computation power develop over time, the necessity for reliable electricity increases. It is predicted that energy usage increase might become completely unsustainable by 2040 if nothing is done to tackle the issue (A. Andrae, 2019; Barlage & Shoute, 2021). The growing electricity usage will increase carbon emissions, may it be in production or expanding the energy delivery network. This is not good for the environment for obvious reasons thereby steps need to be taken to make the process and devices more efficient on every scale and frontier possible.

2.2. Effect of ICT on the Environment

Between 2008 and 2021, the ICT sector has grown responsible for almost 1.9 increased percent of the world's carbon emissions, with the remaining 97.9 to 96.1% coming from other industries including the transportation and agricultural industries (Freitag et al., 2021; Webb, 2008). The environment and the economy suffer as a result of the rise in carbon emissions brought on by Greenhouse Gases and other causes (Murugrsan, 2007). The ICT sector can significantly contribute to lowering global carbon emissions by reducing the carbon footprints of its products and services because there is growing global demand for ICT goods and services. Energy-efficient hardware and other embedded systems have been the subject of extensive study (Hosangadi et al., 2005; Schmitz et al., 2005; Shiri et al., 2020; Simunic et al., 1999), but software and application development should also receive significant attention (Capra et al., 2012; D'Agostino et al., 2021).

2.3. Green Software

Green or energy-efficient software is defined as using less energy for effective computing while causing little environmental harm (Naumann et al., 2011). The energy efficiency of web-based software applications and software features has been the subject of numerous research (Kor et al., 2015; Olaoluwa et al., 2015). The software can easily be estimated to use between 25% and 40% of the total energy used by a device, depending on the laptop or mobile battery (Engel, 2015). However, because it is correlated with the host device's life cycle, the indirect impact of software is more challenging to measure (Dastbaz et al., 2015). Only when both the positive and negative impact is taken into consideration throughout the design and deployment phases can the energy efficiency of software be truly accomplished. In light of this, optimizing ICT application services is essential to lowering harmful environmental effects.

2.4. Software's Impact on Hardware-Related energy consumption

The software's energy usage patterns strongly impact how much energy hardware uses and how long a device's battery lasts (Ardito et al., 2015). A device's energy usage may eventually increase if software or application that is poorly built disables various hardware-based energy-saving capabilities (Murugesan, 2008). For instance, it can prevent hardware from using energy-saving features and impact how the hardware is used, which could ultimately increase indirect energy usage (Ferreira et al., 2013). One of the trickiest tasks during the design stage of an embedded system is the development of energy-efficient software that enhances the energy efficiency of a piece of hardware. Various trade-offs between productivity and sustainability will need to be considered to increase software and application productivity while maintaining energy efficiency (Bener et al., 2014).

2.5. Analysis of Energy Consumption in algorithms implementations

Overall, there has been little focus on algorithm-related energy consumption in the currently existing literature. In 2015, Rashid and the team evaluated the energy efficiency of four sorting algorithms implemented in three different programming languages. They found that

Table 1 – Comparative analysis of energy consumption in algorithmic implementations in existing studies.

Year	Title	Author	Description	Result	Remarks
2015	Energy Consumption Analysis of Algorithms Implementations	Mohammad Rashid, Luca Ardito, Marco Torchiano	Experimented measuring energy consumption of different soring algorithms implementations using various programming languages on an ARM device	ARM Assembly and Counting sort are most efficient respectively	Energy Consumption is determined by the computational complexity
2018	Analysis of Energy Consumption of Sorting Algorithms on Smartphones	Mutlidhar Verma, K.R. Chowdhary	Implemented and tracked energy consumption of different sorting algorithms on smartphones	Quick sort is the most efficient	Concept of energy-complexity to track energy demand of algorithm
	Time and Energy Efficiency: A Comparative Study of Sorting Algorithms Implemented in C	T. Deepthi, Antoinette Mary J Birunda	Tracked time and energy efficiency of sorting algorithms when implemented in C	Quick sort is the most efficient	A lot of similar studies are taking place due to numerous factors that can affect energy consumption like programming language, device, etc.
2019	A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms Using Three Programming Languages II: Energy Consumption Analysis	Oluwakemi Sade Ayodele, Bamidele Oluwade	Measured Energy consumption of sorting algorithms for different implementation styles and languages	Iteratively Merge sort is most efficient and Recursively Quick sort is most efficient. C is most energy efficient language for sorting algorithmic implementations	Energy consumption scales up with increasing data size.
2022	Analyzing energy consumption of nature-inspired optimization algorithms	Kor Green, Mohammad Newaj Jamil, Ah-Lian Kor	Measured Energy consumption of some Nature-inspired optimization algorithms using benchmark functions.	Differential Evolution is the most efficient algorithm among their suite.	Need to test NIOAs when applied to real world problems to extend the picture regarding their energy consumption behaviour.

ARM assembly was the most energy-efficient language and Java was the most power-hungry among the three. Among the algorithms, the one with the best performance was the Counting sort. They also concluded that a large part of energy consumption was determined by the computational complexity (or time performance) (Rashid et al., 2015).

Murlidhar and K.R. Chowdhary conducted a study regarding the analysis of energy consumption by sorting algorithm implementations on smartphones in 2018. They found Quick sort to be the most energy-efficient and Bubble Sort to be the least efficient. They also proposed the concept of energy complexity which could be extended in the future to be a similar parameter to time complexity to keep track of the energy consumption of both algorithms and software systems (Verma & Chowdhary, 2018). Another study in 2018 tracked the time and energy efficiency of sorting algorithms when implemented in C. They similarly found Quicksort to be the most energy efficient one among the tested algorithms (Deepthi T & Birunda, 2018).

A 2019 study by Ayodele and Oluwade found that energy consumption scales up with increasing data size. Among iterative and recursive approaches, the iterative approach is more energy efficient for Merge sort but the Recursive approach is more efficient for Quick sort and C is the most energy efficient among the languages they tested (Sade Ayodele & Oluwade, 2019).

In research conducted by Jamil and Kor, the energy consumption of a few nature-inspired algorithms has been analysed on a dataset (Green et al., 2022). Particle swarm optimization, differential evolution, the artificial bee colony algorithm, and the genetic algorithm were all employed. They found Differential Evolution (DE) to be the most environmentally friendly among other optimization algorithms, using much less energy than each of the other methods (See table 1 for tabular representation).

2.6. Current state of Energy-efficient algorithms:

As seen from the previous subsection, the current state of energy-efficient algorithms mainly focuses on comparing sorting algorithms in different implementations, platforms, and languages. While this type of research provides valuable insights for specific use cases, it does not provide a comprehensive understanding of the general energy efficiency of algorithms across different fields of software development. Additionally, most of the existing literature deals with creating specific energy-efficient algorithmic solutions for use cases such as flow time minimization (Albers & Fujiwara, 2007), RFID estimation problem (T. Li et al., 2010), or cloud computing algorithms (Zhou et al., 2020). However, the field is still in its early stage and more research is needed to better understand the energy efficiency of different types of algorithms in different application fields. This can help to identify the most energy-efficient algorithms for various use cases and to reduce energy consumption in software development.

