الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة

People's Democratic Republic of Algeria

وزارة التعليـم العالـي والبحـث العلمـي

Ministry of Higher Education and Scientific Research

Nº Ref :……

University center
Abd Elhafid Boussouf

Mathematics & Computer Science Institute **Computer Science Department**

Thesis prepared for the Master's degree

Specialty: Artificial intelligence and its applications (I2A)

Algorithm Selection on Continuous problems using Deep Learning

Prepared by:

- **Harkati Said Dhiya Edinne**
- **Kadri Abderraouf**

Jury Reviewed:

Dr. Boulmerka Aissa……..… Président

Mrs. Meriem Talai ……..…Examiner

Mrs. Afri Faiza……...Superviser

Academic year: 2023/2024

Acknowledgment

First and foremost we want to express with these few lines our thanks ,our gratitude to all those in whom, through their presence, their support, their availability and their advice, we had the courage to accomplish this work. Above all, it seems appropriate to give thanks to Allah almighty for given us the courage, the will, the patience, and the knowledge to accomplish this work and without his aid; we would not have been able to.

We would first like to thank our supervisor Mrs Faiza afri, teacher at the university center of Mila, for all the support, help, orientation, guidance, as well as for her valuable advice and encouragement during the completion of our dissertation. we learned a lot from her and it was an honor and great pleasure to work with her. We would like to express my sincere thanks as well for th jury members the president Dr Boulmerka Aissa and for Mrs Talai Meriem for giving us the privilege of their presence and being our examiners today.

We want to give this special message of love and appreciation to all teachers that guided us throughout our journey and all the university team that allowed for the smooth college experience

We would also like to thank all the people who, directly or indirectly, helped us in the development of this dissertation. Thank you to all friends and family members for their help, support and encouragement during difficult times.

Dedication

With the expression of my utmost gratitude, I dedicate this modest work to those who, whatever the terms encapsulate, I would not be able to never to express my sincere love to them.

My beloved parents whom I could never repay them enough for their love and sacrifices, I present this humble work in your names

To my brothers and sisters all the love and appreciation for you who stood by my side along my journey, thank you for being the pillar that I leaned on through my growth

To all my friends close they were or far my greatest thanks from deepest of my heart

To all those whose names I may not mention but whose contributions have nonetheless left an indelible mark on my journey thank you. Your words of encouragement, your belief in my abilities, and your willingness to lend a listening ear during moments of doubt have not gone unnoticed.

This work stands as an evidence of your presence, shining as my beacon of light throughout my academic journey, thank you for being my guidance toward this endeavor.

Harkati Said Dhiya eddine

Dedication

This thesis is dedicated to those whose unfailing support and belief in my abilities have illuminated my path through the many turns and turns of academia. To my family, the pillars of my journey have been their unending love, support, and understanding. My will to succeed has been strengthened by your unfailing faith in me and your persistent presence.

I offer my success to my parents, the dearest people to my hearts as a symbol of my appreciation and gratitude for always giving without any reservation, I cannot repay your favors.

To my brothers and sisters whom are the closest companions and best of friends that gave me strength and wisdom by supporting without end

Thanks to my friends, specially my comrades along my student years

All my gratitude to all education team, to all the teachers that contributed across the years in my academic journey.

I give this humble work to all those that contributed to my success.

Kadri abderraouf

Contents

Acknowledgment.....II

Dedication III

Contents V

Figures table VIII

List of tables X

List of abbreviations XI

Abstract.....XII

General introduction 1

Chapter 1 3

Introduction:..... 4

1.1 History and development 4

1.1.1 What is Benchmarking?..... 4

1.2 BBOB Framework..... 7

1.2.1 Characteristics of Black-Box continuous Optimization Problems 7

1.3 COCO Framework 8

1.4 BBOB Test Suites 9

1.4.1 Benchmark Functions 9

1.4.2 Evaluation metrics for algorithm performance:..... 11

1.5 Importance of Benchmarking 11

Conclusion..... 12

Chapter 2 13

Contents

Introduction 14

2.1Exploratory Landscape Analysis (ELA) for characterizing problem landscapes 14

2.3 Evaluation metrics for algorithm performance: 16

2.6 Algorithm selection problem: 16

2.4 Algorithm selection framework 17

2.5 Algorithm Selection Approaches 18

2.6 Algorithm Selection Strategies 20

2.7 Adaptive and Online Learning 20

2.8 Related work for algorithm selection in black-box continuous optimization 20

Conclusion 22

Chapter 3 23

Introduction 24

3.1 The Dataset 24

3.2 Preprocessing 36

3.3 Network architecture: 37

Chapter 4 44

Introduction 45

4.1 Model evaluation and comparison 45

4.2 Performance plots 46

4.3 Confusion matrix review 48

4.4 Interpretation of Confusion Matrix 51

4.5 Solvers review 55

Comparing the ASM‘s with the SBS : 56

4.6 Discussion and results 57

4.7 Application's implementation 57

4.8 Used libraries and frameworks 63

4.9 Our machine capabilities 66

General conclusion 68

5 Bibliography 69

5.1 Webographie 75

Figures table

Figure 1.1 : *Benchmarking process plan* *6*

Figure 1.2 *: Schema for black box optimization framework* *7*

Figure 1.3 *: COCO project general structure* *8*

Figure 2.1: *Relationships between high-level features and low-level feature classes* *15*

Figure 2.2 : *Schematic view of how (ELA) can be used for improving the automated algorithm selection process.* *32*

Figure 2.3: *Diagram of the algorithm selection framework proposed by Rice* *17*

Figure 2.4: *Taxonomy of different algorithm selection solutions* *19*

Figure 3.1 : *sample of the raw performance data pre-treatment of dimension 2 data.* *27*

Figure 3.2: *The BBOB datasets by the coco-data archive.* *28*

Figure 3.3: *ERT calculation example* *29*

Figure 3.4: *RELERT calculation example* *30*

Figure 3.5: *the graphical user interface Flacco for BBOB features extraction* *33*

Figure 3.6 : *scheme model development plan* *35*

Figure 3.7: *pseudo code for the preprocessing phase* *37*

Figure 4.1: *Accuracy and loss plot of the ASM 1on the Dataset* *47*

Figure 4.2 : *Accuracy and loss plot of the ASM 2on the Dataset* *47*

Figure 4.3 : *the home interface for the web app* *58*

Figure 4.4 : *the pretrained model interface* *58*

Figure 4.5 : *ASM training plot* *59*

Figure 4.6 : *testing model interface* *59*

Figure 4.7 : *uploading the testing data* *60*

Figure 4.8 : *Testing results* *61*

Figure 4.9 : *train model on new dataset* *61*

Figure 4.10 : *uploading the training dataset* *62*

Figure 4.11 : *results from training* *63*

Figure 4.12 : *TensorFlow Logo* *63*

Figure 4.13 : *Keras Logo* *64*

Figure4.14 : *Pytorch Logo* *64*

Figures table

Figure 4.15 : *Pandas Logo* *65*

Figure 4.16 : *matplotlib Logo* *65*

Figure 4.17 : *Numpy Logo* *66*

Figure 4.18 : *Google colab Logo* *66*

Figure 4.19 : *Our machine capabilities* *67*

List of tables

Table 1.1 Classification of the noiseless BBOB functions based on their properties *10*
Table 3.1: Table that show the dataset composition. 26
Table 3.2: Data split table 37
Table 4.1: The results obtained on training the ASM 45
Table 4.2: Table for the encoded classes values. 49
Table 4.3: Represent the confusion matrix for the ASM 1. 50
Table 4.4: Represent the confusion matrix for the ASM 2. 51
Table 4.5: Summary of the relative expected runtime of the 10 portfolio solvers 55
Table 4.6: confusion matrix for the testing of the ASM against the SBS 56

List of abbreviations

ML: Machine learning

DL: Deep learning

ELA: Exploratory Landscape Analysis

ASP: Algorithm Selection Problem

AAS: Automated Algorithm Selection

GECCO: Genetic and Evolutionary Computation Conference

ANN: Artificial neural network

SBS: Single best solver

BBOP: Black-Box Optimization Problem

BBOB: Black-Box Optimization Benchmark

COCO: COmparing Continuous Optimizers

NaN: Not a Number

SMOTE: Synthetic Minority Over-sampling Technique

VBS: Virtual best solver

LHS: Latin hypercube sampling

FID: Function Identifier

IID: Instance Identifier

ERT: Expected Run Time

RELERT: Relative Expected Runtime

ReLu: Rectified Linear Unit

ASM: Algorithm selection model

AS: Algorithm selection

Abstract

Selecting the most appropriate algorithm to use when tackling a black-box continuous optimization problem is a challenging task. with a wide range of optimization algorithms being benchmarked each year and easy access to black-box optimization functions, primarily due to the efforts of the COCO platform, an automated method for algorithm selection in single-objective black-box optimization problems has become necessary. In this dissertation, we have chosen to employ DL technology in combination with ELA techniques to predict the optimal solver (SBS) for any given problem set. The performance of the proposed DL-based model is superior to a framework based on machine learning algorithms when applied to a continuous black-box optimization problem set, showcasing its effectiveness.

Key words:

Algorithm Selection, Black-Box Optimization, Exploratory Landscape Analysis, Deep Learning,

Single-Objective Continuous Optimization, COCO framework

Abstract

Résumé

Sélectionner l'algorithme le plus approprié pour résoudre un problème d'optimisation continu à boîte noire est une tâche complexe. Avec une gamme étendue d'algorithmes d'optimisation évalués annuellement et un accès facilité aux fonctions d'optimisation à boîte noire, principalement grâce aux efforts de la plateforme COCO, une méthode automatisée de sélection d'algorithmes pour les problèmes d'optimisation à boîte noire à objectif unique est devenue nécessaire. Dans ce memoire, nous utilisons la technologie Deep Learning combinée à des techniques d'Exploratory Landscape Analysis pour prédire le solveur optimal (SBS) pour n'importe quel ensemble de problèmes donné. Les performances du modèle proposé basé sur le Deep Learning surpassent celles d'un modèle basé sur des algorithmes d'apprentissage automatique lorsqu'il est appliqué à un ensemble de problèmes d'optimisation à boîte noire continu, démontrant ainsi son efficacité.

Mots-clés : Sélection d'algorithmes, Optimisation à boîte noire, Analyse du paysage exploratoire, Apprentissage profond, Optimisation continue à objectif unique, plateforme COCO.

ملخص

يعد اختيار الخوارزمية الأنسب لاستخدامها عند التعامل مع مشكلة تحسين مستمرة غير معروفة مهمة صعبة. مع وجود مجموعة واسعة من خوارزميات التحسين التي يتم تقييمها كل عام والوصول السهل إلى دوال التحسين غير المعروفة، وبشكل ، أصبح من الضروري وجود طريقة تلقائية لاختيار الخوارزمية في مشاكل التحسين COCO أساسي بفضل جهود منصة بالتزامن مع تقنيات تحليل (DL) غير المعروفة أحادية الهدف. في هذه الأطروحة، اخترنا استخدام تقنية التعلم العميق لأي مجموعة من المشاكل. أداء النموذج المقترح القائم على (SBS) للتنبؤ بالمحلل الأمثل (ELA) خصائص المشكلات التعلم العميق يتفوق على إطار العمل القائم على خوارزميات التعلم الآلي عند تطبيقه على مجموعة من مشاكل التحسين ..المستمرة غير المعروفة، مما يبرز فعاليته

الكلمات الرئيسية:

اختيار الخوارزمية، الأمثلة ذات الصندوق الأسود، تحليل المشهد التجريبي، التعلم العميق، الأمثلة المستمرة ذات الهدف الفردي، منصة COCO

General introduction

In continuous optimization problems, selecting the most suitable algorithm plays a vital role in achieving effective and efficient solutions. These problems are very common in scientific and engineering fields, often possess complex landscapes characterized by numerous local optima points, non-linearity, and high dimensionality. Traditionally, practitioners relied on manual selection from a variety of available algorithms, which proved challenging and prone to suboptimal choices as the complexity increased.

It was at this time that the interest in machine learning as a very promising solution to algorithm selection gained momentum, focusing mainly on statistical learning and traditional machine learning models to guide decision-making. For quite a long time, meta-learning was the favorite among a lot of fields due to the insight it gave into the optimal choices of algorithms. Thus, as the number of benchmarked optimization algorithms increased and black-box optimization functions were supplied by platforms like COCO, an automated algorithm selection method for single-objective black-box optimization problems became realized.

In recent years, DL was able to gain strength in modeling higher-level abstractions using higher complexity DNN architectures. Having seen success in many fields, the avenues of automating the algorithm selection process open up. On the other hand, DL models are good at taking hold of complex patterns and representations from data and hence projecting problem-specific characteristics in a critical way to predict algorithm performance, this data-driven approach has huge potential for revolutionizing the field and eventually turns automated Algorithm Selection into a classification-based model.

This thesis focuses on the development of an automated algorithm selection model for black-box optimization problems. The core objective is to address the challenges practitioners encounter in selecting the best algorithm out of a pool. This will be achieved by using Deep Learning technology combined with ELA techniques within the framework for developing a robust efficient model that can predict an optimal algorithm—a Single Best Solver—against the given black-box optimization problem sets.

By automating the process of algorithm selection for continuous optimization problems, we can unlock several key benefits. First, it will increase efficiency. On the other hand, it reduces

the burden of manual effort. Additionally, automated algorithm selection has the potential to produce better optimization results via the power of Deep Learning models, which will include problem specific characteristics with the help of Exploratory Landscape Analysis techniques (ELA). This data-driven approach enables us to make informed predictions about algorithm performance, increasing the likelihood of identifying the optimal solver for a given black-box optimization problem.

This manuscript is structured as follows:

Chapter 1 this chapter delves into the fundamental concepts of Black-Box Optimization Problems (BBOP), encompassing critical principles, a variety of architectural strategies, and widely used frameworks integral to the development and implementation of deep learning solutions. By examining these foundational elements, we aim to provide a comprehensive understanding of how BBOPs operate.

Chapter 2 explores the core concepts for Algorithm Selection Problem (ASP) and its characteristics, architectural approach, as well presenting the Exploratory Landscape Analysis (ELA) technique and its role in optimizing the algorithm selection.

In **Chapter 3**, we outline the steps involved in the implementation of the Deep learning model and briefly the data set and evaluation methods.

In **Chapter 4**, we present the results obtained along with the discussion over said results and the application made for our work

Chapter 1

Black-Box Optimization Problem (BBOP)

Introduction:

We call an optimization problem as black-box optimization (BBO) when the objective function is not defined, understood, complex or expensive to evaluate and there is no explicit mathematical formulation. These types of problems are found widely in various fields such as engineering, design, machine-learning hyperparameters tuning, and scientific modeling. traditional optimization techniques generally fail in BBO problems thereby necessitating advanced methodologies and benchmarks for algorithm performance assessment.

1.1 History and development

The concept of black-box optimization has changed over time significantly. Initially, these problems were addressed using simple heuristic methods. However, the complexity of real-world applications that came into play indicated a need for advanced algorithms.

Consequently, several optimization algorithms like Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Differential Evolution (DE) and others were developed. Black-Box Optimization Benchmarking (BBOB) originated from the necessity to systematically evaluate and compare such algorithms thus necessitating a standardized benchmarking platform. It is now an important research tool where researchers and practitioners have access to a variety of test functions as well as performance measures.

1.1.1 What is Benchmarking?

In the field of algorithm selection, benchmarking involves analyzing and comparing how different algorithms perform on problem instances that are not common, in identifying which is the best performing algorithm in different conditions and developing a strong method for predicting which algorithm is most appropriate for a given problem, its main objective is to test as many algorithms as possible.

Benchmarking is typically conducted following the next steps:

1. **Problem Instance Collection**

- ✓ **Diverse Dataset**: Collect a large and diverse set of problem instances that represent the range of difficulties and characteristics the algorithms will encounter in real-world scenarios.

- ✓ **Representation**: Ensure that the problem instances are well-represented across different domains, scales, and types of problems.

2. **Choosing Algorithms**

- ✓ **Variety of Algorithms**: Choose a diverse set of algorithms to benchmark, they should include different types and approaches to solving the problem at hand.
- ✓ **Parameter Settings**: Test each algorithm with various parameter settings to capture a wide range of performance outcomes.

3. **Performance Metrics**

- ✓ **Accuracy and Efficiency**: Measure both the accuracy (quality of the solution) and efficiency (computation time, resource usage) of each algorithm.
- ✓ **Other Metrics**: Depending on the problem domain, other metrics like robustness, scalability, and stability might also be crucial.

4. **Evaluation Method**

- ✓ **Cross-validation**: Use cross-validation techniques to ensure that the performance evaluation is robust and not biased by the particular set of problem instances.
- ✓ **Statistical Analysis**: Employ statistical methods to analyze the performance data, identifying significant differences and trends among the algorithms.

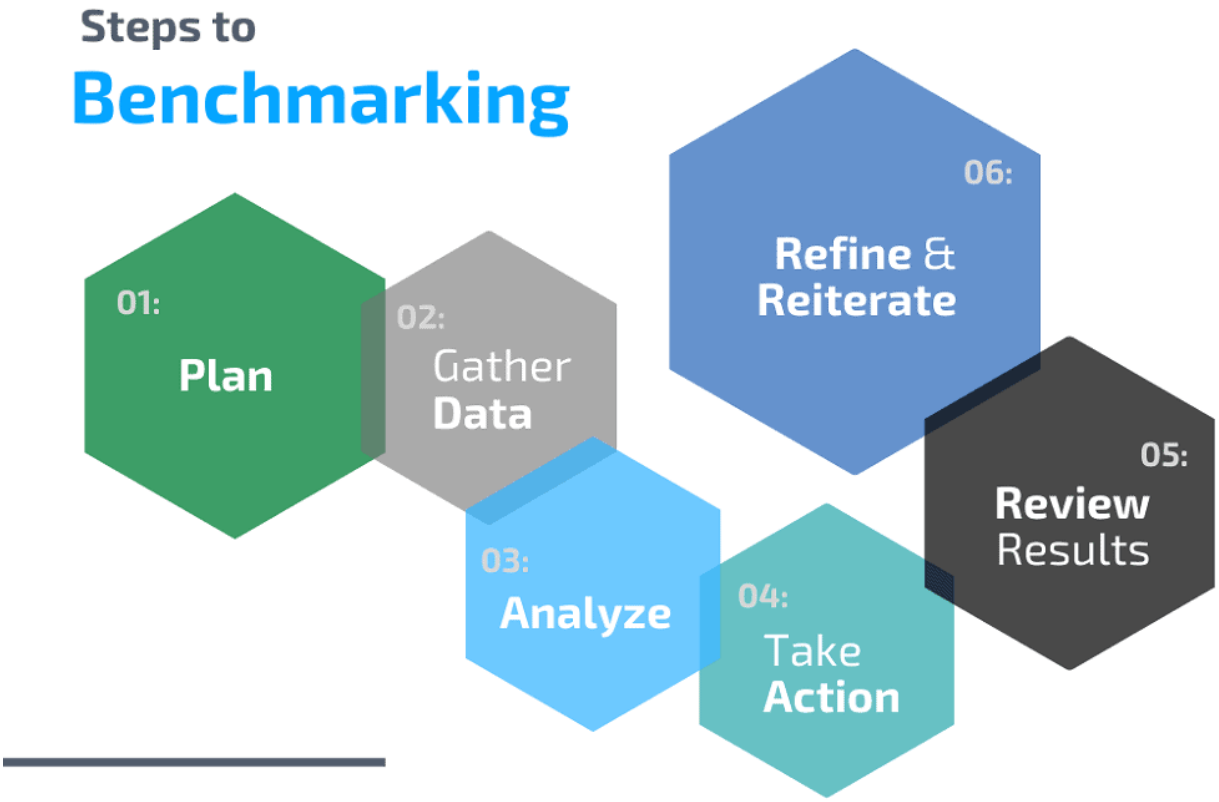
5. Benchmarking Frameworks

- ✓ **Standardized Frameworks**: Use standardized benchmarking frameworks and tools to ensure consistency and reproducibility of the results. Examples include the Algorithm Selection Library (ASlib) and related tools.
- ✓ **Community Benchmarks**: Engage with the research community to compare results and improve benchmarking practices through shared datasets and methodologies.