2.7. Nature-inspired optimization algorithms:

Current research on nature-inspired algorithms primarily focuses on the following areas: optimization (Barontini et al., 2017; H. Li et al., 2020; X. S. Yang, 2020; X.-S. Yang, 2021) utilizing metaheuristics (Abdollahzadeh et al., 2021) or heuristic methods (Mohanty et al., 2022); improving environmental sustainability, such as optimizing supply chain processes (Sadrnia et al., 2014), managing energy resources intelligently (Nguyen et al., 2020), and increasing energy efficiency in data centres (Usman et al., 2019); and energy efficiency (Sharma et al., 2019) and optimization (Agbehadji et al., 2021) in wireless sensor network clustering. A little detail about the algorithms this study will cover is below, to help with a better understanding of the algorithms undergoing examination.

Bat Algorithm (BAT/BA): There are around 1000 species of bats. The Bat Algorithm (BA) is based on the Echolocation behaviour of microbats (X. S. Yang, 2010). Microbats are medium-sized bats that eat insects. They used a SONAR technique called echolocation to detect prey. Artificial bats' that imitate actual bats' natural pulse loudness and emission rate serve as search agents in the search process carried out by the Bat Algorithm. Furthermore, it aids in undertaking global optimization since it uses a meta-heuristic approach (Gandomi, Yang, Alavi, et al., 2013). In a variety of fields, including data mining, big data, and machine learning, BA has been used to address challenging issues.

Camel Algorithm (CAM/CA): It is a cutting-edge optimization method motivated by camel migration patterns in the desert and other challenging situations. A camel will often travel toward an area with food and water. In light of that, several variables and operators are taken into account to outline the CA algorithm procedure, including the temperature effect, the supply (water and food), the camel endurance, the camel visibility (and/or hearing) range, random walk, the group effect (multi-solution), the termination condition (dying or moving back), the land conditions (oasis, quicksand, storms, etc.), and limitations (max speed, age, and carrying weight). The camel algorithm's simple structure, along with its efficient search ability allows it to deal effectively with unimodal and multimodal test functions, to find an optimal solution even in the case of difficult ones (Ibrahim & Ali, 2016).

Cuckoo Search (CUC/CS): Xin-She Yang and Suash Deb created the Cuckoo Search optimization algorithm in 2009 (X. S. Yang & Deb, 2010). The obligate brood parasitism of some cuckoo species, which involves the cuckoos' laying their eggs in the nests of host birds of other species, served as its model. Direct combat between some host birds and the trespassing cuckoos is possible. For instance, if the host bird learns the eggs are not its own, it may either discard the alien eggs or quit the nest and make a new one elsewhere. Some cuckoo species, like the New World brood-parasitic Tapera, have developed in such a way that female parasitic cuckoos are frequently extremely skilled at mimicking the colors and patterns of the eggs of a select few host species. Such breeding behaviour was idealized by cuckoo search, which can be used to solve numerous optimization issues (Gandomi, Yang, & Alavi, 2013).

Firefly Algorithm (FIR/FA): The glowing pattern that firefly swarms exhibit serves as the inspiration for Firefly Algorithm (H. Li et al., 2020). FA is incredibly flexible and easy to use. It is based on the idea that attractiveness and brightness are inversely correlated and that Fireflies are attracted to one another if two fireflies have the same level of brightness. The software creates creative approaches and continues to search the solution space. The Random Walk unpredictability factor refers to this. There are several applications for FA, including image compression, antenna design optimization, classification, feature selection, etc.

Particle Swarm Optimization (PAR/PSO): The concept of Particle Swarm Optimization (PSO) is inspired by the swarm intelligence of fish schooling and bird flocks that are in search of food (Shi & Eberhart, 1998). Each particle in these groups has its velocity and position. It has been applied to solve a variety of issues, including data clustering (Esmin et al., 2015), human motion tracking (Saini et al., 2014), cloud resource allocation (Mohana, 2015), assembly line balance (Delice et al., 2017), and cost prediction for software (Sheta et al., 2010). However, a drawback of PSO is that it has a poor rate of convergence during the iterative process and is prone to falling into local optimum in high-dimensional space (M. Li et al., 2014).

Since there are currently very few studies on energy-efficient, nature-inspired algorithms, as this analytical literature evaluation has indicated, this study intends to address this issue to further encourage research in this field.

2.8. Bayesian Optimization:

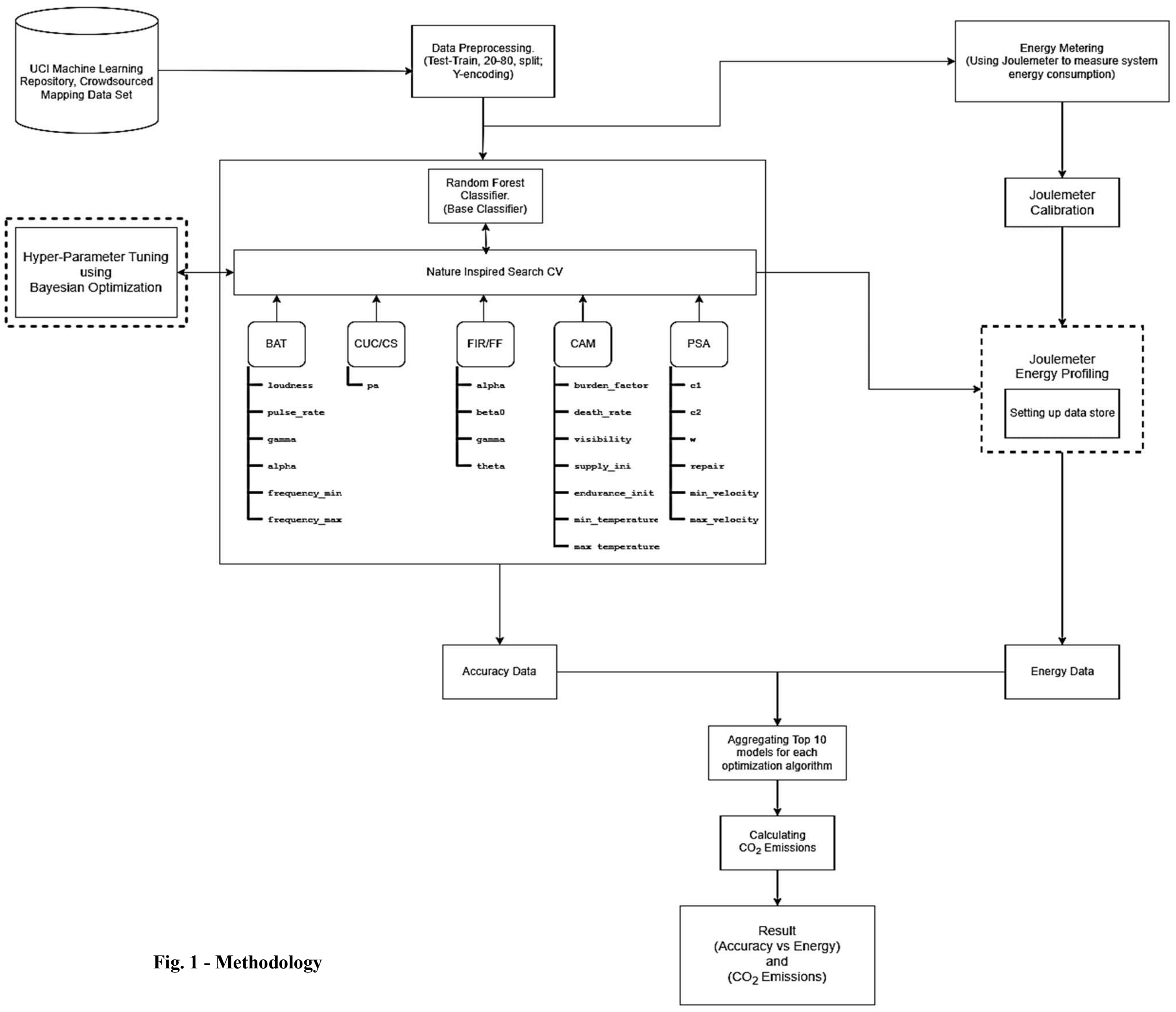
Bayesian optimization (BO) is a powerful technique used to find the optimal set of parameters for a given model. BO is widely adopted in various fields, such as machine learning, computer vision, and control systems, due to its ability to effectively handle high-dimensional parameter spaces. It is useful in a situation where traditional optimization methods may be computationally expensive or impractical (Frazier, 2018). In the context of energy efficiency, Bayesian optimization is used to optimize the energy consumption of different algorithms by searching for the optimal set of parameter values that minimize energy consumption. A study conducted in 2016 demonstrated the suitability of Bayesian optimization as a tool for tuning parameters in Evolutionary Algorithms (Roman et al., 2016). This study will also utilize Bayesian optimization to perform Hyper-Parameter tuning for the algorithm implementations.

3. Methodology

The next section will go through the various tools and software that are used in the study.

3.1. Macro Methodology

The initial step includes data collection and its subsequent pre-processing. Afterward, the accuracy, energy consumption, and carbon footprint of the five algorithms, that are BAT, CUC, FIR, CAM, and PAR are estimated and analysed. The findings are used to draw conclusions that specify the algorithm having the best ratio of Accuracy to Energy Consumption. Lastly, all the results are summarized and discussed (see Fig. 1).



UCI Machine Learning Repository, Crowdsourced Mapping Data Set
Data Preprocessing. (Test-Train, 20-80, split; Y-encoding)
Energy Metering (Using Joulemeter to measure system energy consumption)
Random Forest Classifier. (Base Classifier)
Joulemeter Calibration
Hyper-Parameter Tuning using Bayesian Optimization
Nature Inspired Search CV
BAT
loudness
pulse_rate
gamma
alpha
frequency_min
frequency_max
CUC/CS
pa
FIR/FF
alpha
beta0
gamma
theta
CAM
burden_factor
death_rate
visibility
supply_ini
endurance_init
min_temperature
max temperature
PSA
c1
c2
w
repair
min_velocity
max_velocity
Joulemeter Energy Profiling
Setting up data store
Accuracy Data
Energy Data
Aggregating Top 10 models for each optimization algorithm
Calculating CO_2 Emissions
Result (Accuracy vs Energy) and (CO_2 Emissions)

Fig. 1 - Methodology

3.2. Micro Methodology

3.2.1. Data Set and Programming

The dataset used in this study has been obtained from the UCI machine learning repository. It's called the Crowdsourced Mapping Data Set. Automated classification of satellite pictures into several land cover classifications (impervious, farm, forest, grass, orchard, and water) is done using crowdsourced data from OpenStreetMap (Johnson & Iizuka, 2016).

The NIO methods taken into consideration in this study were implemented accordingly using Python programming language with the sklearn-nature-inspired-algorithms, a machine learning library (Zaťko, 2021), and NiaPy, a library dedicated to Nature Inspired Optimization Algorithms in Python (NiaOrg, 2022).

3.2.2. Energy Profiling

To measure and calculate the expected energy consumption of each NIO technique, we used the Microsoft Joulemeter software (Kansal et al., 2010). This software is designed to monitor the energy usage of running applications or software, as well as specific hardware resources such as CPU, monitor, disk, and idle or base power. It provides a detailed breakdown of energy usage by different components, making it an ideal tool for evaluating the energy efficiency of different NIO techniques.

3.2.3 Carbon Footprint

To calculate carbon emissions, the Central Electricity Authority of India's recommendations have been followed (Singh et al., 2018). After collecting the amount of energy used for an experiment (in kWh), the data is converted to the equal amount of carbon released using the formula below.

- CO2 Emissions = 0.85* E(kW-hr/year) where E is the energy consumed.
- 1kWhr of Energy Consumed = 0.85Kg of CO2 emission
- 72 Joules = 17 mg of CO2 emissions

3.3. Experiment Setup

3.3.1. System Specification

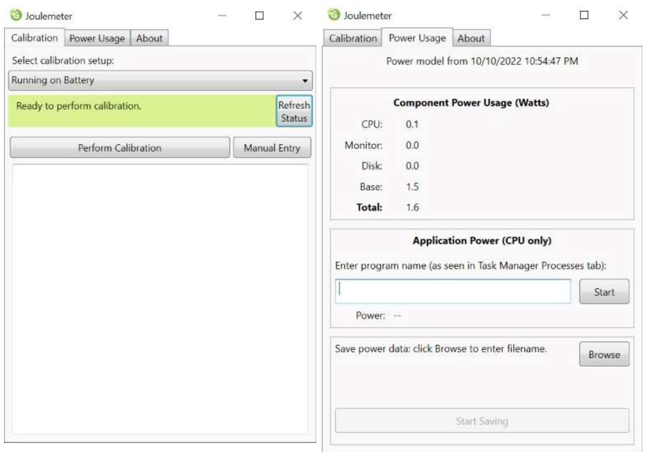
Different hardware requirements would produce various outcomes.
Consequently, a laptop with the following characteristics was used for all experiments (see Table 2):

Table 2 – System Specifications

Specification of Laptop Used	
Model	Lenovo Ideapad 530S
Operating System	Windows 10 (19043.2006)
Processor	Intel® Core™ i5-8250U CPU @ 1.60Hz
RAM	8 GB
Storage	256 GB

3.3.2. Calibrating Joulemeter

In case the auto-calibration (in Joulemeter) does not work, one needs to manually calibrate Joulemeter to get the required power readings. For that define the model using the Manual entry option and add system parameters.



Select calibration setup: Running on Battery

Ready to perform calibration.

Power model from 10/10/2022 10:54:47 PM

Component Power Usage (Watts)

CPU:	0.1
Monitor:	0.0
Disk:	0.0
Base:	1.5
Total:	1.6

Application Power (CPU only)

Enter program name (as seen in Task Manager Processes tab):

Power: --

Save power data: click Browse to enter filename.