6. Reporting and Analysis

- ✓ **Detailed Reports**: Provide detailed reports and visualizations of the benchmarking results, highlighting the strengths and weaknesses of each algorithm.

- ✓ **Interpretability**: Ensure that the results are interpretable and actionable, providing insights into why certain algorithms perform better on specific types of problem instances.



Figure(1.1): Benchmarking process plan

Benchmarking in algorithm selection is crucial for understanding the landscape of algorithm performance and for developing reliable methods to automatically select the best algorithm for a given problem instance. It involves a careful and systematic approach to evaluating algorithms, extracting meaningful insights, and building predictive models that can guide algorithm selection in practice.

1.2 BBOB Framework

Black-box Optimization Framework is a term used to refer to optimization problems with objective functions that are either unknown or too complicated to explicitly state. Here, the objective function will not have an explicit analytical form. instead, it is known as “black box” where one evaluates it at specific points but does not know about its internal details.

The task of solving black-box optimization (BBOP) problems presents considerable challenges. These challenges primarily arise from the lack of an analytical form, the absence of derivative information, and the substantial computational expense involved in numerically approximating derivatives.

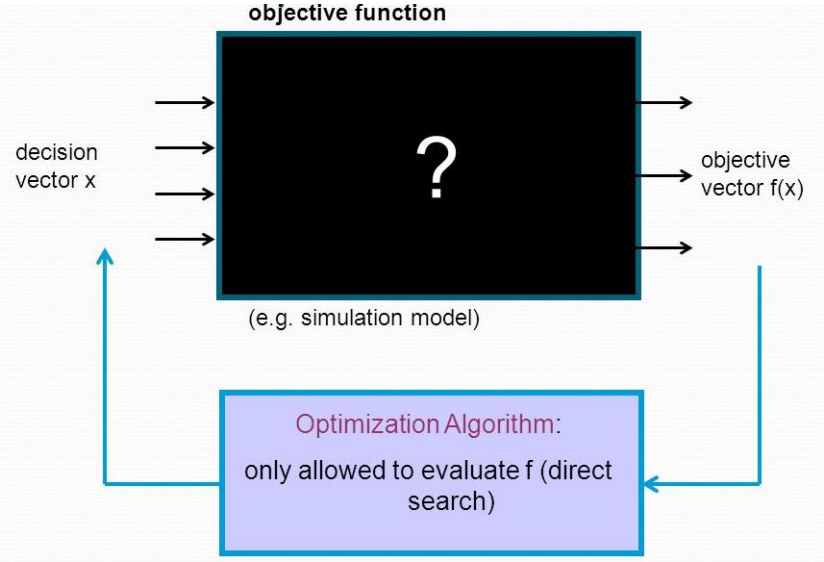


Figure (1.2) : Schema for black box optimization framework

1.2.1 Characteristics of Black-Box continuous Optimization Problems

The main characteristics of black-box continuous optimization problems are:

1.Lack of mathematical expression: Black-box problems often do not have a mathematical model that describes the relationship between the input variables and the objective function. The objective function is typically evaluated through black-box evaluations, where the function's behavior is observed through input-output samples without explicit knowledge of its internal workings.

2.Absence of Derivative Information: In black-box optimization, derivative information such as gradients or Hessians of the objective function is typically unavailable or computationally expensive to obtain.

3.Computational Cost: Evaluating the objective function in black-box optimization problems can be computationally expensive

4.Noisy and Stochastic Behavior: Black-box objective functions may exhibit noise or stochasticity, meaning that repeated evaluations of the same input may yield slightly different function values.

5.Global Optimization Challenges: Black-box continuous optimization problems often involve searching for the global optimum in a high-dimensional search space. The presence of multiple local optima and complex landscape structures makes finding the global optimum a challenging task.

6.Limited Problem-Specific Knowledge: In black-box optimization, there is limited or no prior knowledge about the problem structure, properties, or constraints. The optimization algorithms must rely solely on the evaluations of the objective function to guide the search process, making it crucial to design adaptive and intelligent search strategies.

1.3 COCO Framework

One of the most prominent initiatives in the field of black-box optimization benchmarking is the Comparing Continuous Optimizers (COCO) platform. COCO is an open-source platform designed to provide rigorous and standardized benchmarks for continuous optimization problems. It offers a wide range of test functions, performance indicators, and tools for analyzing algorithm performance.

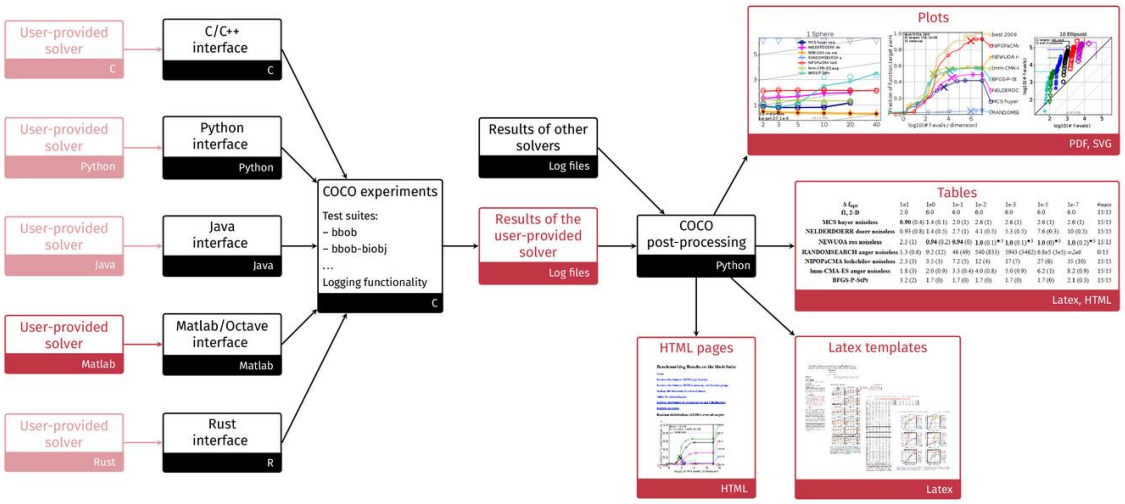


Figure (1.3): COCO project general structure

COCO benchmarking platform has contributed immensely to the field of optimization algorithm benchmarking by providing standardized test functions where it provides a diverse set of test functions that mimic real-world optimization challenges. These functions vary in terms of dimensionality, modality, separability, and ruggedness, offering a comprehensive evaluation environment.

The platform includes well-defined performance metrics such as runtime distributions, convergence rates, and success rates, allowing for detailed analysis and comparison of algorithms and on top of that COCO supports automated benchmarking, enabling researchers to evaluate their algorithms efficiently and consistently.

1.4 BBOB Test Suites

The blackbox optimization benchmarking (BBOB) test suite is COCO's standard test suite with 24 noiseless, single-objective and scalable test functions. Each function is provided in dimensions (2, 3, 5, 10, 20, 40) and in 15 instances, however also available for arbitrary dimensions and number of instances.

1.4.1 Benchmark Functions

The Black-Box Optimization Benchmarking (BBOB) suite consists of 24 single-objective noiseless functions which are categorized into five groups based on their characteristics. This suite allows for the creation of different problem instances, making it ideal for evaluating the

stability and invariance of optimization algorithms under various transformations. This feature is particularly useful in testing the robustness of algorithms by applying different transformations to the same problem, ensuring that the results are consistent and reliable across different scenarios [3]. A brief description of each group and the functions in the table

Group	Functions	Characteristics
1. Separable Functions	f1: Sphere Function f2: Separable Ellipsoidal Function f3: Rastrigin Function f4: Büche-Rastrigin Function f5: Linear Slope	separable and can be optimized independently.
2. Functions with low or moderate conditioning	f6: Attractive Sector Function f7: Step Ellipsoidal Function f8: Rosenbrock Function (original) f9: Rosenbrock Function (rotated)	low or moderate conditioning, meaning the curvature of the function changes gradually.
3. Functions with high conditioning and unimodal	f10: Ellipsoidal Function f11: Discus Function f12: Bent Cigar Function f13: Sharp Ridge Function f14: Different Powers Function	high conditioning and unimodal, meaning they have a single global optimum.
4. Multi-modal functions with adequate global structure	f15: Rastrigin Function f16: Weierstrass Function f17: Schaffer's F7 Function f18: Schaffer's F7 Function f19: Composite Griewank-Rosenbrock Function F8F2	multi-modal, meaning they have multiple local optima, but they still have a clear global structure.
5. Multi-modal functions with weak global structure	f20: Schwefel Function f21: Gallagher's Gaussian 101-me Peaks Function f22: Gallagher's Gaussian 21-hi Peaks Function f23: Katsuura Function f24: Lunacek bi-Rastrigin Function	multi-modal and have weak global structure, meaning the local optima are not clearly related to the global optimum.

Table 1.1: Classification of the noiseless BBOB functions based on their properties (multi-modality, global structure, separability, variable scaling, homogeneity, basin-sizes, global to local contrast))

1.4.2 Evaluation metrics for algorithm performance:

The choice of evaluation metrics depends on the specific optimization problem, the algorithm being evaluated, and the desired characteristics of the solution.

Several evaluation metrics are commonly used to assess the performance and effectiveness of different algorithms. Some of the key evaluation metrics are:

1.**Convergence**: It measures how speed an algorithm reaches a satisfactory solution or approaches the optimal solution. It typically assesses the rate of improvement over iterations or generations. Faster convergence indicates more efficient performance.

2.**Fitness/Objective Function Value:** Fitness or objective function value represents the quality of the solution obtained by an algorithm. It measures how well an algorithm performs on the objective function being optimized.

3.**Solution Quality**: Solution quality evaluates the optimality or closeness to the global optimum achieved by an algorithm. It compares the obtained solution with known or benchmark solutions to assess the accuracy and effectiveness of the algorithm. Metrics such as distance to the global optimum or error rate can be used to measure solution quality.

4.**Efficiency/Computational Time**: Efficiency metrics focus on the computational resources required by an algorithm to find a solution. It calculates the time taken to reach a solution, the number of function evaluations performed, or the computational complexity.

1.5 Importance of Benchmarking

Benchmarking is essential in the field of black-box optimization for several reasons:

Algorithm Comparison: Benchmarking allows for systematic comparison of different optimization algorithms under standardized conditions. This helps identify the most effective algorithms for specific types of problems.

Performance Evaluation: Through benchmarking, researchers can evaluate the performance of their algorithms in terms of convergence speed, solution quality, robustness, and scalability.

Algorithm Development: Insights gained from benchmarking can guide the development of new algorithms and improvements to existing ones. Understanding where current methods fall short can inspire novel approaches.

Reproducibility: Standardized benchmarks ensure that results are reproducible and comparable across different studies and research groups. This fosters collaboration and cumulative progress in the field.

Conclusion

Black-box optimization benchmarking, especially through platforms like COCO and its BBOB test suites, has become pivotal in the research and development of optimization algorithms. These platforms offer a standardized, rigorous evaluation environment, enabling researchers to methodically compare and refine optimization techniques. This systematic approach drives advancements in tackling complex real-world challenges. As the field progresses, benchmarking will continue to be essential in enhancing our understanding and capabilities in black-box optimization.

Looking ahead, the field is guided toward further growth with developments in machine learning, reinforcement learning, and meta-heuristic approaches, promising even more advanced solutions to black-box optimization challenges. For researchers and practitioners, mastering these methodologies and their applications provides robust tools for efficiently solving complex real-world problems.

Chapter 2

The Algorithm Selection Problems (ASP)

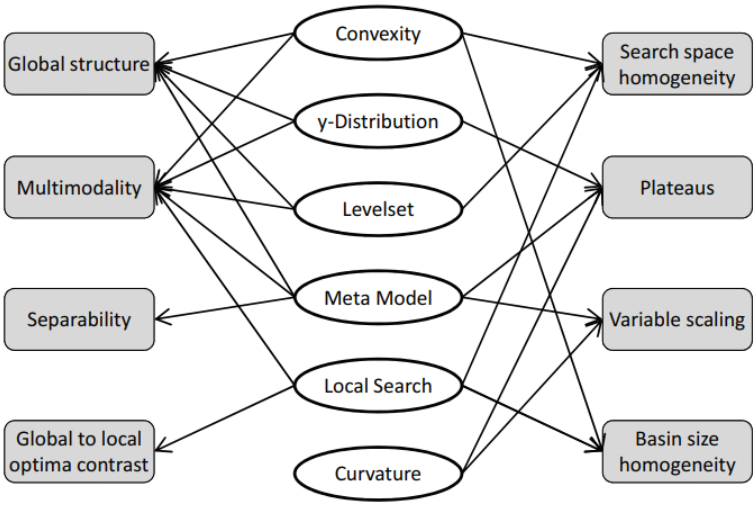
Introduction

For computationally challenging problems, there is a plethora of algorithms available, each with its own benefits and limits when applied to different instances of these problems. As a result, continuously, the question of determining the most suitable algorithm for a specific problem instance to achieve optimal performance is arisen. The field of algorithm selection aims to address this question by devising decision policies known as algorithm selectors. These selectors provide recommendations on which algorithm should be chosen for a given problem instance

In this chapter, the Algorithm Selection Problem will be inquired trough, we introduce the concept of ELA technique, which is widely regarded as the sole valid approach for measuring the problem characteristics of Black Box Continuous Optimization Problems. Additionally, we explore the challenges associated with algorithm selection and highlight relevant research conducted in this field and popular methods used for solving the ASP problem.

2.1Exploratory Landscape Analysis (ELA) for characterizing problem landscapes

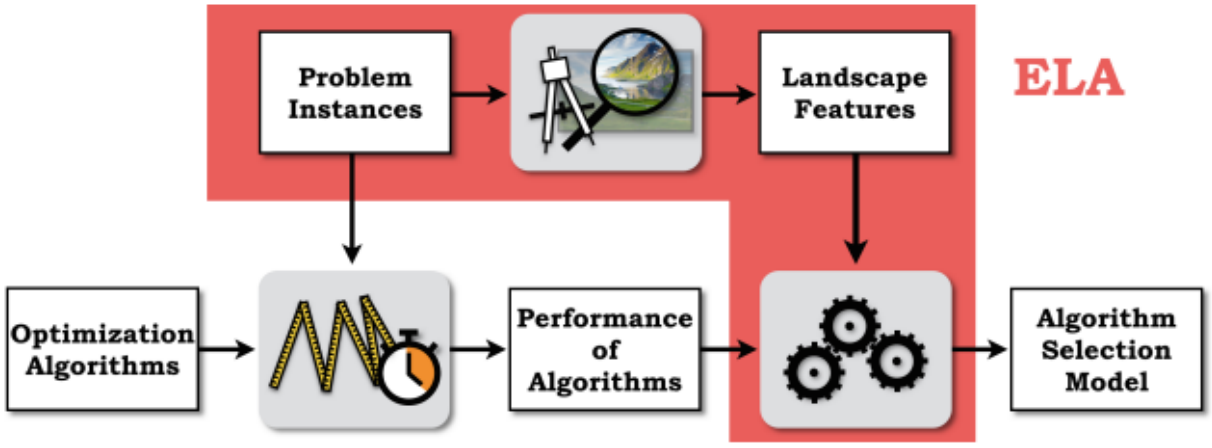
Exploratory Landscape Analysis (ELA) is a sophisticated approach to characterize a landscape of a problem by means of numerical feature values introduced by [2].
In its original version, ELA covered a total of 50 features - grouped into six so-called low-level properties (Convexity, Curvature, y-Distribution, Levelset, Local Search and Meta Model). These (numerical values) were used to characterize (usually categorical and expert-designed) high-level properties, such as the Global Structure, Multimodality or Variable Scaling. The **figure (2.1)** below visualizes the connections between the low- and high-level properties.



Figure(2.1) : Relationships between high-level features (grey) and low-level feature classes (white)

[1] argued that the introduction of the low-level features can be seen as a major step towards automatically computable landscape features and hence automated algorithm selection .

The **figure (2.2)** below illustrates the application of ELA to enhance the process of automated algorithm selection.



figure(2.2) : Schematic view of how (ELA) can be used for improving the automated algorithm selection process.

2.3 Evaluation metrics for algorithm performance:

The choice of evaluation metrics depends on the specific optimization problem, the algorithm being evaluated, and the desired characteristics of the solution

several evaluation metrics are commonly used to assess the performance and effectiveness of different algorithms. Some of the key evaluation metrics are:

1.**Convergence**: It measures how speed an algorithm reaches a satisfactory solution or approaches the optimal solution. It typically assesses the rate of improvement over iterations or generations. Faster convergence indicates more efficient performance.

2.**Fitness/Objective Function Value:** Fitness or objective function value represents the quality of the solution obtained by an algorithm. It measures how well an algorithm performs on the objective function being optimized.

3.**Solution Quality**: Solution quality evaluates the optimality or closeness to the global optimum achieved by an algorithm. It compares the obtained solution with known or benchmark solutions to assess the accuracy and effectiveness of the algorithm. Metrics such as distance to the global optimum or error rate can be used to measure solution quality.

4.**Efficiency/Computational Time**: Efficiency metrics focus on the computational resources required by an algorithm to find a solution. It calculates the time taken to reach a solution, the number of function evaluations performed, or the computational complexity.

2.6 Algorithm selection problem:

In Algorithm selection [4], the goal is to find the best Algorithm A_i from a set of candidate Algorithms $\{A_1, A_2, \dots, A_K\} = A$ for a problem instance $I \in I$ from a problem instance space I. formally, the aim is to find a mapping, called algorithm selector s:$I \rightarrow A$, which maximizes a costly-to-evaluate performance measure m:$A \; x \; I \rightarrow R$. the optimal selector is defined as

$$s^*(I) \in \arg\max_{A \in \mathcal{A}} \mathbb{E}[m(A, I)]$$

2.4 Algorithm selection framework

In [4], the components of the algorithm selection framework are interconnected and have mutual influence on each other. Together, they form a comprehensive approach for selecting the best algorithm for a specific problem instance. This ensures that the chosen algorithm not only matches the problem requirements but also exhibits strong performance in terms of the desired evaluation metrics. **Figure (2.3)** displays the diagram of the algorithm selection framework.