Fig. 2 – Microsoft Joulemeter

3.4. Experiment Design

Initially, Microsoft Joulemeter is calibrated on the specified system. Random forest classifier was implemented as the base classifier and the NIO algorithms were utilized to optimize the results. The CPU energy consumption for each instance for all the different algorithms was calculated by hyper-parameter tuning using Bayesian Optimization. Microsoft Joulemeter was used to evaluate power usage with a precision of 0.1 watts. The obtained results were then plotted in form of graphs using the Pandas and Pyplot libraries. The results for each algorithm were recorded in individual CSV files. To compare the energy usage of these five methods, all the results from each algorithm were combined in an Excel file. The experiment design can be summarized as follows

1. Nature Inspired Optimization Algorithm (NIOA): PAR, CUC, CAM, BAT, FIR
2. Programming Language: Python
3. Benchmark Function: Sphere Function
4. Space for Search: [-5.12, 5.12]

4. Findings

Using Bayesian Optimization we have created the 50 parameter sets that will be trained and then ranked according to their performance (accuracy). From them the details of energy consumed by each optimization algorithm for the top 10 parameter sets are shown in Table 3-7 in order:

- Bat Algorithm
- Camel Algorithm
- Cuckoo Search Algorithm
- Firefly Algorithm
- Particle Swarm Optimization Algorithm

Table 3 – Energy Consumed by top 10 parameter sets of Bat Algorithm

Keys	Time Taken (s)	Total Energy (J)	CPU Energy (J)	Disk Energy (J)	Base Energy (J)	Power Consumed (W)
BAT-0042	100	323.20	177.20	0.0	150.0	3.232
BAT-0034	793	2728.30	1561.6	0.1	1189.5	3.440
BAT-0008	76	247.1	136.40	0.0	114.0	3.251
BAT-0029	230	761.90	427.20	0.0	345.0	3.313
BAT-0011	78	251.80	138.5	0.0	117.0	3.228
BAT-0017	533	1741.1	964.30	0.0	799.5	3.267
BAT-0003	306	980.20	535.40	0.0	459.0	3.203
BAT-0040	150	490.00	270.80	0.0	225.0	3.267
BAT-0016	115	371.60	204	0.0	172.5	3.231
BAT-0020	9	29.4	16.4	0.0	13.5	3.267

Table 4 – Energy Consumed by top 10 parameter sets of Camel Algorithm

Keys	Time Taken (s)	Total Energy (J)	CPU Energy (J)	Disk Energy (J)	Base Energy (J)	Power Consumed (W)
CAM -0000	164	527.00	287.30	0.3	264.0	3.213
CAM -0049	704	2264.00	1241.10	0.0	1056.0	3.215
CAM -0024	222	708.40	384.10	0.0	333.0	3.191
CAM -0003	164	529.00	289.40	0.0	246.0	3.226
CAM -0026	97	313.80	172.60	0.0	145.5	3.235
CAM -0008	19	61.3	33.5	0.0	28.5	3.226
CAM -0041	38	123.7	69.10	0.0	57.0	3.255
CAM -0033	840	2695.80	1475	0.6	1260.0	3.209
CAM -0025	124	398.60	218.50	0.0	186.0	3.214
CAM -0006	53	172.80	95.70	0.0	79.5	3.260

Table 5 – Energy Consumed by top 10 parameter sets of Cuckoo Search Algorithm

Keys	Time Taken (s)	Total Energy (J)	CPU Energy (J)	Disk Energy (J)	Base Energy (J)	Power Consumed (W)
CUC-0009	16	51.10	27.70	0.0	24.0	3.194
CUC-0031	264	852.80	468.30	0.0	396.0	3.230
CUC-0010	84	269.3	146.50	0.0	126.0	3.206
CUC-0042	678	2185.10	1198.30	0.0	1017.0	3.223
CUC-0035	40	130.0	71.90	0.0	60.0	3.25
CUC-0040	612	1978.50	1086.70	0.1	918.0	3.233
CUC-0041	258	832.00	455.3	0.0	387.0	3.225
CUC-0039	63	206.30	114.50	0.0	94.5	3.275
CUC-0046	84	273.6	151.30	0.0	126.0	3.257
CUC-0048	69	222.40	122.40	0.0	103.5	3.223

Table 6 – Energy Consumed by top 10 parameter sets of Firefly Algorithm

Keys	Time Taken (s)	Total Energy (J)	CPU Energy (J)	Disk Energy (J)	Base Energy (J)	Power Consumed (W)
FIR-0042	340	1045.90	551.20	0.0	510.0	3.076
FIR-0037	218	709.10	388.80	0.0	327.0	3.253
FIR-0003	933	3013.40	1654.10	0.0	1399.5	3.230
FIR-0049	93	288.2	152.8	0.0	139.5	3.099

FIR-0043	527	1623.00	855.80	0.0	790.5	3.080
FIR-0041	129	397.70	209.50	0.0	193.5	3.083
FIR-0014	74	238.5	131.50	0.0	111.0	3.223
FIR-0009	119	386.5	212.30	0.0	178.5	3.248
FIR-0012	86	280.20	154.70	0.0	129.0	3.258
FIR-0000	92	295.70	162.2	0.0	138.0	3.214

Table 7 – Energy Consumed by top 10 parameter sets of Particle Swarm Optimization Algorithm

Keys	Time Taken (s)	Total Energy (J)	CPU Energy (J)	Disk Energy (J)	Base Energy (J)	Power Consumed (W)
PAR-0016	186	609.60	338.9	0.0	279.0	3.277
PAR-0043	21	67.90	37.8	0.0	31.5	3.233
PAR-0030	75	269.00	157.5	0.3	112.5	3.587
PAR-0003	168	551.70	307.70	0.0	252.0	3.284
PAR-0022	116	380.20	211.10	0.1	174.0	3.278
PAR-0004	87	285.70	159.30	0.0	130.5	3.284
PAR-0045	64	207.2	114.4	0.0	96.0	3.238
PAR-0044	61	200.80	111.80	0.0	91.5	3.292
PAR-0032	153	493.50	270.90	0.0	229.5	3.224
PAR-0015	33	107.9	59.50	0.0	49.5	3.270

Fig. 3 compares the energy usage of each method for clarity, and Fig. 4 displays the average accuracy of the five algorithms for the top 10 epochs.

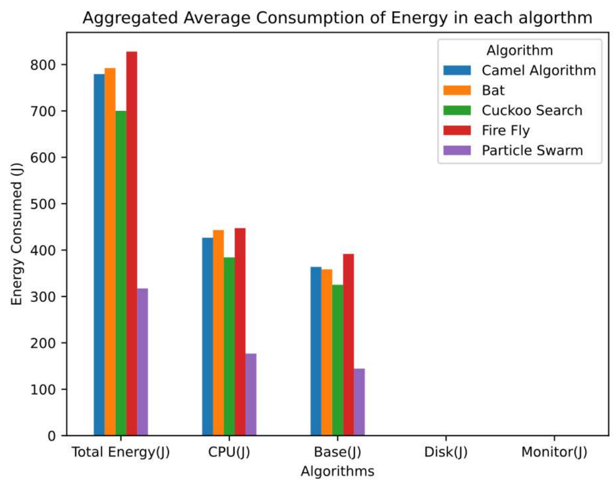


Fig. 3 - Aggregated Average consumption of Energy in each Algorithm

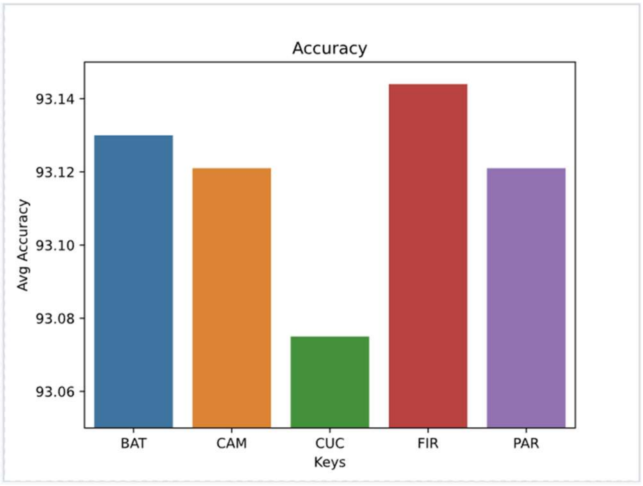


Fig. 4 - Average Accuracy of top 10 epochs achieved for each of the five nature-inspired algorithms