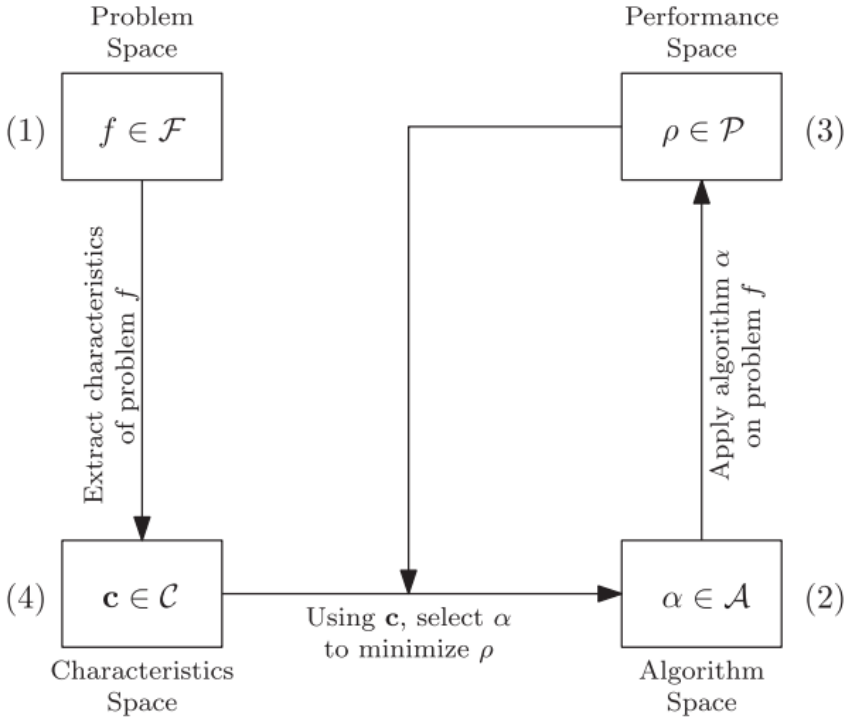


Figure (2.3): Diagram of the algorithm selection framework proposed by Rice in his 1976 paper

The diagram consists of four components: the problem, algorithm, algorithm selection, and performance evaluation. Each component plays a crucial role in the algorithm selection process.

- **Problem Space**: represents the set of all possible problem instances that the algorithm selection framework aims to address. It encompasses a range of optimization problems, each with its distinct characteristics, requirements, and objectives. [3] declares that ELA methods, are the only valid approach to measure the problem characteristics for Black box Continuous optimization Problems

- **Performance Space**: Refers to the set of performance metrics used to evaluate the effectiveness and quality of the algorithms in solving the problem instances. It can include convergence rate, solution accuracy, robustness, … The performance space provides a quantifiable way to assess and compare the performance of different algorithms.
- **Algorithm Space**: Denotes the collection of candidate algorithms (Algorithm Portfolio) available for solving the optimization problems in the problem space. It encompasses various algorithms with different characteristics
- **Characteristics Space:** Represents the set of algorithm properties to describe and distinguish the algorithms in the algorithm space. These characteristics can include algorithm complexity, search strategy, population size, mutation rate, selection mechanism, or any other relevant attributes of the algorithms. The characteristics space helps in understanding the strengths and weaknesses of different algorithms and their suitability for solving specific problem instances.

2.5 Algorithm Selection Approaches

Since the topic of AS is around for quite some time, naturally, a lot of different approaches to AS have been suggested in the literature.

In the following, a taxonomy of different algorithm selection solutions based on the type of learning problem presented by [5]. A visual overview of the taxonomy is presented in **Figure (2.4).**

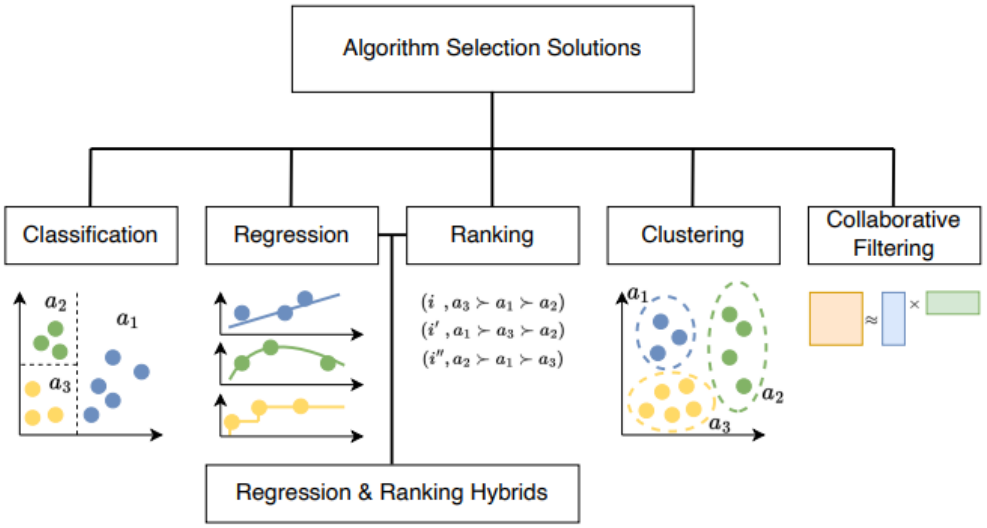


Figure (2.4) Taxonomy of different algorithm selection solutions [5]

- **Classification Solutions**

Classification-based algorithm selection approaches work by directly trying to learn the by the means of classification instead of learning a surrogate loss function. There exist various approaches in the literature, which directly follow this scheme using different kinds of machine learning models to model h such as decision trees [6] [7], k-nearest neighbor models [8] or support vector machines [9].

- **Regression Solutions**

Instead of treating the AS problem as a classification problem, it is treated as a multi-target regression problem with one regression target for each algorithm. A large set of literature exists, which suggests methods leveraging regression models to predict the performance of an algorithm on an instance [10, 15, 18, 19].

- **Ranking Solutions**

Ranking-based AS solutions are motivated by the observation that the ability to predict a correct ranking across the algorithms for a given instance is potentially an ability easier to achieve than the ability to correctly predict the true loss function value for each algorithm. As examples, work of [24] [1] and [26].

- **Hybrids of Ranking and Regression Solutions**

This idea was proposed by [27] and was later refined by [32]

- **Clustering Solutions**

They decompose the algorithm selection on the complete instance space to multiple AS problems. Examples following this strategy are [33], [36], [37].

- **Collaborative Filtering Solutions**

Treating the AS problem as a recommendation problem similar to, for example, recommending products (i.e., algorithms) to customers (i.e. instances). As examples, [38] and [39; 42].

2.6 Algorithm Selection Strategies

Static vs. Dynamic Selection

- **Static Selection**: In static selection, the algorithm is chosen based on the initial problem instance characteristics, and the choice remains fixed throughout the solution process.
- **Dynamic Selection**: In dynamic selection, the algorithm choice can change during the solution process based on intermediate results and evolving instance characteristics.

Portfolio Approaches

Portfolio approaches combine multiple algorithms to handle a variety of problem instances. Key strategies include:

- **Algorithm Portfolios**: A set of algorithms is maintained, and the best algorithm for each new instance is selected based on past performance data.
- **Hybrid Algorithms**: Algorithms are combined in a way that leverages the strengths of each. For example, a hybrid optimization algorithm might use a genetic algorithm for global search and a local search algorithm for fine-tuning solutions.

2.7 Adaptive and Online Learning

Adaptive and online learning strategies continuously improve algorithm selection over time:

- **Adaptive Learning**: The selection strategy adapts based on feedback from previous performance, refining its predictions and choices.
- **Online Learning**: The selection model updates in real-time as new problem instances are solved, ensuring that the model remains current and effective.

2.8 Related work for algorithm selection in black-box continuous optimization

In the field of continuous black-box optimization, have been proposed and have shown promising results. The following are some related works in case of single-objective continuous black-box optimization.

- [1] proposed a sophisticated machine learning techniques combined with informative exploratory landscape features. This work provides an extensive study on feature-based algorithm selection in case of single-objective continuous black-box optimization.

Three classes of supervised learning strategies were considered for training the algorithm selection models: A classification approach, a regression approach, and pairwise regression.

- [55] explored feature-free approaches that leverage advanced deep learning techniques (convolutional Reduced Multi-Channel conv-rMC) applied to either images or point clouds. they demonstrate that point-cloud-based approaches are competitive and significantly decrease the size of the solver portfolio needed.

- [56] introduced. the CNN-HT two-stage algorithm selection framework is In the initial stage, a Convolutional Neural Network (CNN) is utilized to classify problems. Subsequently, in the second stage, the Hypothesis Testing (HT) technique is employed. They adopted Exploratory Landscape Analysis (ELA) features of the problem as the input and utilize feature selection techniques to reduce the redundant ones.

- [57] proposes an algorithm selector (AS), capable of selecting a promising configuration of a modularized version of the most prominent meta-heuristic CMA-ES for single-objective continuous optimization. They construct a two-dimensional gray-scale picture which is called fitness map. Using this fitness map as input, they construct an algorithm selector based on a Deep CNN. The algorithm selector is a collection of different distinct CMA-ES versions provided by the modular CMA-ES framework.

Conclusion

In this chapter, we first provided an explicit definition of Algorithm Selection problem, then emphasized on its importance to improve performance and efficiency over different applications.

Then, we explored both theoretical and practical aspects of algorithm selection by discussing factors influencing algorithm choice. Different continuous problems where algorithm selection plays a vital role were also explained together with criteria for assessing effectiveness of algorithms.

The next chapter will be more pragmatic than theory as it explains our proposed model in details. Our development process will be presented including design options, strategies in implementation as well as rational behind our technical approach.

Chapter 3

A Deep Learning Approach for Algorithm Selection in Numerical BBOP

Introduction

This chapter outlines the research methods used in our study, detailing each step taken. We begin with an overview of our proposed approach, which integrates Algorithm Selection on continuous Black-Box optimization problems with deep learning instead of a traditional machine learning approach.

Next, we discuss data collection, focusing on selecting high-performing solvers from the COCO platform's optimization algorithm results. Our goal is to create a model that surpasses any single solver's performance. To achieve this, we expanded our dataset through data augmentation to ensure we have sufficient data for the training and testing phases.

We then move on to feature engineering, where we employ Exploratory Landscape Analysis (ELA) to characterize problem landscapes numerically. We describe the preprocessing steps applied to the raw data and the network architecture before proceeding with model training.

In this section, we provide details about the construction and training of the network architectures and present our results. Our key contribution lies in combining ELA techniques with deep learning to predict the optimal outcomes for single-objective noiseless problems. This approach yields superior results while requiring fewer computational resources.

3.1 The Dataset

The dataset is composed of two main separate bodies. The first body concerns the performance of the portfolio of algorithms we chose to train the model on. We begin by defining the Algorithm Selection Problem (ASP) as follows: Given a set of optimization algorithms A, which represents our portfolio of algorithms, and a set of problem instances I, our objective is to find a model m: $I \rightarrow A$ that selects the best algorithm $a \in A$ from the portfolio for an unseen problem instance $i \in I$.

Although there exists a plethora of competent algorithms, even when considering only single-objective continuous optimization problems, none of them can be considered a superior choice over all others across all optimization problems. Therefore, it is still relevant to find a sophisticated mechanism to select the optimal solver for each problem.

The dataset is composed of 2 main separate bodies one is for the performance of the portfolio of algorithms we chose to train the model on, we start by defining the ASP as follows: given a set of optimization algorithms A which represents our portfolio of algorithms, and a set of problem instances I our objective is to find a model m: $I \rightarrow A$ that selects the best algorithm $a \in A$ from the portfolio for an unseen problem instance $i \in I$, although there exist a plethora of competent algorithms even when we only specify the single-objective continuous optimization problems, none of them can be considered as superior choice over other ones across all the optimization problems so it's still relevant to find a sophisticated mechanism to select the optimal solver to each problem.

The performance data remained static over the years as a setup in the COCO-platform, specifically in terms of dimensions (D) ∈ {2, 3, 5, 10, 20, 40} for the benchmarked functions. Each algorithm is evaluated on the functions (FID) ∈ {1, …,24}, and the number of instances (IID) ∈ {1, ...,15} [1]. The data contains a log of the performed number of function evaluations and the achieved fitness value. This information allows us to determine if the solver successfully found a global optimum for the problem instance i ∈ I within a proximity of ε = 10^-2. Additionally, it provides insights into the number of function evaluations ($FE_i(\varepsilon)$) performed by the solver until termination, irrespective of success or failure.

For our work, we consider various data collection settings based on the combinations of problem dimensionality. Specifically, we utilize the domain D = {2, 3, 5, 10} to represent different dimensional versions of the benchmark functions.

We can determine the success of a solver by considering the precision domain of ε. In other words, we define Success (ε) = 1 if the solution found, denoted as x*, lies within the range [-5, +5]d and its fitness value falls within [f(xopt), f(xopt) + ε], where xopt represents the global optimum. To calculate the ERT (Expected Runtime to Target), we consider both the success of the solver and the number of function evaluations.

The performance results obtained by running an algorithm on the COCO platform provide us with detailed fitness value calculations and graphical representations for all the problem instances. We utilize these calculations to determine the Expected Runtime (ERT) for each pair of Algorithm A and function F on dimension D. The ERT is calculated using the following formula:

$$ERT(\varepsilon) = \frac{\sum i\ FE_i(\varepsilon)}{\sum i\ Success_i(\varepsilon)}$$

We proceed by comparing the Expected Runtime (ERT) of different algorithms with the best algorithm found, known as the Success-Based Selection (SBS), on the same function. This comparison is done using the Relative Expected Runtime (RELERT) formula, which is provided below.

$$RELERT_{A,F,D} = \frac{ERT_{A,F,D}}{Min(ERT_{F.D})}$$

Our final dataset for training our model is created by merging the features and performance data during the preprocessing stage. This dataset is obtained by combining the performance of the algorithm portfolio with the feature set collected from the Flacco GUI [28]. The dataset is represented in **Table 3.1**, which displays the merged data.

Dataset	Number of Rows	Number of features
dataset 1	480	151

Table (3.1): Table that show the dataset composition.

3.2.1 Performance:

The performance data for each optimization algorithm is organized into columns, with instances separated by the "%" symbol, as outlined below:

The 1st column indicates the problem instance selected by the algorithm.

The 2nd column displays the function evaluation.

The 3rd column represents the best noise-free fitness - Fopt.

The 4th column represents the best noise-free fitness, -Fopt, for the entire instance.

The 5th column displays the measured fitness.

The 6th column represents the best measured fitness.

Starting from the 7th column and onward, the columns contain the values of the variables (x) that define the instance. The number of x values depends on the dimension of the problem. Figure 3.1 provides a sample of the preprocessed performance data for dimension 2.

```
=== 2..txt ===
% f evaluations | best noise-free fitness - Fopt | noise-free fitness - Fopt (7.948000000000e+01) | measured fitness | b
1  +2.548460290e+01 +2.548460290e+01 +1.049646029e+02 +1.049646029e+02 -3.5054e+00 +2.2137e+00
2  +1.361478386e+00 +1.361478386e+00 +8.084147839e+01 +8.084147839e+01 +2.9566e-01 -2.3228e+00
4  +9.605204258e-01 +9.605204258e-01 +8.044052043e+01 +8.044052043e+01 +2.5824e-01 -2.1368e+00
61  +1.096345213e-01 +1.096345213e-01 +7.958963452e+01 +7.958963452e+01 +4.9262e-01 -9.2850e-01
179  +7.100934711e-02 +7.100934711e-02 +7.955100935e+01 +7.955100935e+01 +4.3536e-01 -9.6269e-01
257  +2.576151887e-02 +2.576151887e-02 +7.950576152e+01 +7.950576152e+01 +1.7651e-01 -1.0156e+00
282  +2.183563286e-02 +2.183563286e-02 +7.950183563e+01 +7.950183563e+01 +1.1642e-01 -1.2137e+00
439  +1.626385826e-02 +1.626385826e-02 +7.949626386e+01 +7.949626386e+01 +2.1124e-01 -1.0362e+00
524  +1.609451209e-02 +1.609451209e-02 +7.949609451e+01 +7.949609451e+01 +3.7460e-01 -1.1923e+00
571  +1.578960318e-02 +1.578960318e-02 +7.949578960e+01 +7.949578960e+01 +3.0720e-01 -1.2701e+00
612  +9.875797017e-03 +9.875797017e-03 +7.948987580e+01 +7.948987580e+01 +1.5806e-01 -1.1868e+00
721  +1.334365585e-03 +1.334365585e-03 +7.948133437e+01 +7.948133437e+01 +2.1632e-01 -1.1548e+00
763  +6.112868803e-04 +6.112868803e-04 +7.948061129e+01 +7.948061129e+01 +2.3351e-01 -1.1723e+00
1159  +5.834247689e-04 +5.834247689e-04 +7.948058342e+01 +7.948058342e+01 +2.6746e-01 -1.1376e+00
1196  +1.070057146e-04 +1.070057146e-04 +7.948010701e+01 +7.948010701e+01 +2.5105e-01 -1.1466e+00
1501  +9.045741345e-05 +9.045741345e-05 +7.948009046e+01 +7.948009046e+01 +2.4331e-01 -1.1562e+00
1541  +5.961910446e-05 +5.961910446e-05 +7.948005962e+01 +7.948005962e+01 +2.4510e-01 -1.1574e+00
1637  +3.449297779e-06 +3.449297779e-06 +7.948000345e+01 +7.948000345e+01 +2.5095e-01 -1.1569e+00
1757  +1.910415222e-06 +1.910415222e-06 +7.948000191e+01 +7.948000191e+01 +2.5156e-01 -1.1562e+00
1847  +8.104174896e-07 +8.104174896e-07 +7.948000081e+01 +7.948000081e+01 +2.5213e-01 -1.1562e+00
2147  +4.012343311e-07 +4.012343311e-07 +7.948000040e+01 +7.948000040e+01 +2.5252e-01 -1.1574e+00
2336  +1.744265887e-07 +1.744265887e-07 +7.948000017e+01 +7.948000017e+01 +2.5301e-01 -1.1564e+00
2551  +4.383706198e-08 +4.383706198e-08 +7.948000004e+01 +7.948000004e+01 +2.5291e-01 -1.1570e+00
2911 |+1.571336838e-09 +1.571336838e-09 +7.948000000e+01 +7.948000000e+01 +2.5281e-01 -1.1568e+00
```

figure (3.1) Sample of the raw performance data pre-treatment of dimension 2 data.

It was determined that we would utilize 5 instances from each function within the domain {1...24}. Thanks to the numerous GECCO (Genetic and Evolutionary Computation Conference) BBOB competitions held over the years, the performance data of the best-performing state-of-the-art algorithms across the 480 BBOB [1] instances is available online in the BBOB data archive. This availability is a result of the systematic construction of an online algorithm selector archive, and the results were aggregated based on the 256 solvers that were submitted to the COCO-platform.

The BBOB datasets collected by the COCO platform can be accessed at [29]. Figure 3.2 illustrates the online archive for the BBOB datasets collected by the COCO platform.

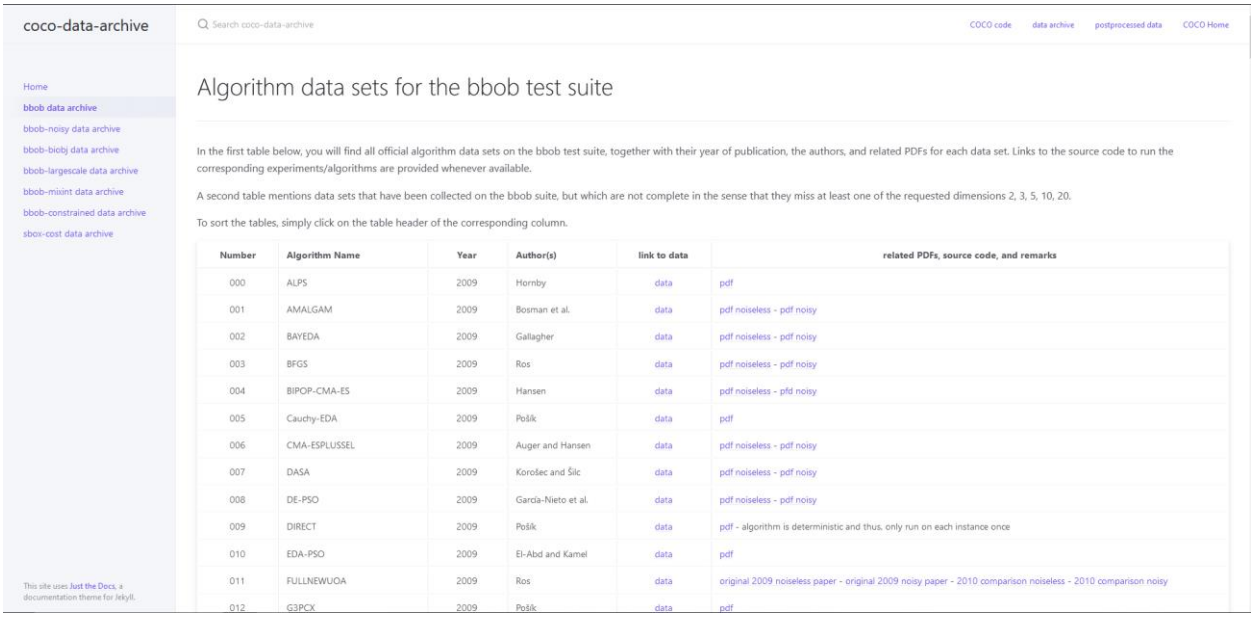


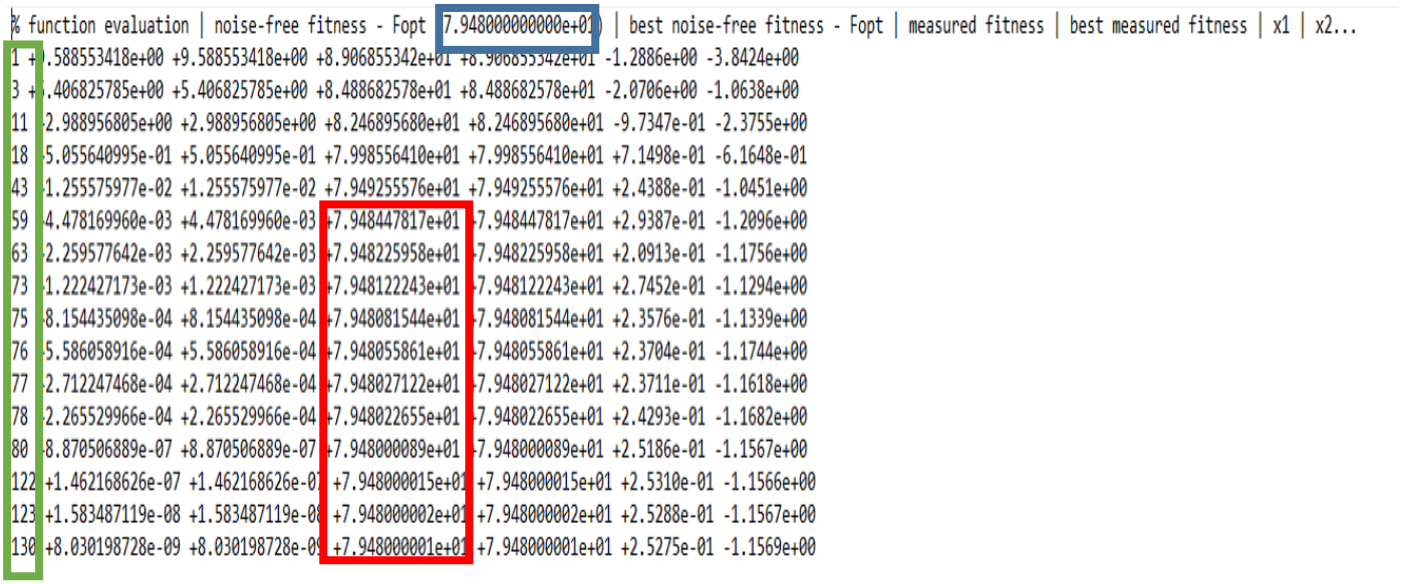
Figure (3.2): The BBOB datasets by the coco-data archive.

3.2.2 Demonstrative examples on calculating ERT and RELERT

ERT (Expected Runtime)

In this section, we provide illustrative examples of how the measures used to rank algorithms were calculated. we begin with an example of calculating the Expected Runtime (ERT).

For the ERT calculation, we identify the success cases by examining the measured fitness values (shown in red) that have achieved the Fopt (shown in blue) within a range of $\varepsilon = 0.01$. Figure (3.3) serves as a representative example of this process.



% function evaluation	noise-free fitness - Fopt (7.948000000000e+01)	best noise-free fitness - Fopt	measured fitness	best measured fitness	x1	x2...
1	+9.588553418e+00	+9.588553418e+00	+8.906855342e+01	+8.906855342e+01	-1.2886e+00	-3.8424e+00
3	+5.406825785e+00	+5.406825785e+00	+8.488682578e+01	+8.488682578e+01	-2.0706e+00	-1.0638e+00
11	+2.988956805e+00	+2.988956805e+00	+8.246895680e+01	+8.246895680e+01	-9.7347e-01	-2.3755e+00
18	+5.055640995e-01	+5.055640995e-01	+7.998556410e+01	+7.998556410e+01	+7.1498e-01	-6.1648e-01
43	+1.255575977e-02	+1.255575977e-02	+7.949255576e+01	+7.949255576e+01	+2.4388e-01	-1.0451e+00
59	+4.478169960e-03	+4.478169960e-03	+7.948447817e+01	+7.948447817e+01	+2.9387e-01	-1.2096e+00
63	+2.259577642e-03	+2.259577642e-03	+7.948225958e+01	+7.948225958e+01	+2.0913e-01	-1.1756e+00
73	+1.222427173e-03	+1.222427173e-03	+7.948122243e+01	+7.948122243e+01	+2.7452e-01	-1.1294e+00
75	+8.154435098e-04	+8.154435098e-04	+7.948081544e+01	+7.948081544e+01	+2.3576e-01	-1.1339e+00
76	+5.586058916e-04	+5.586058916e-04	+7.948055861e+01	+7.948055861e+01	+2.3704e-01	-1.1744e+00
77	+2.712247468e-04	+2.712247468e-04	+7.948027122e+01	+7.948027122e+01	+2.3711e-01	-1.1618e+00
78	+2.265529966e-04	+2.265529966e-04	+7.948022655e+01	+7.948022655e+01	+2.4293e-01	-1.1682e+00
80	+8.870506889e-07	+8.870506889e-07	+7.948000089e+01	+7.948000089e+01	+2.5186e-01	-1.1567e+00
122	+1.462168626e-07	+1.462168626e-07	+7.948000015e+01	+7.948000015e+01	+2.5310e-01	-1.1566e+00
123	+1.583487119e-08	+1.583487119e-08	+7.948000002e+01	+7.948000002e+01	+2.5288e-01	-1.1567e+00
130	+8.030198728e-09	+8.030198728e-09	+7.948000001e+01	+7.948000001e+01	+2.5275e-01	-1.1569e+00

Figure (3.3): ERT calculation example

$$\text{ERT(i)} = \frac{\sum iFE(\varepsilon)}{\sum iSuccess(\varepsilon)} = \frac{1032}{11} = \mathbf{93.8181818}$$

The ERT value is calculated for each individual instance across all algorithms. These calculated ERT values are then utilized in the calculation of the Relative Expected Runtime (RELERT).

RELERT (Relative Expected Runtime)

To calculate the RELERT, we require the minimum ERT value for all instances within a specific dimension and the minimum ERT value for the corresponding function. These minimum ERT

values are used as reference points for comparison. By comparing the RELERT values of different algorithms, we can determine the ranking of the algorithms based on their performance.

Algorithm	FID	IID	Dimensio	ERT	min_ERT	RELERT
BIPOPsaA(	1.0	1	2	93.81818181818	93.81818181818181	1.0
BIPOPsaA(	1.0	1	3	135.2	124.08333333333333	1.0895903290799194
BIPOPsaA(	1.0	1	5	226.44	222.35	1.0183944232066562
BIPOPsaA(	1.0	1	10	391.333333333333	391.3333333333333	1.0
BIPOPsaA(	1.0	2	2	174.142857142857	93.81818181818181	1.8561738648947952
BIPOPsaA(	1.0	2	3	170.571428571428	124.08333333333333	1.3746522114554351
BIPOPsaA(	1.0	2	5	233.047619047619	222.35	1.0481116215319048
BIPOPsaA(	1.0	2	10	402.72	391.3333333333333	1.0290971039182284
BIPOPsaA(	1.0	3	2	126.2	93.81818181818181	1.34515503875969
BIPOPsaA(	1.0	3	3	138.0	124.08333333333333	1.112155809267965
BIPOPsaA(	1.0	3	5	256.652173913043	222.35	1.1542710767395705
BIPOPsaA(	1.0	3	10	392.958333333333	391.3333333333333	1.0041524701873936
BIPOPsaA(	1.0	4	2	116.0	93.81818181818181	1.2364341085271318
BIPOPsaA(	1.0	4	3	139.75	124.08333333333333	1.1262592343854936
BIPOPsaA(	1.0	4	5	227.96	222.35	1.0252304924668316
BIPOPsaA(	1.0	4	10	458.863636363636	391.3333333333333	1.1725646585101441
BIPOPsaA(	1.0	5	2	103.625	93.81818181818181	1.10453003875969
BIPOPsaA(	1.0	5	3	124.083333333333	124.08333333333333	1.0

Figure (3.4): RELERT calculation example

$$\text{RELERT(i)} = \frac{ERT(i)}{Min(ERT(i))} = \text{Result}$$

3.2.3 Algorithm portfolio

We have selected 10 algorithms to train our model, which are variations of the two main optimization algorithms we relied on: Covariance Matrix Adaptation (CMA) and Differential Evolution (DE). The list of selected algorithms is as follows:

The CMA-based algorithms that we have selected are as follows:

-CMAES : short for covariance matrix adaptation-evolution strategy is an evolutionary algorithm for continuous stochastic non-linear and non-convex optimization problems with non-linear and non-convex search landscapes, population-based optimization algorithm that means it works with population of candidate solutions ,the candidates solution selected are evaluated and the best-

performing ones are used to update the mean and covariance matrix of the sampling distribution for the next gen of solutions[6]

-CMA: covariance matrix adaptation is a strategy used in the evolutionary algorithms domain for complex functions it is famous for its implementation in the CMA-ES [5]

-BIPOPsaACM : short for Biased Initialization Population-based Optimization for Self-adaptive Active Covariance Matrix biased initialization mean that the process with the initial population is biased toward certain region of the search space, pop-based indicate that the algorithm works with a groupe of candidate solutions rather than a single solution just like genetic algorithms and particle swarm optimizations, self-adaptive mean that the algorithm can adapt his parameters based on the process of the optimization

An advanced optimization technique that combines biased initialization and self-adaptive strategies within a population-based framework, and it actively modifies the covariance matrix to enhance the search efficiency. This type of algorithm can be particularly effective for complex optimization problems where the search space is high-dimensional, noisy, or has many local optimal [11].

-CMAa: Covariance Matrix Adaptation with active covariance matrix updates Active updates refer to a specific strategy within CMA-ES that enhances the adaptation process. It involves adjusting the covariance matrix using not only successful mutations (which lead to better solutions but also considering unsuccessful mutations (which do not improve the solution) [13].

-IPOP-ACTCMA-ES: Increasing Population Size with Active Covariance Matrix Adaptation Evolution Strategy [12]

-CMA CSA: Covariance Matrix Adaptation with Covariance Matrix Self-Adaptation. This is an advanced variant of the Covariance Matrix Adaptation Evolution Strategy. It self-adaptation mechanisms specifically for the covariance matrix. This approach aims to enhance the flexibility and efficiency of the optimization process by allowing the algorithm to dynamically adjust the covariance matrix based on the evolving search dynamics [14].

-CMA-ESPLUSSEL: Covariance Matrix Adaptation Evolution Strategy with Plus Selection This is a specific variant of the standard Covariance Matrix Adaptation Evolution Strategy (CMA-ES) that incorporates a plus selection mechanism

The DE (Differential Evolution) algorithms that we have selected are as follows:

-JADEb: algorithm (Joint Approximate Diagonalization of Eigen-matrices for blind source separation is an adaptive variant of the DE

-DE-rand: is arguably the most popular DE variant [16][17]
-DE uniform_fialho_noiseless: DE uniform is a variant that employs a uniform crossover mechanism. The DE uniform variant developed by Fialho and colleagues enhances DE by incorporating adaptive strategies to improve performance [22], particularly in noiseless environments where precision is crucial.

3.2.2 Features

The second part of the dataset comprises the features, which play a crucial role in our work. While the landscape of features can be computed for any optimization problem, our focus is specifically on single-objective continuous optimization problems.

Our work revolves around the ELA (Exploratory Landscape Analysis) technique, which involves characterizing a problem landscape using numerical feature values. This characterization aids in the selection of the most suitable algorithm for solving the problem.

To obtain these features, we employ systematic sampling of the decision space using various methods. In our case, we utilize Latin hypercube sampling (LHS) [4]. LHS generates a near-random sample of parameter values for multidimensional distributions, allowing us to effectively capture the characteristics of the landscape.

These features enable us to identify specific landscape characteristics, such as convexity, roughness, or funnel structure. These characteristics have a significant impact on the performance of optimization algorithms. By considering these features, we can make informed decisions and choose algorithms that are well-suited to the problem's landscape. To obtain these problem-specific features, we utilized the online tool called Flacco GUI, as shown in Figure (3.3). The Flacco GUI provides a graphical user interface that simplifies the extraction and

analysis of ELA features. It offers a variety of tools for computing different landscape descriptors. The Flacco GUI can be accessed at [28], where users can utilize its features to compute and analyze landscape descriptors specific to their optimization problems.

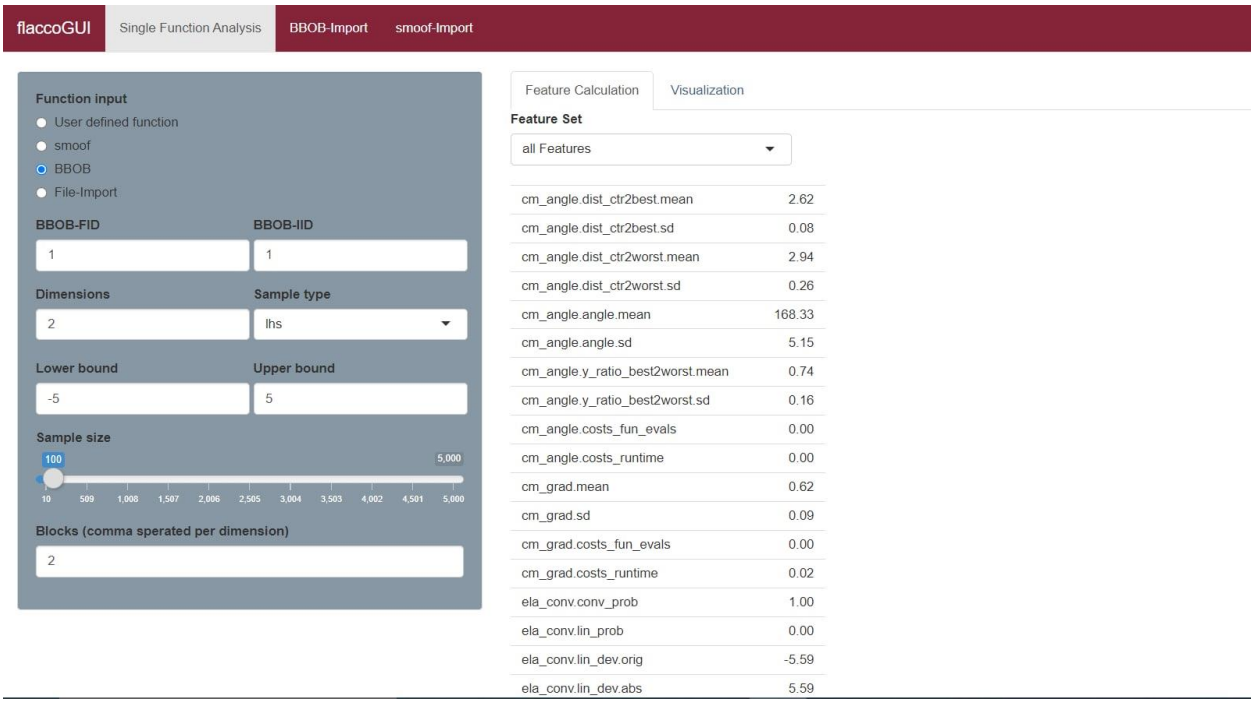


figure (3.5) The graphical user interface Flacco for BBOB features extraction

Feature extraction was performed using the Flacco GUI tool, specifically with Benchmarking Black-Box Optimization (BBOB) instances. The steps involved in this process are outlined below:

- BBOB-FID Selection: We chose the BBOB-FID values ranging from 1 to 24, representing 24 different functions, for each algorithm.
- BBOB-IID Selection: We selected the BBOB-IID values from 1 to 5, covering a set of 5 instances that span the dimension domain. The dimension domain, denoted as D, includes values 2, 3, 5, and 10.
- Sample Size Calculation: We determined the sample size using the following formula: Sample configuration for each dimension equals to 50 times the dimension. The bounds for the samples were set from -5 to 5. For instance, if the dimension is 2, we used 100

samples, while for a dimension of 10, we used 500 samples. This configuration ensures an adequate number of samples to capture the landscape characteristics effectively.

We opted to use the Flacco GUI tool specifically because it was configured to extract features from the BBOB instances. The tool simplifies the feature extraction process by providing parameter selection options and performing the necessary calculations without requiring data to be loaded into the tool separately. This streamlined approach facilitated the extraction of features for our analysis.