Table 8 depicts the average power, accuracy, and CO2 emission of selected models.

Table 8 – Average Accuracy and Energy Consumption for each algorithm

Name Of Algorithm	Keys	Avg. Accuracy	Avg. Time Taken (s)	Avg. Energy Used (J)	Avg. Equivalent CO_2 Emission (mg)
Bat Algorithm	['BAT-0042', 'BAT-0034', 'BAT-0008', 'BAT-0029', 'BAT-0011', 'BAT-0017', 'BAT-0003', 'BAT-0040', 'BAT-0016', 'BAT-0020']	93.13047	239.0	792.46	187.10861
Camel Algorithm	['CAM-0000', 'CAM-0049', 'CAM-0024', 'CAM-0003', 'CAM-0026', 'CAM-0008', 'CAM-0041', 'CAM-0033', 'CAM-0025', 'CAM-0006']	93.12125	242.5	779.44	184.03444
Cuckoo Search	['CUC-0009', 'CUC-0031', 'CUC-0010', 'CUC-0042', 'CUC-0035', 'CUC-0040', 'CUC-0041', 'CUC-0039', 'CUC-0046', 'CUC-0048']	93.07515	216.8	700.11	165.30375
Firefly Algorithm	['FIR-0042', 'FIR-0037', 'FIR-0003', 'FIR-0049', 'FIR-0043', 'FIR-0041', 'FIR-0014', 'FIR-0009', 'FIR-0012', 'FIR-0000']	93.14431	261.1	827.82	195.45750
Particle Swarm Optimization	['PAR-0016', 'PAR-0043', 'PAR-0030', 'PAR-0003', 'PAR-0022', 'PAR-0004', 'PAR-0045', 'PAR-0044', 'PAR-0032', 'PAR-0015']	93.12125	96.4	317.35	74.92986

Fig. 5 shows the average energy consumed by the five algorithms for the top 10 epochs, while Fig. 6 shows the average CO2 emitted by the five algorithms for the top 10 epochs.

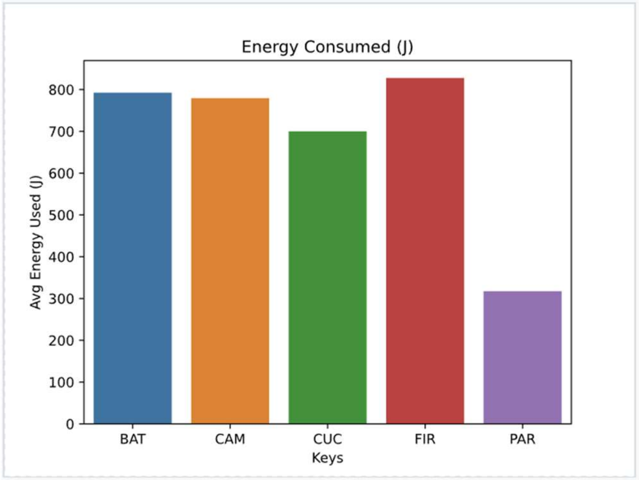


Fig. 5 - Average Energy Consumed of top 10 epochs achieved for each of the five nature-inspired algorithms

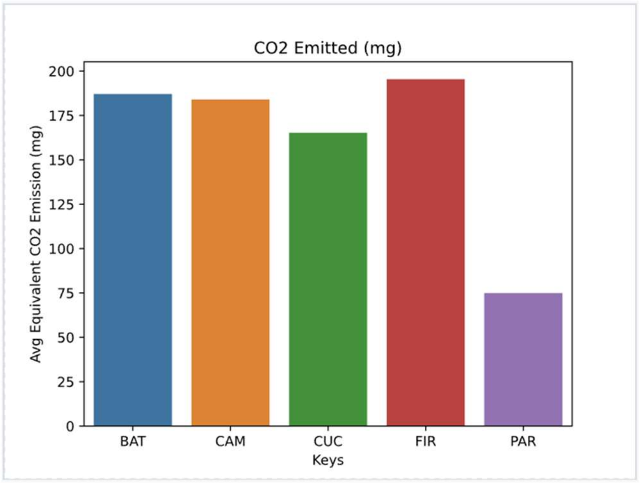


Fig. 6 - Average Equivalent CO2 emissions of top 10 epochs achieved for each of the five nature-inspired algorithms

As was previously said, different hardware specs would produce various outcomes. As a result, the outcomes will alter if the trials are carried out on a laptop with different specifications. PAR is used as the foundation to explore the energy consumption ratio of other algorithms, which is presented in Tables 9 & 10 for CPU energy consumption and Total energy consumption respectively because it is discovered to have the lowest energy consumption.

Table 9 - CPU Energy Consumption Ratio of each algorithm relative to Particle Swarm Optimization

Algorithm	CPU Energy Consumption(J)	Ratio Comparison to PAR
Bat Algorithm	443.18	2.505
Camel Algorithm	426.63	2.412
Cuckoo Search	384.29	2.172
Firefly Algorithm	447.29	2.529
Particle Swarm	176.89	1.0

Table 10 - Total Energy Consumption Ratio of each algorithm relative to Particle Swarm Optimization

Algorithm	Total Energy Consumed(J)	Ratio Comparison to PAR
Bat Algorithm	792.46	2.497
Camel Algorithm	779.44	2.456
Cuckoo Search	700.11	2.206
Firefly Algorithm	827.82	2.609
Particle Swarm	317.35	1.0

Energy usage for each optimization algorithm varies greatly. But as the number of decision trees increases, it is observed that the Particle Swarm Optimization algorithm has the highest accuracy to energy consumption ratio of 0.29343. Firefly Algorithm performs the worst with the accuracy to energy consumption ratio of 0.11252 (Fig 7).