3.3.3 The types of extracted features:

We extracted various types of features using the Flacco GUI, including numerous descriptors that capture different aspects of the optimization landscape. Some of the classic features that were extracted include:

Convexity: a measure of the convexity of the landscape, which help the algorithms to make the process of finding the global optimal an easier task.[35]

Curvature: which is an analysis of the local curvature of the landscape, it is crucial for the gradient information based algorithms.

Level set: Distribution of feature levels, it provides information about landscape topology.

Local Search: performance of the local search methods on the landscape, indicating the presence of local minima.[35]

Metamodel: fitting simple models to the data, to figure out the complexity of the landscape.

Distribution of y: Statistical analysis of function values, to evaluate the dispersion and shape of the data.[23]

We can say that the general workplan for our implementation can be presented in **figure (3.6)** below:

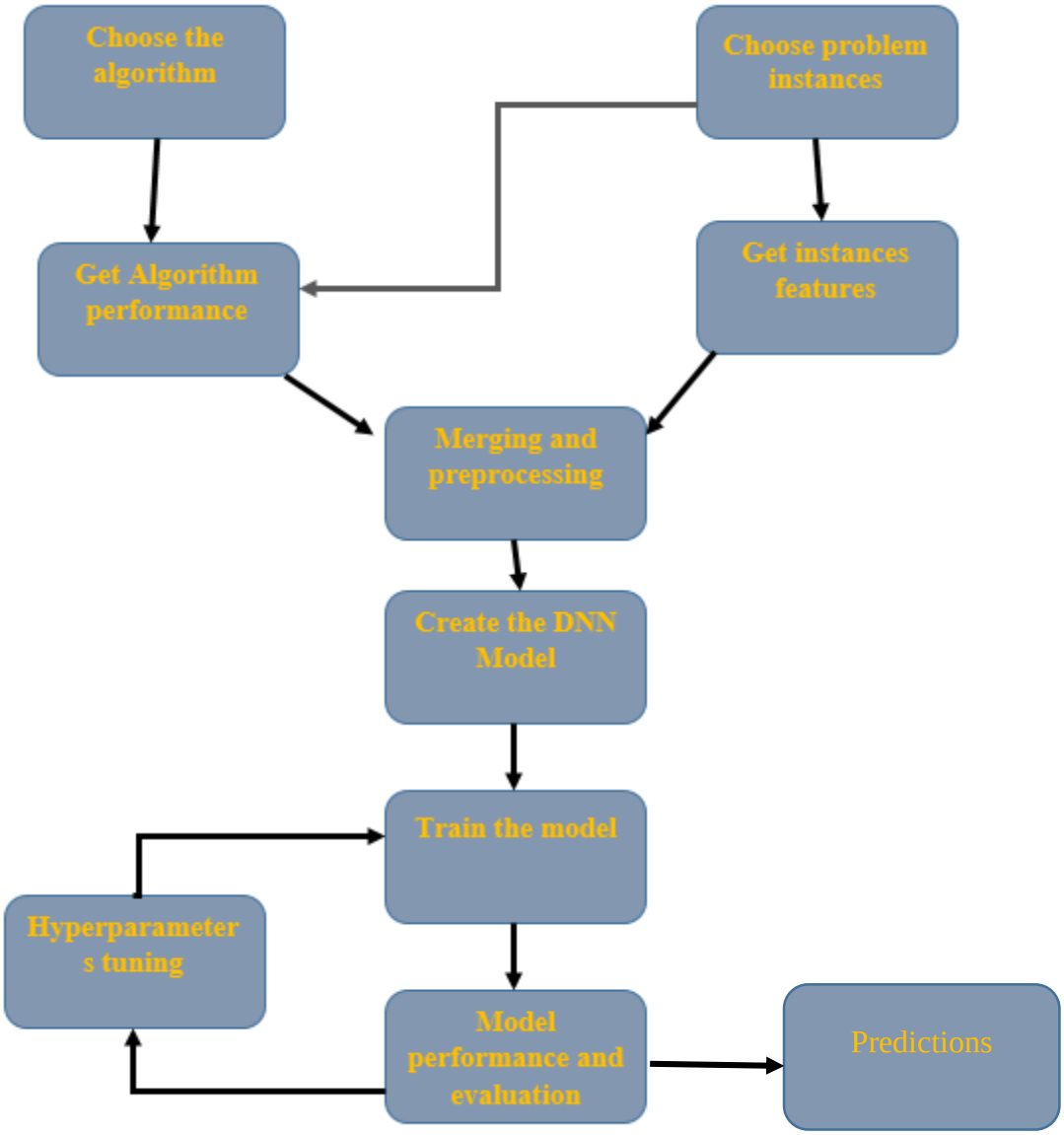


Figure (3.6): Scheme model development plan

3.2 Preprocessing

The preprocessing phase received special attention as it was meticulously designed to ensure that the data was properly cleaned, consistent, and suitable for training the deep learning model. The process involved loading the ELA features and transforming them, handling missing values and NaNs, as well as addressing the class imbalance problem. For balancing the class distribution imbalance, the SMOTE (Synthetic Minority Over-sampling Technique) method was utilized, which creates synthetic samples for the minority classes, the steps are detailed below:

- **Data Loading and Transformation:**

The ELA data was loaded from its corresponding CSV file. Next, the data was pivoted to transform the 'Feature name' into columns, while using 'FID', 'IID', and 'Dimension' as indices. This reshaping process was performed to prepare the data in a format suitable for deep learning models. The transformed data was then saved into a new CSV file for future use.

1. **Handling Missing Values:**

Columns such as 'measured_fitness', 'Fotp', and 'N_function_evaluation' were converted to numeric types, ensuring that any non-numeric data and NaN values were appropriately handled.

2. **Oversampling with SMOTE**

To address the issue of class imbalance, we employed the SMOTE (Synthetic Minority Over-sampling Technique) method. This technique is used to generate synthetic samples for the minority classes, which helps in balancing the class distribution and enhances the model's ability to learn from all classes equally

Below is **figure 3.7**, which shows the pseudo code for the preprocessing phase.

Algorithm 1 Preprocessing

```
Function PREPROCESS(Performance, Features)
data1 ← LOADDATA(Performance)
data2 ← LOADDATA(Feature)
data1 ← CALCULATERELERT(data1)
data ← MERGEDATA(data1, data2)
data ← FILTERDATA(datas)
data ← CLEANDATA(data)
X ← data.DROP_COLUMNS(['Target'])
y ← data['Target']
X ← NORMALIZEDATA(X)
y ← ENCODETARGET(y)
X_resampled, y_resampled ← APPLYSMOTE(X, y)
return Normlized data
End Function
```

Figure (3.7): Pseudo code for the preprocessing phase

The data was split according to Table 3.2 after the preprocessing phase was completed:

Data split table after smote

Dataset split	Percentages	Number of rows
Train	60%	712
Validation	20%	238
Test	20%	238
All	100%	1188

Table (3.2): Data split table

3.3 Network architecture:

We developed two distinct models for the purpose of AS, each with its own network architecture. The descriptions of these network architectures are outlined below:

3.4.1 Model architecture (Model 1)

1. **Input layer :** The input layer receives the input data with a shape corresponding to the number of features in X_train (151).
2. **First Dense Layer:**

- **Dense Layer:** 32 neurons, ReLU activation function.
- **Regularization:** L1 regularization with a regularization coefficient of 0.001 to prevent overfitting by adding a penalty for large weights.

3. **Batch Normalization:**

   It is a normalization step for the outputs of the first dense layer. Such a normalization step ensures that outputs have an average of close to zero and an output standard deviation of close to one, hence improving the training speed and stability. With the normalized outputs, it will be more feasible to have better learning and optimization of subsequent layers during training.

4. **Dropout Layer:**
   - Dropout rate of 0.3, meaning 30% of the neurons are randomly dropped out during training to prevent overfitting by introducing noise and forcing the network to learn more robust features.
5. **Output Layer :**
   - **Dense Layer:** Number of neurons equal to the number of unique classes in y_train(10).
   - **Activation Function:** Softmax activation function to output a probability distribution over the classes.

3.4.2 Model Compilation (Model 1)

- **Optimizer:** The model utilizes the Adam optimizer with a learning rate of 0.0001. The Adam optimizer is chosen for its efficiency and adaptive gradient-based optimization capabilities.
- **Loss Function:** The model employs the Sparse Categorical Crossentropy loss function, which is specifically designed to handle multi-class classification problems where the labels are represented as integers.
- **Metrics:** The chosen metric for monitoring the training and validation performance of the model is accuracy. Accuracy is a commonly used metric in classification tasks, which measures the proportion of correctly classified samples over the total number of samples.

- **Callbacks**

1. Early Stopping:

- Through the training process, the validation loss will be monitored to prevent overfitting. In the case when validation loss does not improve, the training shall be stopped through early stopping. This procedure prevents the model from too tight a fit to the training dataset and helps in generalizing better to unseen data.
- It has a patience of 15 epochs, which means that the training will stop if there is no improvement in validation loss for 15 epochs in a row. This early stopping mechanism avoids overfitting and saves time used for training by stopping it when there is no significant improvement in the validation loss over some number of epochs.
- The parameter restore_best_weights=True ensures that the model weights from the epoch with the best validation loss are used.

2. ReduceLROnPlateau:

- Monitors validation loss (val_loss) and reduces the learning rate by a factor of 0.1 if the validation loss does not improve for 5 consecutive epochs.
- Minimum learning rate set to 0.000001 to ensure the learning rate does not reduce to an ineffective level.

- **Class Weights**

  Class weights are calculated to handle class imbalance by assigning a weight to each class inversely proportional to its frequency in the training data. This ensures the model does not become biased towards the majority class.

3.4.3 Model training (Model 1)

The model is trained for a maximum of 290 epochs, reduced from the original 600 epochs. Each epoch utilizes a batch size of 32. The training process incorporates class weights and early stopping to mitigate overfitting.

During training, the validation data is used to assess the model's performance on unseen data. The validation loss (val_loss) is monitored, and if there is no improvement after five consecutive

epochs, the learning rate is reduced by a factor of 0.1. To ensure the learning rate does not drop below an ineffective level, a minimum learning rate of 0.000001 is set.

This comprehensive approach ensures that the model is well-regularized and capable of generalizing to new data, making it suitable for the classification task at hand. By using class weights, early stopping, and monitoring the validation loss, the model is trained in a controlled manner to achieve optimal performance while avoiding overfitting.

Training parameters:

- **Epochs:** 600.
- **Batch Size:** 32.
- **Validation Data:** Provided during training.
- **Callbacks:** Early stopping and learning rate reduction.

3.4.4 Model architecture (Model 2)

1. **Input layer :** The input layer receives the input data with a shape corresponding to the number of features in X_train.

   First Dense Layer:

   - **Dense Layer:** 128 neurons, ReLU activation function.
   - **Regularization:** L1 regularization with a regularization coefficient of 0.001 to

   Prevent overfitting by adding a penalty for large weights.

   Batch Normalization:

   - Normalizes the outputs of the first dense layer to improve training speed and stability by maintaining the mean output close to zero and the output standard deviation close to one.

   Dropout Layer:

   - Dropout rate of 0.3, meaning 30% of the neurons are randomly dropped out during training to prevent overfitting by introducing noise and forcing the network to learn more robust features.

   Hidden Layer:

We added two additional Dense Layers each one with 128 neurons with ReLU activation function, L1 Regularization of 0.001 followed by Batch Normalization and a Dropout rate of 30%.

Output Layer:

- **Dense Layer:** Number of neurons equal to the number of unique classes in y_train.
- **Activation Function:** Softmax activation function to output a probability distribution over the classes.

3.4.5 Model Compilation (Model 2)

- **Optimizer:** Adam optimizer with a learning rate of 0.001 for efficient and adaptive gradient-based optimization.
- **Loss Function:** Sparse Categorical Crossentropy to handle multi-class classification problems where labels are provided as integers.
- **Metrics:** Accuracy to monitor the training and validation performance.

Callbacks

1. Early Stopping:

- Monitors validation loss (val_loss) to prevent overfitting by stopping training when the validation loss stops improving.
- Patience set to 15 epochs, meaning training will stop if the validation loss does not improve for 15 consecutive epochs.
- restore_best_weights = True ensures the model weights from the epoch with the best validation loss are used.

2. ReduceLROnPlateau:

- Monitors validation loss (val_loss) and reduces the learning rate by a factor of 0.1 if the validation loss does not improve for 5 consecutive epochs.
- Minimum learning rate set to 0.000000001 to ensure the learning rate does not reduce to an ineffective level.

Class Weights

- Class weights are calculated to handle class imbalance by assigning a weight to each class inversely proportional to its frequency in the training data. This ensures the model does not become biased towards the majority class.

3.4.6 Model training (Model 2)

The second model is trained for a maximum of 178 epochs, reduced from the original 600 epochs. Each epoch employs a batch size of 32. The training process incorporates class weights and early stopping to prevent overfitting. Additionally, the validation data is used during the training to see how good it is on unseen data. Moreover, the validation loss (val_loss) is monitored and if there is no improvement after five consecutive epochs, learning rate gets reduced by a factor of 0.1. To ensure that the learning rate does not drop below an ineffective level, a minimum learning rate of 0.000000001 is set. This prevents the learning rate from becoming too small, which could hinder the model's ability to converge or make meaningful updates to its parameters.. This comprehensive approach can guarantee that the model is well-regularized, capable of generalizing to new data, making it suitable for the classification task at hand.

Training parameters:

- **Epochs:** 600.
- **Batch Size:** 32.
- **Validation Data:** Provided during training.
- **Callbacks:** Early stopping and learning rate reduction.

Conclusion

In this chapter, we focused on the implementation of the suggested method. We began by discussing the data collection phase and then proceeded to perform the necessary preprocessing on the collected data to make it suitable for training the models. Additionally, we provided an

explanation of the network architecture we utilized for our model, along with the process of model compilation and training to optimize the hyperparameters.

Using a deep learning model for Algorithm Selection shows immense potential as a significant advancement in the field of continuous optimization. By exploiting the power of deep learning, our model can effectively navigate the complex landscape of algorithm selection, providing robust and accurate recommendations.

Chapter 4

Experiments and results

Introduction

In this chapter, we present and discuss the results obtained by our DL-based models through a series of experiments. The primary objective of these experiments is to assess the performance, robustness, and generalizability of the models in the task of selecting the most suitable algorithms across a diverse set of continuous problem sets.

4.1 Model evaluation and comparison

The results observed for the performance metrics (accuracy, loss) after training the models will be presented in Table 4.1 below. The table displays the different results obtained for both the models used in the training phase.

Model	Epochs	Train loss	Train acc	Val loss	Val acc
Model 1	290	1.2525	0.6825	1.2781	0.7381
Model 2	178	1.3716	0.9014	1.5516	0.8912

Table (4.1): The results obtained on training the two ASM

- **Training vs Validation Accuracy(M1):** The model achieved a higher accuracy on the validation set (73.81%) compared to the training set (68.25%). This suggests that the model generalizes well to unseen data.
- **Training vs. Validation Loss :**
  The training and validation losses are very close, indicating that the model is not overfitting.
- **Training vs Validation Accuracy(M2):** The model achieved a validation accuracy of 89.12% and a training accuracy of 90.14%, with the training accuracy slightly higher than the validation accuracy. This indicates that the model is performing well and demonstrating adequate generalization to the validation set. The fact that the training accuracy is slightly higher suggests that the model has learned the underlying patterns in the training data effectively. Overall, these results indicate that the model is likely performing well and has the potential to make accurate predictions on unseen data.

- **Training vs. Validation Loss (M2):** There is a slight difference between the training loss and validation loss, which can be observed. However, as the training progresses, this difference decreases over epochs. The loss curves converge and stabilize, with the validation loss slightly higher than the training loss. This pattern is commonly observed and indicates a good fit without severe overfitting.

Analysis :

- **Epochs and convergence :**
  Model 1 required 290 epochs to converge, whereas Model 2 achieved convergence faster, taking only 178 epochs. The convergence speed of Model 2 indicates that it was able to reach a satisfactory level of performance in a shorter amount of time compared to Model 1.
- **Training Performance:**
  Model 1 has a lower training loss (1.2525) compared to Model 2 (1.3716), indicating that Model 1 fits the training data better. Training accuracy for Model 1 is 0.6825, whereas Model 2 reach a significantly higher training accuracy of 0.9014.
- **Validation Performance :**
  Model 1 has a lower validation loss (1.2781) compared to Model 2 (1.5516), suggesting that Model 1 generalizes better to the validation data**.**
  Validation accuracy for Model 1 is 0.7381, which is lower than Model 2's validation accuracy of 0.8912.

4.2 Performance plots

In this section, we will present the evolution of validation and accuracy values per epoch. We will visually depict the performance of both models on the dataset using plots in Figures 4.1 and 4.2. These figures highlight the differences in training and validation accuracy and loss over

epochs.

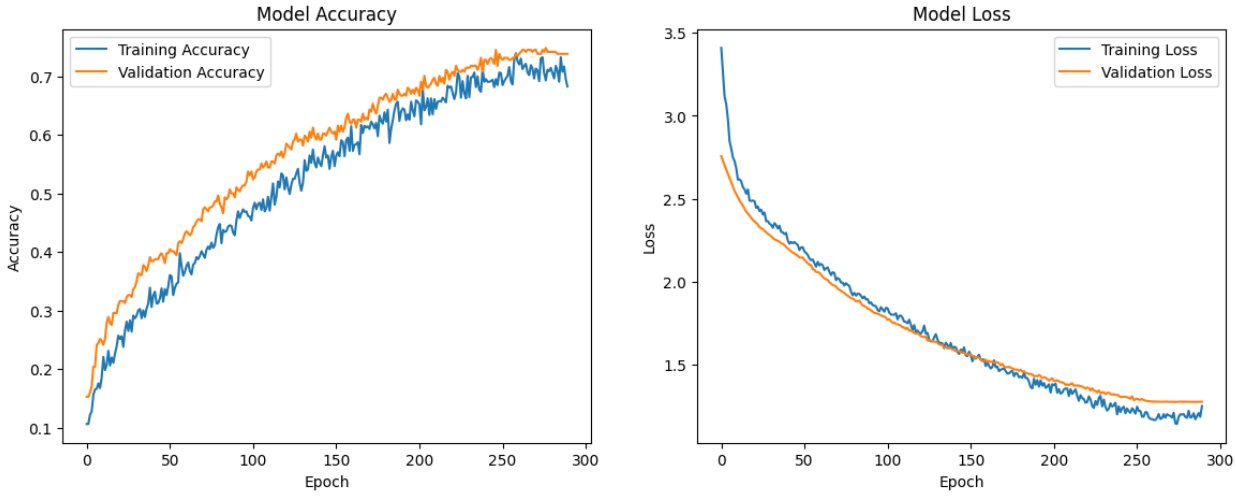


Figure (4.1): Accuracy and loss of the ASM 1 on the Dataset

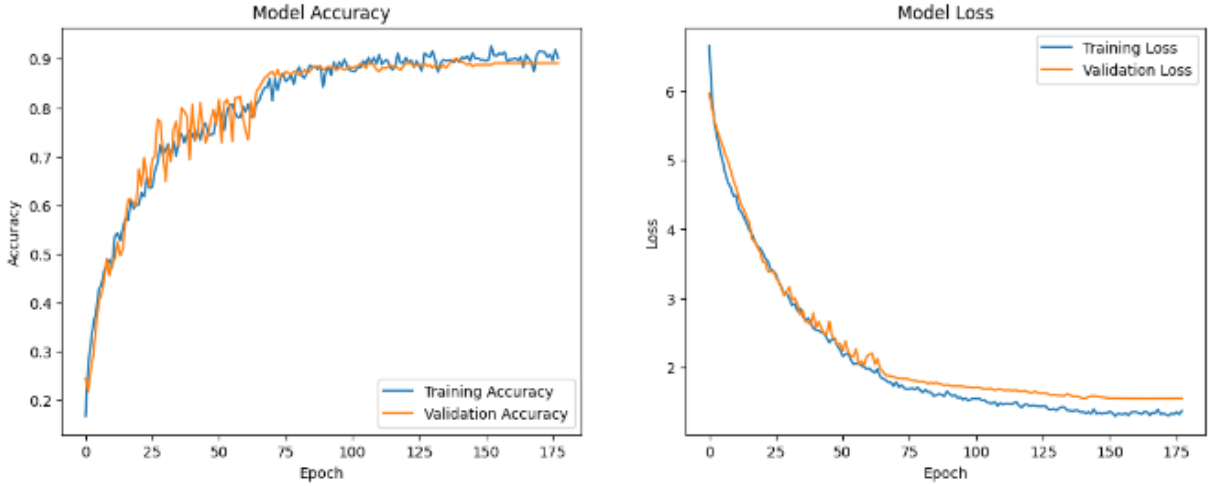


Figure (4.2): Accuracy and loss of the ASM 2 on the Dataset

The provided plots show the training and validation accuracy as well as the training and validation loss for both models

Models Accuracy and Loss Analysis

1. **Model Accuracy :**

Model 1 :

- The training and validation accuracy both show an increasing trend reaching 0.68

- Validation accuracy surpasses training accuracy at 0.73, which suggests good generalization.

Model 2 :

- The training accuracy increase at a high rate and stabilize around 0.90
- Validation accuracy does increase as well and become stable at 0.89

2. **Models Losses:**

Model 1 :

- Training loss decreases steadily, starting high and reaching around 1.25.
- Validation loss also decreases steadily, following a similar trend and stabilizing around 1.28.

Model 2 :

- Training loss decreases quickly, starting high and reaching around 1.37.
- Validation loss follows a similar trend but is slightly higher than the training loss, stabilizing around 1.55.

Conclusion:

Model 1 shows a consistent improvement in both training and validation accuracy and a steady decrease in loss, indicating good generalization and no significant overfitting.

Model 2 demonstrate a rapid improvement in accuracy and a significant decrease in loss initially, the notable point lay in higher validation loss compared to the training loss, indicating potential presence of overfitting.

4.3 Confusion matrix review

Below we present the confusion matrixes for our models where we illustrate the evaluation on the VBS against the SBS in other words the predicted labels against the true ones

The classes were encoded following the **Table (4.2)**

Class Name	Encoded Value
BIPOPsaACM	0
CMA	1
CMA-ESPLUSSEL	2
CMAES	3
CMAa	4
CMAcsa	5
DE-rand	6
DEuniform_fialho_noiseless	7
IPOP-ACTCMA-ES	8
JADEb	9

Table(4.2) : Table for the encoded classes values

Table (4.3) and **Table (4.4)** represent the confusion matrix for both models.

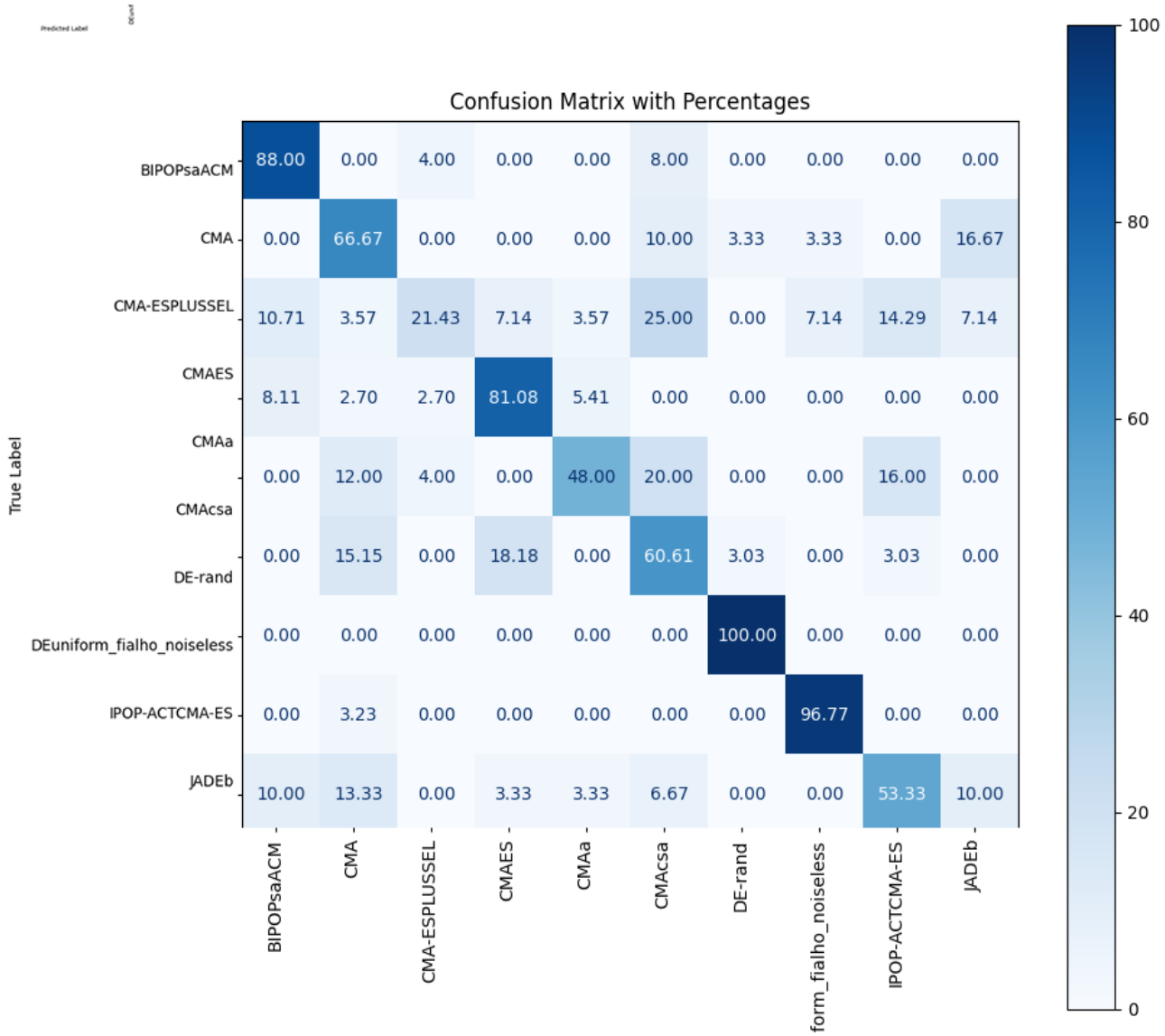


Table (4.3): Represents the confusion matrix for training of the ASM for the first model.

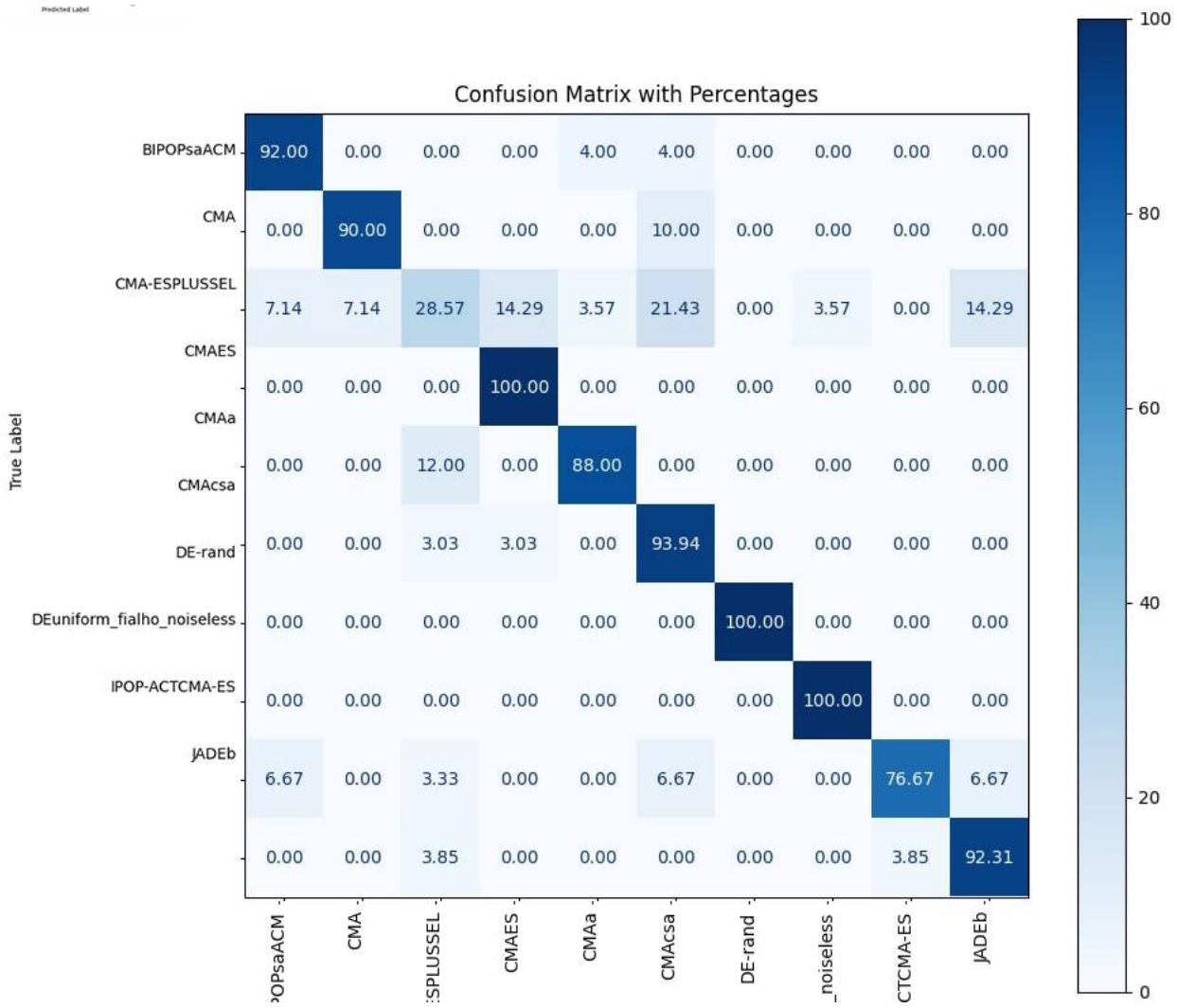


Table (4.4): Represents the confusion matrix for training of the ASM for the second model.

4.4 Interpretation of Confusion Matrix

The provided confusion matrix presents a detailed view of the model's performance. The diagonal values in the matrix represent the percentage of correct predictions, indicating how accurately the model classified instances of each class. On the other hand, the off-diagonal values in the matrix represent the percentage of misclassifications. These values indicate how often instances of one class are predicted as another, representing false positives or false negatives.

Classwise performance :

1. **Class 0 :**

   Model 1

   - 88% correctly classified.
   - 4% misclassified as class 2 and 8% as class 8.

   Model 2

   - 92% correctly classified.
   - 4% misclassed as class 4 with another 4% as class 5.

2. **Class 1 :**

   Model 1

   - 66.67% correctly classified.
   - Misclassifications are spread across several classes, with the highest (16.67%) being misclassified as class 9.

   Model 2

   - 90% correctly classified.
   - The rest were misclassed across different classes the largest was class 5

3. **Class 2 :**

   Model 1

   - 21.43% correctly classified.
   - Significant confusion with classes 0 (10.71%), 4 (25%), and 9 (14.29%).

   Model 2

   - 28.57% correctly classified.
   - 8% misclassed as class 5

4. **Class 3 :**

   Model 1 :

   - 81.08% correctly classified.
   - Some confusion with classes 0 (8.11%) and 1 (5.41%).

Model 2 :

- Had a 100% accurate classification
- There were no misclassifications.

5. **Class 4 :**

Model 1 :

- 48% correctly classified.
- 20% misclassified as class 5 and 16% as class 9.

Model 2 :

- 88% classes were correctly classified
- 12% misclassified as class 2

6. **Class 5 :**

Model 1 :

- 60.61% correctly classified.
- Some misclassifications with classes 1 (15.15%) and 2 (18.18%).

Model 2 :

- 93.94% correctly classified.
- 3.3% mistakenly classified as either class 2 or class 3

7. **Class 6 :**

Model 1 :

- 100% correctly classified.
- No misclassifications.

Model 2 :

- 100% correctly classified.
- No misclassifications.

8. **Class 7 :**

Model 1 :

- 96.77% correctly classified.

- Minor confusion with class 1 (3.23%).

Model 2 :

- 100% correctly classified
- No misclassifications.

9. **Class 8 :**

Model 1 :

- 53.33% correctly classified.
- Misclassified mainly as class 0 (10%) and class 1 (13.33%).

Model 2 :

- 76.67% correctly classified.
- 6.67% were misclassified as classes 0,5,9.

10. **Class 9 :**

Model 1

- 76.92% correctly classified.
- Misclassified as class 0 (3.85%), class 2 (7.69%), class 4 (3.85%), and class 8 (3.85%).

Model 2

- 92.31% correctly classified.
- 3.85% were misclassified as classes 8 and 2.

4.5 Solvers review

D	BBOB Groupe	BIPOPsaACM	CMA	CMA-ESPLUSSEL	CMAES	CMAa	CMAcsa	DE-rand	DEuniform-fialho-noiseless	IPOP-ACTCMA-ES	JADEb
2	F1-F5	40035.4	40020.0	80006.8	***23.8***	40024.1	40021.1	147.9	80009.1	40023.4	40007.5
	F6-F9	2.5	4.2	***2.0***	4.0	3.7	4.0	9.1	5.1	3.0	2.9
	F10-F14	***1.8***	5.1	2.7	4.1	3.6	3.3	10.4	40005.0	4.2	5.5
	F15-F19	17.4	10.0	120016.3	23.9	7.5	9.5	15.3	***7.4***	10.6	80008.4
	F20-F24	53.4	160039.8	240007.1	69.2	160032.9	200036.1	***26.1***	200007.8	200066.2	120012.4
	All	8022.1	40015.8	88007.0	***25.0***	40014.4	48014.8	41.8	64006.9	48021.5	48007.4
3	F1-F5	160148.5	240009.5	160003.7	40021.3	240012.1	240009.6	***129.1***	240011.2	240006.9	40007.7
	F6-F9	2.8	3.4	***2.2***	3.3	2.7	2.7	17.6	8.4	3.1	100004.3
	F10-F14	4.4	4.2	***2.7***	4.3	3.5	3.2	20.2	40006.6	3.4	6.1
	F15-F19	6.4	6.6	320014.7	22.5	7.6	***4.4***	17.7	40009.3	13.1	12.1
	F20-F24	40031.1	200038.6	200023.9	***46.9***	200049.7	280022.9	200023.0	200008.3	200025.5	200035.6
	All	40038.6	88012.4	136009.5	***8019.6***	88015.1	104008.6	40041.5	104008.8	88010.4	68013.2
5	F1-F5	240049.2	280005.6	200001.5	320006.8	360004.6	360003.5	***318.2***	280017.6	440003.4	40009.8
	F6-F9	***2.8***	4.8	7.1	6.6	3.8	3.4	46.2	11.4	3.7	100007.8
	F10-F14	***1.2***	5.2	3.3	5.0	3.9	3.7	82.3	11.0	3.9	10.3
	F15-F19	25.6	13.4	520001.6	200040.1	13.3	***8.2***	240051.7	120054.0	10.2	320017.0
	F20-F24	80197.4	480176.0	440029.4	300014.8	480158.5	360122.1	360118.5	360027.3	480125.6	360092.7
	All	***64055.2***	152041.0	232008.6	164014.7	168036.8	144028.2	120123.4	152024.3	184029.3	164027.5
10	F1-F5	281159.6	400008.2	200002.7	400003.3	440002.9	440002.4	***80815.8***	440018.5	400003.5	120020.6
	F6-F9	***2.7***	5.0	150013.0	15.2	3.5	3.2	212.5	18.5	4.0	250009.9
	F10-F14	***1.2***	5.7	5.8	6.0	3.8	4.1	800039.2	17.1	3.8	31.9
	F15-F19	295.1	101.5	560001.9	441074.1	***67.0***	82.6	600169.8	600004.8	74.2	720005.9
	F20-F24	***200091.7***	600014.9	480022.3	550043.3	680016.7	640015.7	760088.7	760021.8	640015.3	560034.8
	All	***96310.1***	200027.1	278009.1	278228.4	224018.8	216021.6	448265.2	360016.1	208020.1	330020.7
ALL	**All**	***52106.5***	120024.1	183508.5	112571.9	130021.3	128018.3	152118.0	170014.0	132020.3	152517.2

Table 4.5: Summary of the relative expected runtime of the 10 portfolio solvers shown per dimension and BBOB function group. (The single best solver (SBS) for each of these combinations is printed in red and the overall SBS (**BIPOPsaACM**) is highlighted in blue. Values in red indicate that this solver was better than the rest of solvers in general.)

Comparing the ASM's with the SBS :

Dimension	BBOB Groupe	BIPOPsaACM	Model 1	Model 2
2	F1-F5	40035.4	40022.6	80009.6
	F6-F9	2.5	3.2	2.2
	F10-14	1.8	5.7	2.5
	F15-19	17.4	40013.8	40014.9
	F20-24	53.4	200029.4	200010.1
All		8022.1	56014.9	64007.9
3	F1-F5	160148.5	6.1	4.1
	F6-F9	2.8	2.6	1.9
	F10-14	4.4	2.4	2.4
	F15-19	6.4	40008.1	2.1
	F20-24	40031.1	80038.8	80014.7
All		40038.6	24011.6	16005.0
5	F1-F5	240049.2	5.1	40001.9
	F6-F9	2.8	2.1	1.7
	F10-14	1.2	2.7	2.8
	F15-19	25.6	40022.3	6.9
	F20-24	80197.4	400063.1	240041.2
All		64055.2	88019.1	56010.9
10	F1-F5	281159.6	120965.3	320024.1
	F6-F9	2.7	2.2	2.2
	F10-14	1.2	1.5	1.2
	F15-19	295.1	56.9	40054.8
	F20-24	200091.7	360040.6	400018.7
All		96310.1	96213.3	152020.2

Table 4.6: Comparison of RELERT table between SBS and the ASM's across dimensions and BBOB function groups

For most of the BBOB groups, the relative ERTs of our ASM's are below the ones of the SBS

And the overall average of model 1 stands superior to the other model and the SBS.