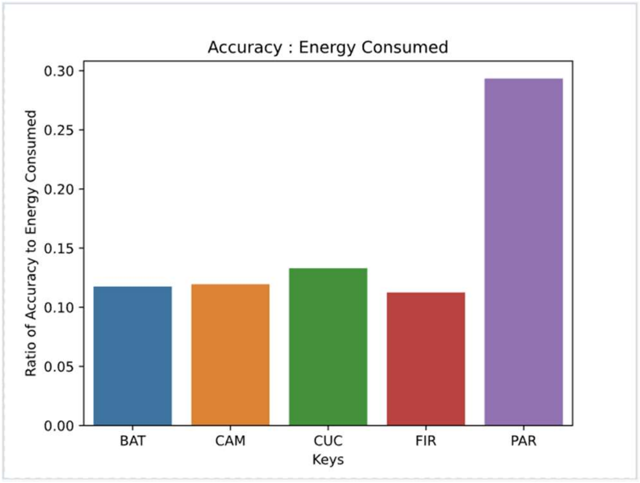


Fig. 7 - Ratio of Accuracy to Energy Consumption of top 10 epochs achieved for each of the five nature-inspired algorithms

5. Conclusion and Future Scope

5.1. Limitations

It is important to note that the present study has certain limitations that should be considered when interpreting the results. Firstly, the results are specific to the problem type and algorithm implementations that were examined and may not be generalizable to other scenarios. Secondly, the study may be influenced by the characteristics of the input data and the hardware utilized. Thirdly, energy consumption was measured using the Microsoft Joulemeter software, which has a resolution of 0.1J per reading. Lastly, it is acknowledged that energy efficiency varies among different hardware, operating systems, and CPUs, thus energy consumption may vary depending on the device and configuration used.

5.2. Conclusion

In the present experiment, it was found that the Particle Swarm Optimization algorithm exhibited the least energy consumption while maintaining relatively high accuracy. As such, it can be concluded that it is the most energy-efficient algorithm among the ones evaluated in this study. The purpose of this study was to emphasize the significance of energy efficiency in technology and the capabilities of nature-inspired optimization algorithms to decrease energy consumption. This will aid in enhancing the energy efficiency of various systems, thereby contributing to a more sustainable future.

5.3. Future Scope

The potential for further research in this field includes the possibility of incorporating other optimization algorithms and applying them to various real-world problems. Additionally, the study can be extended to optimize the algorithms on different hardware platforms to minimize the effects of architectural differences, thus allowing for a comprehensive examination of the energy characteristics of algorithms. Furthermore, there is scope for further research on other methods of hyper-parameter tuning to provide even more optimal and energy-efficient results.

Acknowledgments

The authors and contributors acknowledge active support from their college Maharaja Agrasen Institute of Technology, Rohini, New Delhi.

Appendix A. Additional Information

A.1. Python Libraries

Nature-inspired optimization algorithms implementation provided by the NiaPy machine learning library. To optimize the base classifier using NIOAs, we have utilized NatureInspiredSearchCV from sklearn-nature-inspired-algorithm. Bayesian Optimization is implemented using bayes_opt library. As all of these are open-source libraries, they are subject to change over time.

A.2. Supplementary Materials

All supplementary figures, tables, and code are available at request from the corresponding author

A.3. Data Availability Statement

The dataset used is Crowdsourced data from OpenStreetMap is used to automate the classification of satellite images into different land cover classes (impervious, farm, forest, grass, orchard, water) (Johnson & Iizuka, 2016).

REFERENCES

Abdollahzadeh, B., Soleimanian Gharehchopogh, F., & Mirjalili, S. (2021). Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems*, *36*(10), 5887–5958. https://doi.org/10.1002/INT.22535

Agbehadji, I. E., Millham, R. C., Abayomi, A., Jung, J. J., Fong, S. J., & Frimpong, S. O. (2021). Clustering algorithm based on nature-inspired approach for energy optimization in heterogeneous wireless sensor network. *Applied Soft Computing*, *104*, 107171. https://doi.org/10.1016/J.ASOC.2021.107171

Ahmad, R., Asim, M. A., Khan, S. Z., & Singh, B. (2019). Green IoT — Issues and Challenges. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.3350317

Ajmone Marsan, M., & Meo, M. (2011). Energy efficient wireless Internet access with cooperative cellular networks. *Computer Networks*, *55*(2), 386–398. https://doi.org/10.1016/J.COMNET.2010.10.017

Albers, S., & Fujiwara, H. (2007). Energy-efficient algorithms for flow time minimization. *ACM Transactions on Algorithms (TALG)*, *3*(4). https://doi.org/10.1145/1290672.1290686

Albrecht, K., & Michael, K. (2013). Connected: To Everyone and Everything [Guest Editorial: Special Section on Sensors]. *IEEE Technology and Society Magazine*, *32*(4), 31–34. https://doi.org/10.1109/MTS.2013.2291170

Andrae, A. (2019). Comparison of Several Simplistic High-Level Approaches for Estimating the Global Energy and Electricity Use of ICT Networks and Data Centers. *International Journal of Green Technology*, *5*(1), 50–63. https://doi.org/10.30634/2414-2077.2019.05.06

Andrae, A. S. G., & Edler, T. (2015). On Global Electricity Usage of Communication Technology: Trends to 2030. *Challenges 2015, Vol. 6, Pages 117-157*, *6*(1), 117–157. https://doi.org/10.3390/CHALLE6010117

Ardito, L., Procaccianti, G., Torchiano, M., & Vetrò, A. (2015). Understanding green software development: A conceptual framework. *IT Professional*, *17*(1), 44–50. https://doi.org/10.1109/MITP.2015.16

Baliga, J., Ayre, R. W. A., Hinton, K., & Tucker, R. S. (2011). Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE*, *99*(1), 149–167. https://doi.org/10.1109/JPROC.2010.2060451

Barlage, D., & Shoute, G. (2021). 5. Impact of the Digital Revolution on Worldwide Energy Consumption. *Right Research: Modelling Sustainable Research Practices in the Anthropocene*, 111. https://doi.org/10.11647/OBP.0213.05

Barontini, A., Masciotta, M. G., Ramos, L. F., Amado-Mendes, P., & Lourenço, P. B. (2017). An overview on nature-inspired optimization algorithms for Structural Health Monitoring of historical buildings. *Procedia Engineering*, *199*, 3320–3325. https://doi.org/10.1016/J.PROENG.2017.09.439

Bayer, H., & Nebel, M. (2009). Evaluating Algorithms according to their Energy Consumption. *Mathematical Theory and Computational Practice*.