4.6 Discussion and results

Based on the results achieved, one could argue that the DL models demonstrated superior RELERT across most of the BBOB functions. The SBS (BIPOPsaACM) instances, which were selected as the SBS earlier showed greater RELERT only at the dim=2. However, we can notice that both models managed to surpass the SBS, First model at the high dimensional space (dim= 10) and model 2 at the low dimension function (Dim= {2,5}).

We as well can deduce from the **Table 4.5** the distribution of results across lower dimensions of different algorithms claims the best performance in terms of relative expected runtime while in the higher dimensions that performance was dominated by the selected SBS, even though the ASM's trained managed to achieve better RELERT on average over the SBS at numerous instances in different function groups both at high and low dimensionality that is proved at the **Table 4.6**. Which demonstrate that the classification based DNN models managed to only partially capture the best results at specific dimensions.

We believe that jack of all trades model for algorithm selection can't be achieved based on our work, since capturing the difficulty problems on a single measure basis isn't a realistic approach

It seems to be more logical to set the target for single dimensional feature based selection.

We also think that our dataset creation method wasn't optimal at some point since the results shows high noise specially on low dimension groups ,we also believe that the performance would be boosted more if more problem instances were available for the training. While the performance of the ASM's can be clearly distinguished from the baseline of the SBS we think a single dimensional feature targeting approach for problem instances could be a better potential solution to the ASP.

4.7 Application's implementation

The figures below demonstrate the developed web app for our work:

1- The home interface: is the first interface where the user can choose to use our trained models or train the models on new data set.

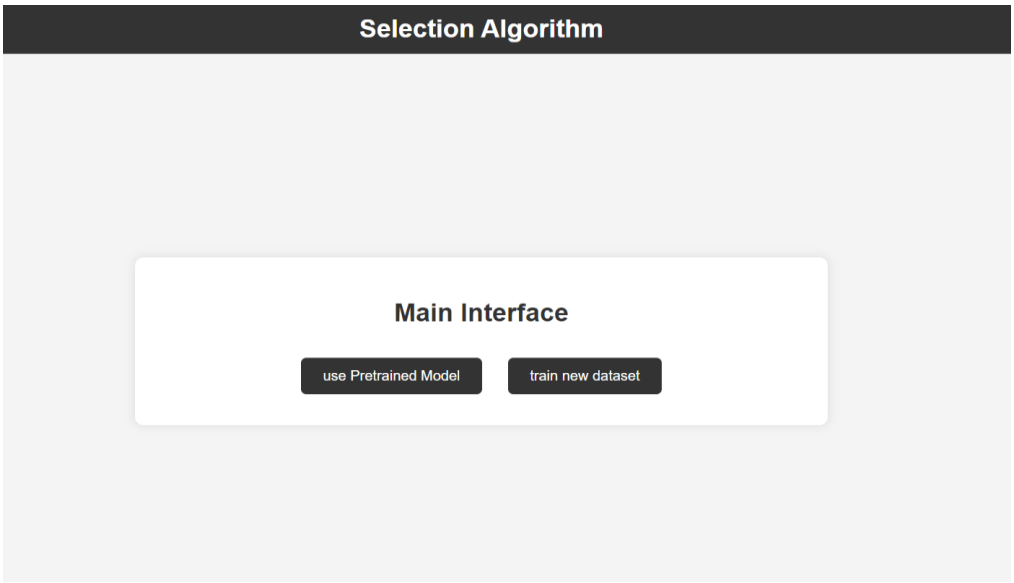


Figure 4.3: The home interface for the web app

2- If the user chose to use the pretrained model he can choose see the results our models managed to achieve in detail where we present the confusion matrix and the plot depending on the user desire.

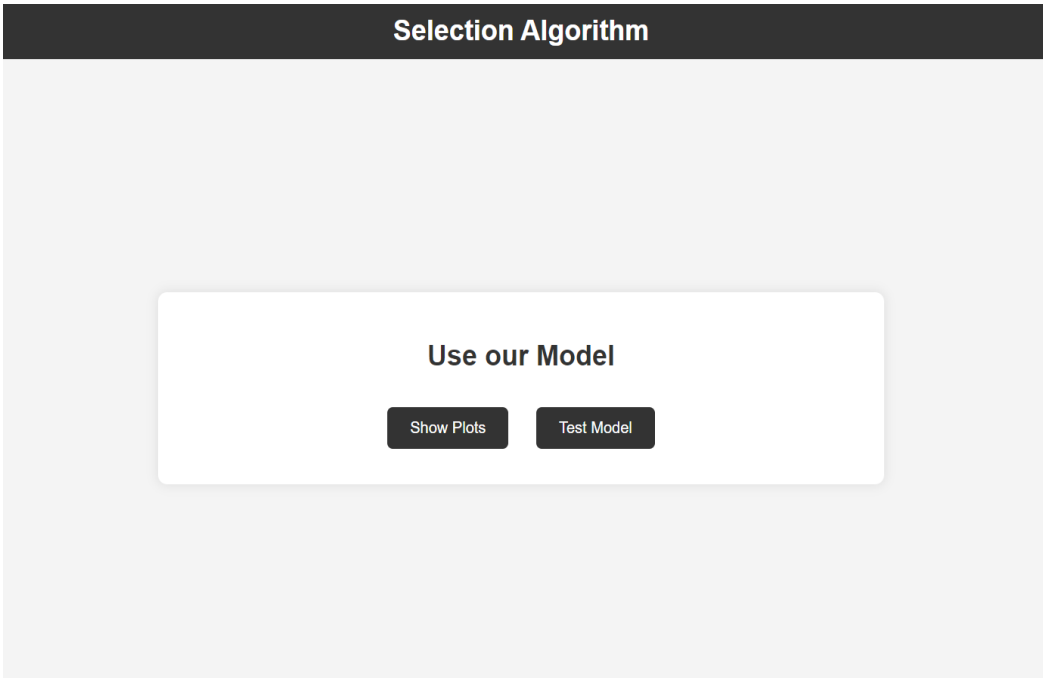


Figure 4.4: The pretrained model interface

3- The show plots button present the accuracy and loss plots were the progress of accuracy and loss for both validation and training is presented

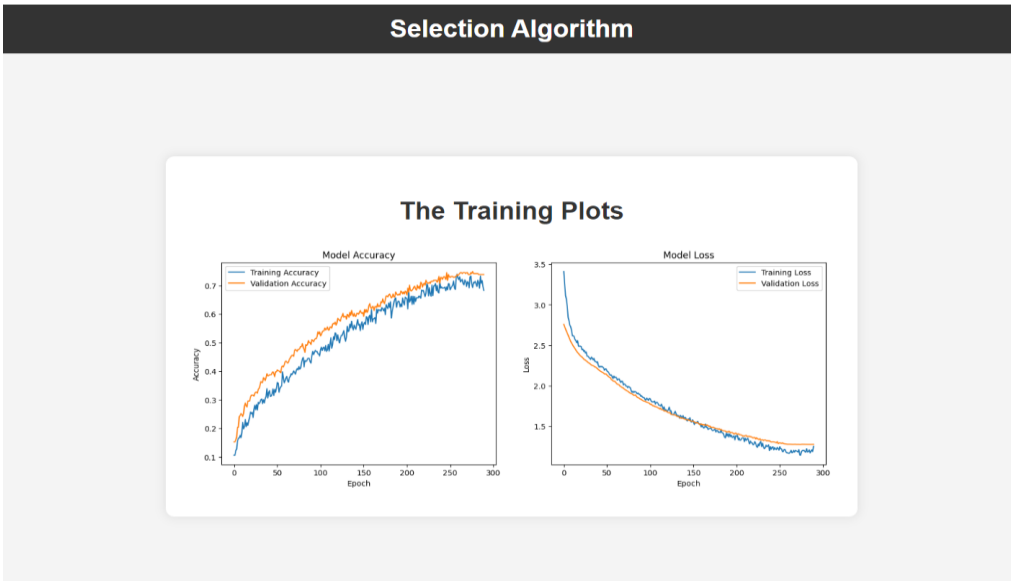


Figure 4.5: ASM training plot

4- Test the model button can test the model performance for any given data the user want to try by uploading the testing data for the model.

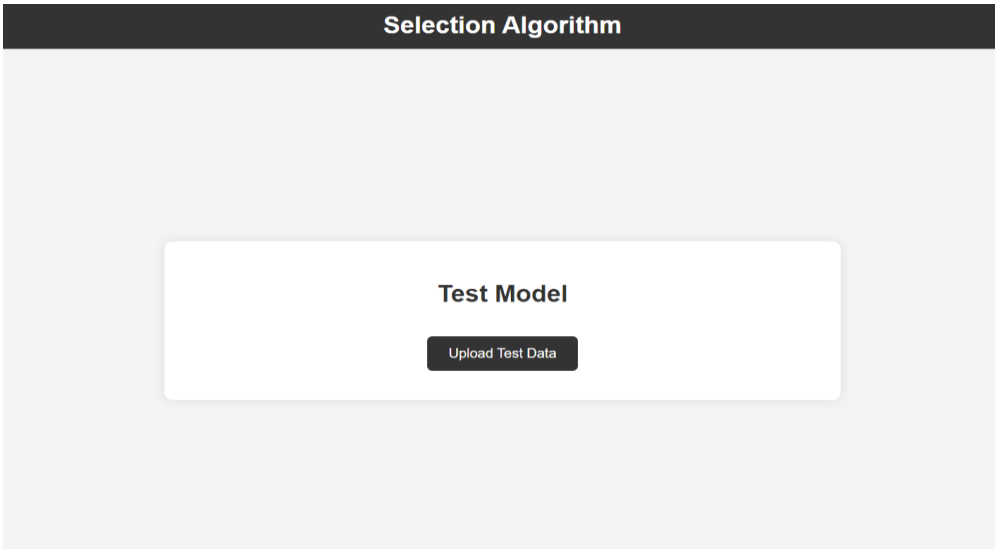


Figure 4.6: Testing model interface

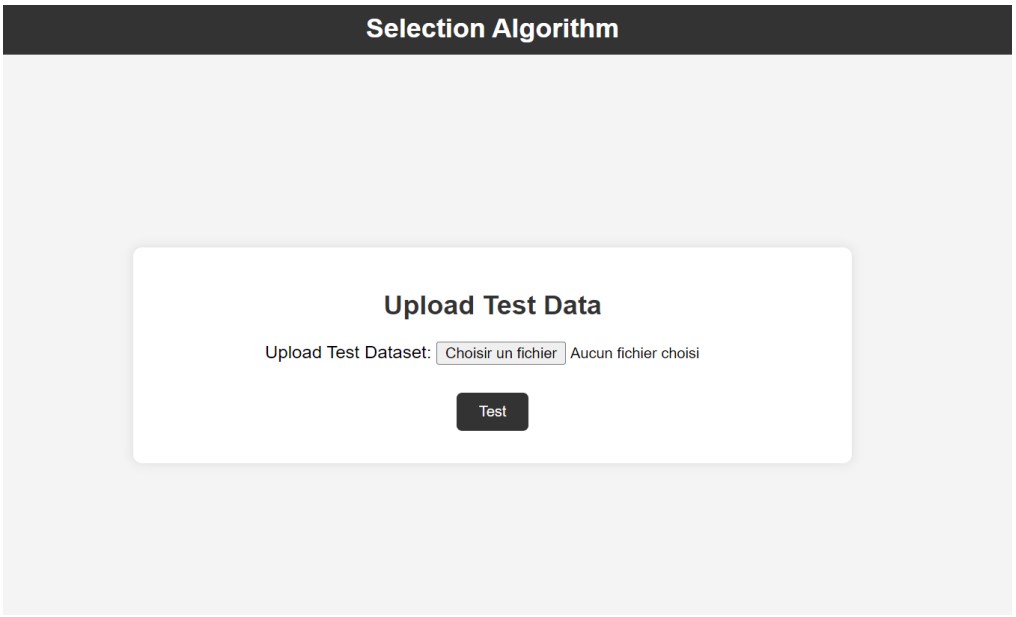


Figure 4.7: Uploading the testing data

5- Testing the model will present the confusion matrix with the choice of downloading test reports and the predictions:

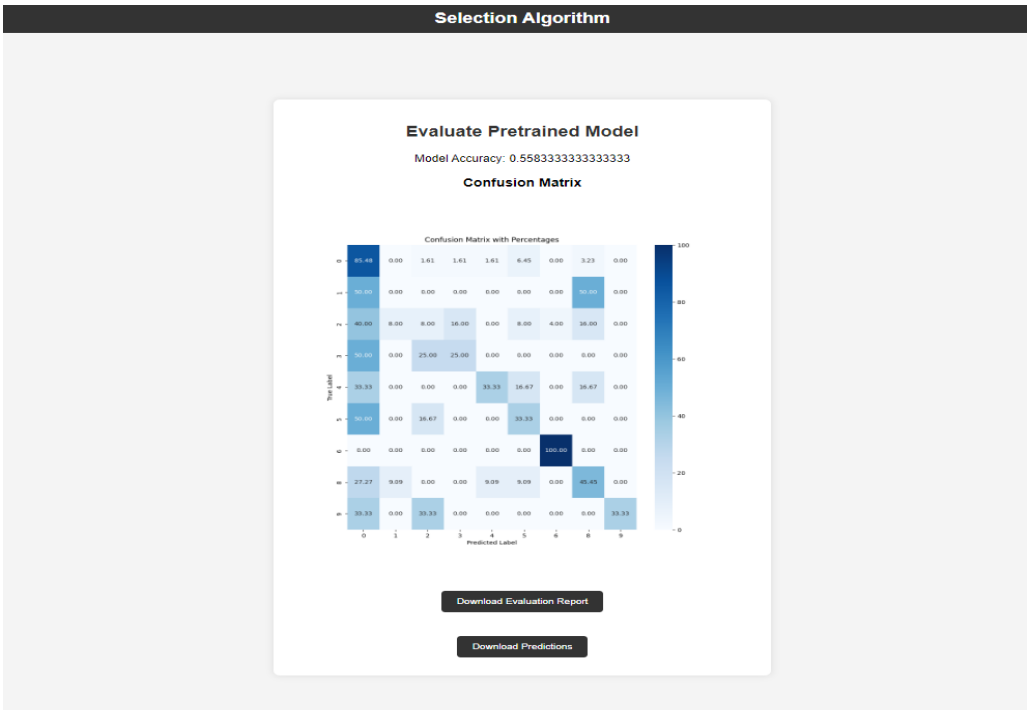


Figure 4.8: Testing results

6- If the user want to train the model on a new data set the option to train the model is available:

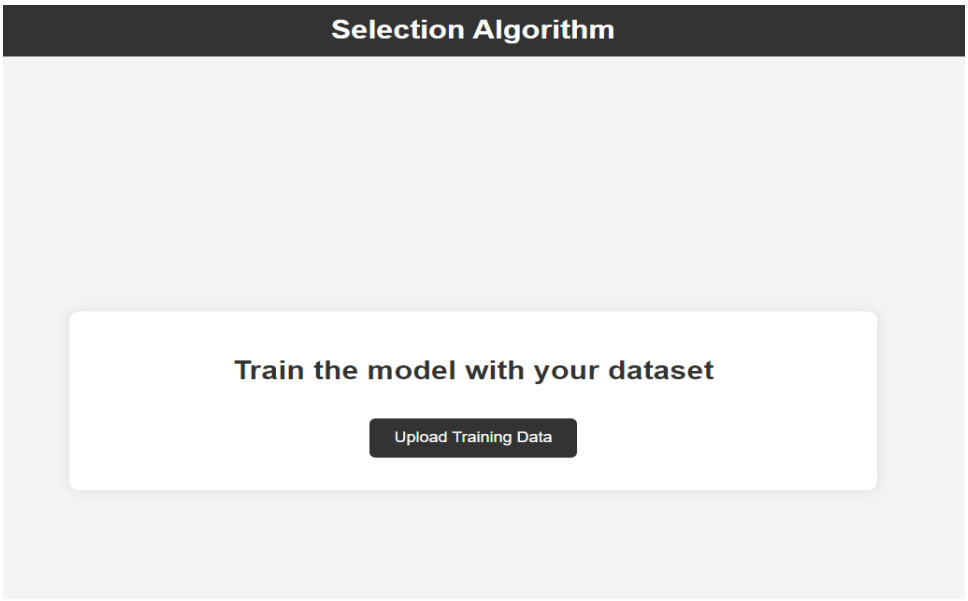


Figure 4.9: Train model on new dataset

7- Upload the new dataset and start the training process and when the model finish training the results can be accessed.

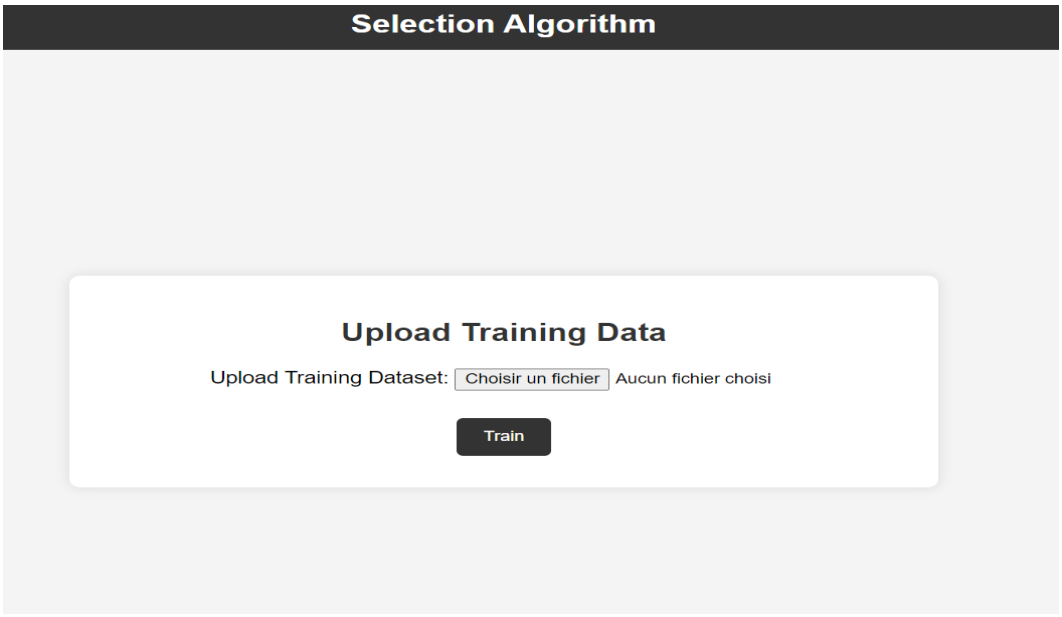


Figure 4.10: Uploading the training dataset

8- The final interface is for presenting the results of the training

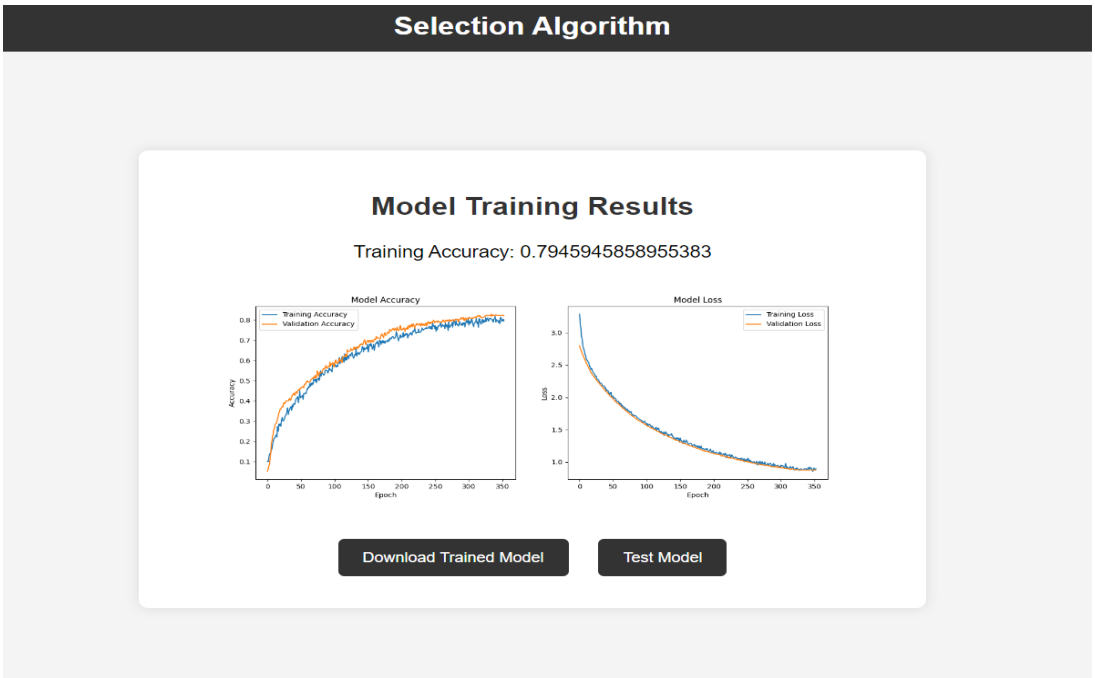


Figure 4.11: Results from training

4.8 Used libraries and frameworks

4.8.1 TensorFlow

TensorFlow, originally created by researchers at Google, is the most popular one among the plethora of deep learning libraries. The term 'Tensor' refers to an N-dimensional array, while 'Flow' denotes the concept of performing calculations based on a data flow diagram."Tensorfow is a system that transfers complex data structures to artifcial neural networks for analysis and processing. It can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java, [20].

Figure (4.12) TensorFlow Logo

4.8.2 Keras

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. Keras also gives the highest priority to crafting great documentation and developer guides.Keras is a deep learning API written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch. Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. Keras also gives the highest priority to crafting great documentation and developer guides. [21]

Figure (4.13) : Keras Logo

4.8.3 PyTorch

PyTorch is a python package that provides the following two advanced features, one is GPU-based tensor calculation, like NumPy, in most cases can replace NumPy. The second is to build a unique dynamic neural network. PyTorch uses a technique known as Reverse-mode auto-diferentiation to allow users to change network performance in a zero-delay or overhead manner. Although this technology is not unique to PyTorch, it is one of the fastest implementations to date. This is also the biggest advantage of PyTorch compared to other frameworks [20].

Figure (4.14) : Pytorch Logo

4.8.4 Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled"data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal .

Figure (4.15) : Pandas Logo

4.8.5 matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy . It provides an object-Oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter , wxPython , Qt, or GTK. There is also a procedural “pylab” interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.The

Figure (4.16) : Matplotlib Logo

4.8.6 Numpy

is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

Figure(4.17): Numpy logo

4.8.7 Google Colab

Google colab is an interactive notebook environment based on the cloud. It is meant for running python code and has a focus on data analysis as well as machine learning tasks. A basic need for collaborative and remote python data science work is Google Colab. Moreover, Google Colab can be applied in coding tutorials and demonstrations that are interactive although this might not be obvious at first glance because it is heavily utilized in data science.

Figure (4.18) : Google Colab logo

4.9 Our machine capabilities:

The results achieved using a laptop with the specification shown at the figure below:

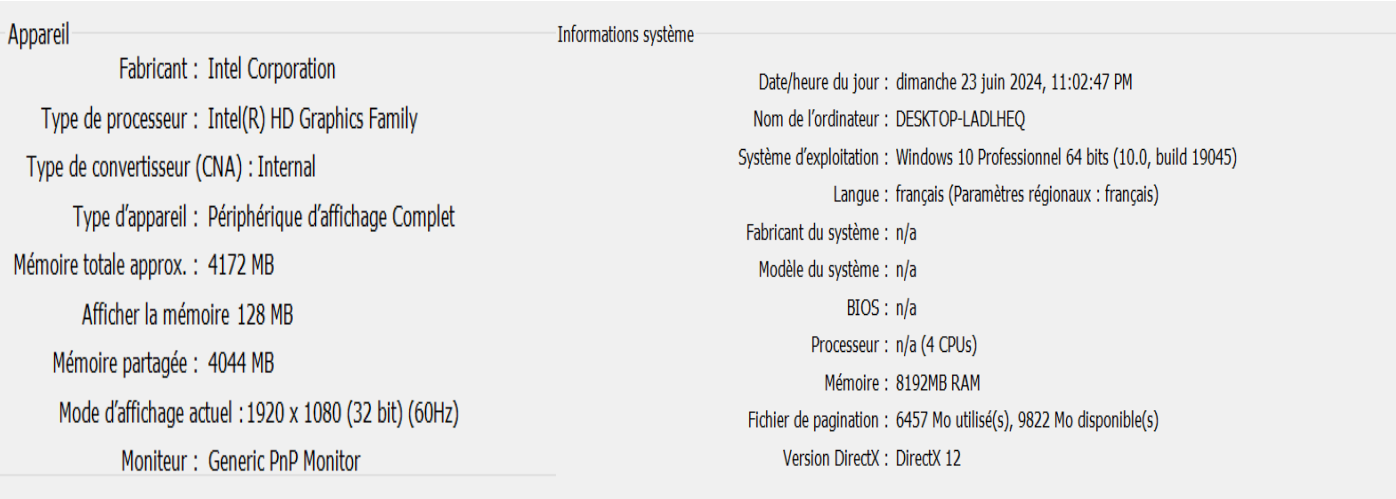


Figure 4.19: Our machine specifications

Conclusion

In conclusion, our study on automated algorithm selection for continuous black-box optimization problems covered the experimental setup, methodology, and results. Through extensive experiments, our deep learning model demonstrated the ability to predict the most suitable solver for a given problem instance. Specifically, we found that BIPOPsaACM outperformed other solvers in terms of robustness for the majority of cases. These findings highlight the effectiveness of our approach and the potential of deep learning models for algorithm selection in optimization problems.

General conclusion

This dissertation was interested in Algorithm Selection problem (ASP) in black box continuous optimization.

The proposed approach considers the ASP task as a classification problem we based the selection process on feature selection where we employ the exploratory landscape analysis technique to Low-level features gathered by systematic sampling of the function on the feasible set are used to predict the most optimal-performing algorithm out of a given portfolio The algorithm selection problem ASP doesn't have a rule of thumb general way of

implementation hence researchers all over the world try to develop their unique specific methods to solve this problem,

we chose in our work to rely on the Exploratory landscape analysis (ELA) techniques to visualize the landscape of the problem features along with several high performing algorithm-portfolio to build a deep learning model that can predict the optimal solver (algorithm) to any given problem set.

The first step we took is the collection and treatment of data needed for the model next comes the treatment of the said data for the subsequent step of preprocessing, then the model building and training after that we present the results, we managed to achieve along with graphic representation we finish with a conclusion.

The results we have obtained are promising. However, it is important to note that while our model outperforms the state-of-the-art study in certain group functions, it falls short in others.

This project has been beneficial to us in multiple ways. It has allowed us to acquire new knowledge and strengthen our existing skills. We have delved into the field of black box optimization and algorithm selection, with a specific emphasis on feature characterization using ELA and deep learning techniques.

5 Bibliography

[1] Kerschke, P., & Trautmann, H. (2019). Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. Evolutionary Computation, 27(1), 99–127. https://doi.org/10.1162/evco_a_00236

[2] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, vol. 200, 2011, doi: https://doi.org/10.1145/2001576.2001690

[3] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge, "Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges," *Information Sciences*, vol. 317, pp. 224–245, Oct. 2015, doi: https://doi.org/10.1016/j.ins.2015.05.010. Available: https://www.sciencedirect.com/science/article/abs/pii/S0020025515003680

[4] J. R. Rice, "The Algorithm Selection Problem," *Advances in Computers*, pp. 65–118, 1976, doi: https://doi.org/10.1016/s0065-2458(08)60520-3

[5] A. Tornede, "Advanced Algorithm Selection with Machine Learning: Handling Large Algorithm Sets, Learning From Censored Data, and Simplyfing Meta Level Decisions," *6*, 2023, doi: https://doi.org/10.17619/UNIPB/1-1780

[6] I. P. Gent *et al.*, "Learning When to Use Lazy Learning in Constraint Solving," *European Conference on Artificial Intelligence*, pp. 873–878, Aug. 2010, doi: https://doi.org/10.3233/978-1-60750-606-5-873

[7] M. Milano and A. Guerri, "Learning Techniques for Automatic Algorithm Portfolio Selection. Learning techniques for Automatic Algorithm Portfolio Selection," 2004.

[8] Y. Malitsky, A. Sabharwal, Horst Samulowitz, and Meinolf Sellmann, "Non-Model-Based Algorithm Portfolios for SAT," *Lecture notes in computer science*, pp. 369–370, Jan. 2011, doi: https://doi.org/10.1007/978-3-642-21581-0_33

[9] Hough, Patricia Diane, and Pamela J. Williams Modern Machine Learning for Automatic Optimization Algorithm Selection. No. SAND2006-5746C. Sandia National Lab.(SNL-CA), Livermore, CA (United States), 2006.

[10] S. Haim and T. Walsh, “Restart Strategy Selection using Machine Learning Techniques,” *arXiv (Cornell University)*, Jan. 2009, doi: https://doi.org/10.48550/arxiv.0907.5032

[11] Ilya Loshchilov, M. Schoenauer, and M. Sebag, “Bi-population CMA-ES algorithms with surrogate models and line searches,” Jul. 2013, doi: https://doi.org/10.1145/2464576.2482696.

[12]Brockhoff, D., Auger, A., & Hansen, N. (2012). On the effect of mirroring in the IPOP active CMA-ES on the noiseless BBOB testbed. HAL (Le Centre Pour La Communication Scientifique Directe). https://doi.org/10.1145/2330784.2330824

[13] László Pál, “Benchmarking a hybrid multi level single linkagealgorithm on the bbob noiseless testbed,” Jul. 2013, doi: https://doi.org/10.1145/2464576.2482692.

[14]A.atamna,& Nikolaus Hansen (2015)Benchmarking IPOP-CMA-ES-TPA and IPOP-CMA-ES-MSR on the BBOB Noiseless Testbed https://doi.org/10.1145/2739482.2768467

[15] A. Howe, “Learned models of performance for many planners,” 2014.

[16]Petr Pošík, & Václav Klemš. (2012). JADE, an adaptive differential evolution algorithm, benchmarked on the BBOB noiseless testbed. https://doi.org/10.1145/2330784.2330814

[17]Aljoša Vodopija, Tea Tušar, & Bogdan Filipič. (2018). Comparing black-box differential evolution and classic differential evolution. Proceedings of the Genetic and Evolutionary Computation Conference Companion. https://doi.org/10.1145/3205651.3208309

[18] F. Hutter, Youssef Hamadi, H. H. Hoos, and K. Leyton-Brown, “Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms,” *Lecture notes in computer science*, pp. 213–228, Jan. 2006, doi: https://doi.org/10.1007/11889205_17

[19] K. Leyton-Brown, E. Nudelman, and Y. Shoham, “Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions,” *Lecture notes in computer science*, pp. 556–572, Jan. 2002, doi: https://doi.org/10.1007/3-540-46135-3_37

[20] Z. Wang, K. Liu, J. Li, Y. Zhu, and Y. Zhang, “Various Frameworks and Libraries of Machine Learning and Deep Learning: A Survey,” *Archives of Computational Methods in Engineering*, Feb. 2019, doi: https://doi.org/10.1007/s11831-018-09312-w.

[21] Staudemeyer, Ralf C and E. R. Morris, “Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks,” *arXiv (Cornell University)*, Sep. 2019, doi: https://doi.org/10.48550/arxiv.1909.09586.

[22]Fialho, Á., Gong, W., & Cai, Z. (2010). Probability matching-based adaptive strategy selection vs. uniform strategy selection within differential evolution. CiteSeer X (the Pennsylvania State University). https://doi.org/10.1145/1830761.1830769

[24] T. Cunha, C. Soares, and F. de, “A label ranking approach for selecting rankings of collaborative filtering algorithms,” Apr. 2018, doi: https://doi.org/10.1145/3167132.3167418

[25] L. Kotthoff, “Ranking Algorithms by Performance,” *arXiv (Cornell University)*, Jan. 2013, doi: https://doi.org/10.48550/arxiv.1311.4319

[26] R. Quantumblack and H. Lau, “Algorithm selection via ranking,” 2015.

[27] J. Hanselle, A. Tornede, M. Wever, and Eyke Hüllermeier, “Hybrid Ranking and Regression for Algorithm Selection,” *Lecture notes in computer science*, pp. 59–72, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-58285-2_5

[30] C. C. Aggarwal, *Neural Networks and Deep Learning : A Textbook*. Cham: Springer International Publishing, 2018.

[31] Petr Pošík and Petr Baudiš, “Dimension Selection in Axis-Parallel Brent-STEP Method for Black-Box Optimization of Separable Continuous Functions,” Cvut DSpace (Czech Technical University), Jul. 2015, doi. https://doi.org/10.1145/2739482.2768469.

[32] L. Fehring, J. Hanselle, and A. Tornede, “HARRIS: Hybrid Ranking and Regression Forests for Algorithm Selection,” *arXiv (Cornell University)*, Jan. 2022, doi: https://doi.org/10.48550/arxiv.2210.17341

[33] Serdar Kadioglu, Y. Malitsky, Meinolf Sellmann, and K. Tierney, “ISAC --Instance-Specific Algorithm Configuration,” *European Conference on Artificial Intelligence*, pp. 751–756, Aug. 2010, doi: https://doi.org/10.3233/978-1-60750-606-5-751

[35] J. C. Ortiz-Bayliss, I. Amaya, J. M. Cruz-Duarte, A. E. Gutierrez-Rodriguez, S. E. Conant-Pablos, and H. Terashima-Marín, “A General Framework Based on Machine Learning for

Algorithm Selection in Constraint Satisfaction Problems," *Applied Sciences*, vol. 11, no. 6, p. 2749, Mar. 2021, doi: https://doi.org/10.3390/app11062749.

[36] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann, "Algorithm Portfolios Based on Cost-Sensitive Hierarchical Clustering Algorithm Portfolios Based on Cost-Sensitive Hierarchical Clustering," 2013.

[37] R. Amadini, M. Gabbrielli, and J. Mauro, "Under consideration for publication in Theory and Practice of Logic Programming SUNNY: a Lazy Portfolio Approach for Constraint Solving," 2014.

[38] D. Stern, H. Samulowitz, R. Herbrich, T. Graepel, L. Pulina, and A. Tacchella, "Collaborative Expert Portfolio Management," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, no. 1, pp. 179–184, Jul. 2010, doi: https://doi.org/10.1609/aaai.v24i1.7561

[39]: M. Misir and M. Sebag, "Algorithm Selection as a Collaborative Filtering Problem," 2013

[41] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," *Archives of Computational Methods in Engineering*, Jun. 2019, doi: https://doi.org/10.1007/s11831-019-09344-w

[42] M. Mısır and M. Sebag, "Alors: An algorithm recommender system," *Artificial Intelligence*, vol. 244, pp. 291–314, Mar. 2017, doi: https://doi.org/10.1016/j.artint.2016.12.001

[43] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[44] S. M. B. P. Samarakoon, M. A. V. J. Muthugala, A. Vu Le, and M. R. Elara, "hTetro-Infi: A Reconfigurable Floor Cleaning Robot With Infinite Morphologies," *IEEE Access*, vol. 8, pp. 69816–69828, 2020, doi: https://doi.org/10.1109/access.2020.2986838.

[45] I. Nusrat and S.-B. Jang, "A Comparison of Regularization Techniques in Deep Neural Networks," *Symmetry*, vol. 10, no. 11, p. 648, Nov. 2018, doi: https://doi.org/10.3390/sym10110648.

[46] Y. Wu *et al.*, "Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks," *arXiv (Cornell University)*, Dec. 2019, doi: https://doi.org/10.1109/bigdata47090.2019.9006104.

[47] L. N. Smith, “A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay,” *arXiv.org*, Apr. 24, 2018. https://arxiv.org/abs/1803.09820v2

[48] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv.org*, Jun. 15, 2017. https://arxiv.org/abs/1609.04747?ref=ruder.io

[49] G. Naidu, Tranos Zuva, and Elias Mmbongeni Sibanda, “A Review of Evaluation Metrics in Machine Learning Algorithms,” *Lecture notes in networks and systems*, pp. 15–25, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-35314-7_2.

[50] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, no. 1, Mar. 2021, doi: https://doi.org/10.1186/s40537-021-00444-8.

[51] D. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *Computer Science*, 2014, doi: https://doi.org/10.48550/arXiv.1412.6980.

[52] M. C. Mukkamala and M. Hein, “Variants of RMSProp and Adagrad with Logarithmic Regret Bounds,” *arXiv.org*, Nov. 28, 2017. https://arxiv.org/abs/1706.05507

[53] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv.org*, Dec. 22, 2012. http://arxiv.org/abs/1212.5701 (accessed Jul. 25, 2023).

[54]C. Darken, J. Chang and J. Moody, "Learning rate schedules for faster stochastic gradient search," *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, Helsingoer, Denmark, 1992, pp. 3-12, doi: 10.1109/NNSP.1992.253713.

[55] Raphael Patrick Prager, Moritz Vinzent Seiler, H. Trautmann, and Pascal Kerschke, “Automated Algorithm Selection in Single-Objective Continuous Optimization: A Comparative Study of Deep Learning and Landscape Analysis Methods,” *Lecture notes in computer science*, pp. 3–17, Jan. 2022, doi: https://doi.org/10.1007/978-3-031-14714-2_1

[56] S. Xu, W. Liu, C. Wu, and J. Li, “CNN-HT: A Two-Stage Algorithm Selection Framework,” *Entropy*, vol. 26, no. 3, pp. 262–262, Mar. 2024, doi: https://doi.org/10.3390/e26030262

[57] R. P. Prager, M. Vinzent Seiler, H. Trautmann, and P. Kerschke, “Towards Feature-Free Automated Algorithm Selection for Single-Objective Continuous Black-Box Optimization,”

Bibliography

IEEE Xplore, Dec. 01, 2021. doi: https://doi.org/10.1109/SSCI50451.2021.9660174. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9660174

5.1 Webographie

[28] http://www.flacco.shinyapps.io/flacco

[29] https://numbbo.github.io/data-archive/bbob/

[34] https://keras.io/

[40] https://intellipaat.com/blog/loss-functions-in-deep-learning/