Bener, A. B., Morisio, M., & Miranskyy, A. (2014). Green software. *IEEE Software*, *31*(3), 36–39. https://doi.org/10.1109/MS.2014.62

Berl, A., Gelenbe, E., di Girolamo, M., Giuliani, G., de Meer, H., Dang, M. Q., & Pentikousis, K. (2010). Energy-efficient cloud computing. *Computer Journal*, *53*(7), 1045–1051. https://doi.org/10.1093/COMJNL/BXP080

Capra, E., Francalanci, C., & Slaughter, S. A. (2012). Is software "green"? Application development environments and energy efficiency in open source applications. *Information and Software Technology*, *54*(1), 60–71. https://doi.org/10.1016/J.INFSOF.2011.07.005

D'Agostino, D., Merelli, I., Aldinucci, M., & Cesini, D. (2021). Hardware and Software Solutions for Energy-Efficient Computing in Scientific Programming. *Scientific Programming*, *2021*. https://doi.org/10.1155/2021/5514284

Dastbaz, M., Pattinson, C., & Akhgar, B. (2015). Green Information Technology: A Sustainable Approach. In *Green Information Technology: A Sustainable Approach*. Elsevier Inc. https://doi.org/10.1016/C2014-0-00029-9

Deepthi T, & Birunda, A. M. J. (2018). Time and Energy Efficiency: A Comparative Study of Sorting Algorithms Implemented in C. *International Conference on Advancements in Computing Technologies*, *4*(2), 25–27. https://www.ijfrcsce.org/download/conferences/ICACT_2018/ICACT_2018_Track/1519365220_22-02-2018.pdf

Delice, Y., Kızılkaya Aydoğan, E., Özcan, U., & İlkay, M. S. (2017). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, *28*(1), 23–36. https://doi.org/10.1007/S10845-014-0959-7/TABLES/7

Engel, M. (2015). Sustainable Software Design. *Green Information Technology: A Sustainable Approach*, 111–127. https://doi.org/10.1016/B978-0-12-801379-3.00007-3

Esmin, A. A. A., Coelho, R. A., & Matwin, S. (2015). A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artificial Intelligence Review*, *44*(1), 23–45. https://doi.org/10.1007/S10462-013-9400-4/METRICS

Ferreira, M. A., Hoekstra, E., Merkus, B., Visser, B., & Visser, J. (2013). Seflab: A lab for measuring software energy footprints. *2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013 - Proceedings*, 30–37. https://doi.org/10.1109/GREENS.2013.6606419

Frazier, P. I. (2018). Bayesian Optimization. *INFORMS Tutorials in Operations Research*, 255–278. https://doi.org/10.1287/EDUC.2018.0188

Freitag, C., Berners-Lee, M., Widdicks, K., Knowles, B., Blair, G. S., & Friday, A. (2021). The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns*, *2*(9), 100340. https://doi.org/10.1016/J.PATTER.2021.100340/ATTACHMENT/F234FB38-A3B0-4C92-99F8-F9960B2E1731/MMC1.PDF

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers*, *29*(1), 17–35. https://doi.org/10.1007/S00366-011-0241-Y/METRICS

Gandomi, A. H., Yang, X. S., Alavi, A. H., & Talatahari, S. (2013). Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, *22*(6), 1239–1255. https://doi.org/10.1007/S00521-012-1028-9/METRICS

Georgiou, S., Kechagia, M., & Spinellis, D. (2017). Analyzing programming languages' energy consumption: An empirical study. *ACM International Conference Proceeding Series*, *Part F132523*. https://doi.org/10.1145/3139367.3139418

Goodfellow, I., Bengio, Y., Courville, A., & Heaton, J. (2017). Deep learning. *Genetic Programming and Evolvable Machines 2017 19:1*, *19*(1), 305–307. https://doi.org/10.1007/S10710-017-9314-Z

Green, K., Newaj Jamil, M., & Kor, A.-L. (2022). Analyzing energy consumption of nature-inspired optimization algorithms. *Green Technology, Resilience, and Sustainability 2022 2:1*, *2*(1), 1–12. https://doi.org/10.1007/S44173-021-00001-9

Hosangadi, A., Fallah, F., & Kastner, R. (2005). Energy efficient hardware synthesis of polynomial expressions. *Proceedings of the IEEE International Conference on VLSI Design*, 653–658. https://doi.org/10.1109/ICVD.2005.90

Ibrahim, M. K., & Ali, R. S. (2016). Novel Optimization Algorithm Inspired by Camel Traveling Behavior. *Iraqi Journal for Electrical & Electronic Engineering*, *12*(2), 167–177. https://www.iasj.net/iasj/download/e32775e0eef3d2eb

Johnson, B. A., & Iizuka, K. (2016). Integrating OpenStreetMap crowdsourced data and Landsat time-series imagery for rapid land use/land cover (LULC) mapping: Case study of the Laguna de Bay area of the Philippines. *Applied Geography*, *67*, 140–149. https://doi.org/10.1016/J.APGEOG.2015.12.006

Kansal, A., Goraczko, M., Liu, J., & Zhao, F. (2010). *Joulemeter: Computational Energy Measurement and Optimization - Microsoft Research*. Microsoft Research. https://www.microsoft.com/en-us/research/project/joulemeter-computational-energy-measurement-and-optimization/

Kor, A. L., Pattinson, C., Imam, I., Alsaleemi, I., & Omotosho, O. (2015). Applications, energy consumption, and measurement. *International Conference on Information and Digital Technologies, IDT 2015*, 161–171. https://doi.org/10.1109/DT.2015.7222967

Kudtarkar, P., DeLuca, T. F., Fusaro, V. A., Tonellato, P. J., & Wall, D. P. (2010). Cost-effective cloud computing: A case study using the comparative genomics tool, roundup. *Evolutionary Bioinformatics*, *2010*(6), 197–203. https://doi.org/10.4137/EBO.S6259/ASSET/IMAGES/LARGE/10.4137_EBO.S6259-FIG2.JPEG

Li, H., Liu, X., Huang, Z., Zeng, C., Zou, P., Chu, Z., & Yi, J. (2020). Newly Emerging Nature-Inspired Optimization-Algorithm Review, Unified Framework, Evaluation, and Behavioural Parameter Optimization. *IEEE Access*, *8*, 72620–72649. https://doi.org/10.1109/ACCESS.2020.2987689

Li, M., Du, W., & Nian, F. (2014). An adaptive particle swarm optimization algorithm based on directed weighted complex network. *Mathematical Problems in Engineering*, *2014*. https://doi.org/10.1155/2014/434972

Li, T., Wu, S., Chen, S., & Yang, M. (2010). Energy efficient algorithms for the RFID estimation problem. *Proceedings - IEEE INFOCOM*. https://doi.org/10.1109/INFCOM.2010.5461947

Maga, D., Hiebel, M., & Knermann, C. (2013). Comparison of two ICT solutions: Desktop PC versus thin client computing. *International Journal of Life Cycle Assessment*, *18*(4), 861–871. https://doi.org/10.1007/S11367-012-0499-3/METRICS

Mohana, R. S. (2015). A Position Balanced Parallel Particle Swarm Optimization Method for Resource Allocation in Cloud. *Indian Journal of Science and Technology*, *8*(Supplementary 3), 1–7. https://doi.org/10.17485/IJST/2015/V8IS3/60501

Mohanty, A., Nag, K. S., Bagal, D. K., Barua, A., Jeet, S., Mahapatra, S. S., & Cherkia, H. (2022). Parametric optimization of parameters affecting dimension precision of FDM printed part using hybrid Taguchi-MARCOS-nature inspired heuristic optimization technique. *Materials Today: Proceedings*, *50*, 893–903. https://doi.org/10.1016/J.MATPR.2021.06.216

Murugesan, S. (2008). Harnessing green IT: Principles and practices. *IT Professional*, *10*(1), 24–33. https://doi.org/10.1109/MITP.2008.10

Murugrsan, S. (2007). Going Green with IT: Your Responsibility Toward Environmental Sustainability — Executive Summary | . In *Cutter Consortium*. https://www.cutter.com/article/going-green-it-your-responsibility-toward-environmental-sustainability-380516

Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, *1*(4), 294–304. https://doi.org/10.1016/J.SUSCOM.2011.06.004

Neves, L., & Krajewski, J. (2012). *GeSI SMARTer 2020: The Role of ICT in Driving a Sustainable Future*. https://www.telenor.com/wp-content/uploads/2014/04/SMARTer-2020-The-Role-of-ICT-in-Driving-a-Sustainable-Future-December-2012._2.pdf

Nguyen, T. H., Nguyen, L. V., Jung, J. J., Agbehadji, I. E., Frimpong, S. O., & Millham, R. C. (2020). Bio-Inspired Approaches for Smart Energy Management: State of the Art and Challenges. *Sustainability 2020, Vol. 12, Page 8495*, *12*(20), 8495. https://doi.org/10.3390/SU12208495

NiaOrg. (2022). *NiaPy Algorithms*. https://niapy.org/en/stable/api/algorithms.html

Olaoluwa, P. O., Kor, A. L., & Pattinson, C. (2015). A Comparative Study on the Energy Consumption of PHP Single and Double Quotes. *Proceedings - 2015 IEEE International Conference on Data Science and Data Intensive Systems*, 232–239. https://doi.org/10.1109/DSDIS.2015.87

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2017). Energy efficiency across programming languages: How do energy, time, and memory relate? *SLE 2017 - Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, Co-Located with SPLASH 2017*, 256–267. https://doi.org/10.1145/3136014.3136031

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2021). Ranking programming languages by energy efficiency. *Science of Computer Programming*, *205*, 102609. https://doi.org/10.1016/J.SCICO.2021.102609

Podder, S., Burden, A., Singh, S. K., & Maruca, R. (2020). *How Green Is Your Software?* Harvard Business Review. https://hbr.org/2020/09/how-green-is-your-software

Rashid, M., Ardito, L., & Torchiano, M. (2015). Energy Consumption Analysis of Algorithms Implementations. *International Symposium on Empirical Software Engineering and Measurement*, *2015-November*, 82–85. https://doi.org/10.1109/ESEM.2015.7321198

Roman, I., Ceberio, J., Mendiburu, A., & Lozano, J. A. (2016). Bayesian optimization for parameter tuning in evolutionary algorithms. *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, 4839–4845. https://doi.org/10.1109/CEC.2016.7744410

Sade Ayodele, O., & Oluwade, B. (2019). A Comparative Analysis of Quick, Merge and Insertion Sort Algorithms Using Three Programming Languages II: Energy Consumption Analysis. *© Afr. J. MIS*, *1*(2), 44–63. https://afrjmis.net

Sadrnia, A., Soltani, H. R., Zulkifli, N., Ismail, N., & Ariffin, M. K. A. (2014). A Review of Nature-Based Algorithms Applications in Green Supply Chain Problems. *International Journal of Engineering and Technology*, *6*(3), 204–211. https://doi.org/10.7763/IJET.2014.V6.697

Saini, S., Rambli, D. R. B. A., Zakaria, M. N. B., & Sulaiman, S. B. (2014). A review on particle swarm optimization algorithm and its variants to human motion tracking. *Mathematical Problems in Engineering*, *2014*. https://doi.org/10.1155/2014/704861

Schmitz, M. T., Al-Hashimi, B. M., & Eles, P. (2005). System-level design techniques for energy-efficient embedded systems. *Springer Science +*

Business Media, 1–194. https://doi.org/10.1007/B106642/COVER

Schwartz, R., Dodge, J., Smith, N. A., & Etzioni, O. (2020). Green AI. *Communications of the ACM*, *63*(12), 54–63. https://doi.org/10.1145/3381831

Sharma, R., Vashisht, V., & Singh, U. (2019). Nature inspired algorithms for energy efficient clustering in wireless sensor networks. *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, 365–370. https://doi.org/10.1109/CONFLUENCE.2019.8776618

Sheta, A. F., Ayesh, A., & Rine, D. (2010). Evaluating software cost estimation models using particle swarm optimisation and fuzzy logic for NASA projects: A comparative study. *International Journal of Bio-Inspired Computation*, *2*(6), 365–373. https://doi.org/10.1504/IJBIC.2010.037016

Shi, Y., & Eberhart, R. (1998). Modified particle swarm optimizer. *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, 69–73. https://doi.org/10.1109/ICEC.1998.699146

Shiri, A., Mazumder, A. N., Prakash, B., Manjunath, N. K., Homayoun, H., Sasan, A., Waytowich, N. R., & Mohsenin, T. (2020). Energy-efficient hardware for language guided reinforcement learning. *Proceedings of the ACM Great Lakes Symposium on VLSI, GLSVLSI*, 131–136. https://doi.org/10.1145/3386263.3407652

Simunic, T., Benini, L., & de Micheli, G. (1999). Energy-efficient design of battery-powered embedded systems. *Proceedings of the International Symposium on Low Power Electronics and Design, Digest of Technical Papers*, 212–217. https://doi.org/10.1145/313817.313928

Singh, N., Jain, C. P., Sharma, K. K., & Jain, P. (2018). *CO2 Baseline Database for the Indian Power Sector*. https://cea.nic.in/wp-content/uploads/baseline/2020/07/user_guide_ver14.pdf

Usman, M. J., Ismail, A. S., Abdul-Salaam, G., Chizari, H., Kaiwartya, O., Gital, A. Y., Abdullahi, M., Aliyu, A., & Dishing, S. I. (2019). Energy-efficient Nature-Inspired techniques in Cloud computing datacenters. *Telecommunication Systems*, *71*(2), 275–302. https://doi.org/10.1007/S11235-019-00549-9/METRICS

Verma, M., & Chowdhary, K. R. (2018). Analysis of Energy Consumption of Sorting Algorithms on Smartphones. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.3168550

Webb, M. (2008). *A report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI)*. https://gesi.org/research/smart-2020-enabling-the-low-carbon-economy-in-the-information-age

Yang, X. S. (2010). A New Metaheuristic Bat-Inspired Algorithm. *Studies in Computational Intelligence*, *284*, 65–74. https://doi.org/10.48550/arxiv.1004.4170

Yang, X. S. (2020). Nature-inspired optimization algorithms: Challenges and open problems. *Journal of Computational Science*, *46*, 101104. https://doi.org/10.1016/J.JOCS.2020.101104

Yang, X. S., & Deb, S. (2010). Cuckoo Search via Levy Flights. *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 210–214. https://doi.org/10.48550/arxiv.1003.1594

Yang, X.-S. (2021). Nature-Inspired Optimization Algorithms. In *Academic Press*. Elsevier. https://doi.org/10.1016/C2019-0-03762-4

Zaťko, T. (2021). *sklearn-nature-inspired-algorithms · PyPI*. https://pypi.org/project/sklearn-nature-inspired-algorithms/

Zhou, Q., Xu, M., Singh Gill, S., Gao, C., Tian, W., Xu, C., & Buyya, R. (2020). Energy Efficient Algorithms based on VM Consolidation for Cloud Computing: Comparisons and Evaluations. *Proceedings - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020*, 489–498. https://doi.org/10.1109/CCGRID49817.2020.00-44