CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

AN ADAPTIVE HYBRID GENETIC ALGORITHM FOR HYPERPARAMETER OPTIMIZATION

Cláudio André da Silva Alves

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do grau de mestre.

Orientadores:
Pedro Henrique González Silva

Rio de Janeiro,

Agosto de 2023

An adaptive hybrid genetic algorithm for hyperparameter optimization

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Cláudio André da Silva Alves

Aprovada por:

Presidente, Prof. Pedro Henrique González Silva, D.Sc. (orientador)

Prof. Eduardo Bezerra da Silva, D.Sc.

Prof. Glauco Fiorott Amorim, D.Sc.

Prof. Israel Mendonça dos Santos, D.Sc.

Prof. Vanessa de Almeida Guimarães, D.Sc.

Rio de Janeiro,

Agosto de 2023

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

A474 Alves, Cláudio André da Silva
An adaptive hybrid genetic algorithm for hyperparameter optimization / Cláudio André da Silva Alves. — 2023.
29f. : il. color. , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2023.
Bibliografia : f. 26-29
Orientador: Pedro Henrique González Silva

1. Otimização combinatória. 2. Algoritmos genéticos. 3. Programação heurística. 4. Aprendizado de máquina. I. Silva. Pedro Henrique González. (Orient.). II. Título.

CDD 004.36

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

DEDICATÓRIA

Dedico este trabalho, aos meus amados pais
Maria e Ciraldo pelo amor, carinho e orientação
desde os meus primeiros passos, e minha irmã
Tamires pelo companheirismo, amizade e
paciência nessa minha jornada.

AGRADECIMENTOS

Agradeço primeiramente à Deus e minha família que sempre me apoiaram e em todos os momentos se dispuseram à ajudar. Não haveria espaço suficiente para listar todos aqueles que me ajudaram nesse plano espiritual e fora dele, à eles deixo meu profundo muito obrigado.

Ao Prof. Dr. Pedro Henrique González Silva agradeço pela orientação, amizade, confiança e por contribuir com minha formação de forma tão espetacular. Sou extremamente grato.

RESUMO

An adaptive hybrid genetic algorithm for hyperparameter optimization

Cláudio André da Silva Alves

Orientadores:
Pedro Henrique González Silva

Resumo da Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do grau de mestre.

O recente aumento na popularidade das aplicações de aprendizado de máquina (AM) levou a um aumento na demanda por modelos de AM eficientes. Uma das principais etapas na construção de tais modelos é selecionar um conjunto adequado de hiperparâmetros. No entanto, com o aumento da complexidade dos modelos e técnicas de treinamento, definir manualmente esses parâmetros tornou-se uma tarefa trabalhosa, exigindo uma quantidade significativa de tempo e conhecimento específico sobre o modelo que está sendo ajustado. Para enfrentar esse desafio, a comunidade do Automatic Machine Learning (AutoML) está se concentrando em criar maneiras de encontrar automaticamente o melhor conjunto de hiperparâmetros para algoritmos de AM por meio de sua área de pesquisa chamada de *Hyperparameter Optimization (HPO)*. Recentemente, o *Hybrid Biased Random Key Genetic Algorithm (HBRKGA)*, um algoritmo genético que usa funções de otimização substitutas na etapa de *exploitation*, tem sido usado para encontrar hiperparâmetros automaticamente de forma eficiente para diferentes conjuntos de dados. No entanto, seu potencial não foi totalmente explorado, pois o HBRKGA usa apenas uma função substituta fixa na etapa de *exploitation*. Esta pesquisa apresenta uma nova abordagem para HPO de modelos de AM baseados no HBRKGA. Um método chamado *Adaptive HBRKGA* (A-HBRKGA) é desenvolvido para melhorar a probabilidade de encontrar a melhor solução. Este método é baseado no princípio de que diferentes passos evolutivos requerem diferentes funções de otimização, o que permite ao HBRKGA ter múltiplas funções substitutas que são escolhidas com base em avaliações anteriores. A abordagem foi testada em vários conjuntos de dados disponíveis publicamente e apresenta melhores resultados quando comparada a outros métodos da literatura.

Palavras-chave:
otimização, otimização de hiperparâmetros, algoritmos genéticos, algoritmos evolutivos, metaheurística

Rio de Janeiro,

Agosto de 2023

ABSTRACT

An adaptive hybrid genetic algorithm for hyperparameter optimization

Cláudio André da Silva Alves

Advisors:
Pedro Henrique González Silva

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

The recent spike in the popularity of machine learning (ML) applications has led to an increased demand for efficient ML models. One of the key steps in building such models is selecting a well-suited set of hyperparameters. However, with the increasing complexity of models and training techniques, manually defining these parameters has become a labor-intensive task, requiring a significant amount of time and specific knowledge about the model being tuned. To address this challenge, the community of AutoML is focusing on devising ways to automatically find the best set of hyperparameters for ML algorithms through its research area called hyperparameter optimization (HPO). Recently, Hybrid Biased Random-Key Genetic Algorithm (HBRKGA), a Genetic Algorithm (GA) that uses surrogate optimization functions at the exploitation step, has been used to efficiently find automatic hyperparameters for different datasets. However, its potential has not been fully explored as HBRKGA uses only one fixed surrogate function at the exploitation step. This research presents a novel approach for HPO of ML models based on HBRKGA. A method called Adaptive HBRKGA (A-HBRKGA) is devised to improve the probability of finding the best solution. This method is based on the principle that different evolutionary steps require different optimization functions, which allows HBRKGA to have multiple surrogate functions that are chosen based on past evaluations. The approach has been tested against several publicly available datasets and it is shown that it presents better results when compared to other methods in the literature.

Key-words:
optimization, hyperparameter optimization, genetic algorithms, evolutionary algorithms, metaheuristic

Rio de Janeiro,

Agosto de 2023

Contents

List of Figures

List of Tables

List of Abreviations

ANN	Artificial Neural Network	13, 21
AUTOML	Automatic Machine Learning	12
BO	Bayesian Optimization	17
BRKGA	Biased Random Key Genetic Algorithm	20
CMA-ES	Covariance Matrix Adaptation Evolution Estrategy	12, 18
CNN	Convolutional Neural Network	22
HBRKGA	Hybrid Biased Random Key Genetic Algorithm	13
HPO	Hyperparameter Optimization	15
MLP	Multilayer Perceptron	13
RKGA	Random-Key Genetic Algorithm	19

Chapter I Introduction

The constant growth in computational power and data availability has led to a significant increase in interest in machine learning research. At the same time, the complexity of models has also increased as the demand for specialized tasks has grown rapidly, with models becoming increasingly tailored to specific tasks rather than general-purpose tools. In order to make these models be trained effectively. The parameters specific of algorithms been used are chosen based on data and experience, and a proper selection can have a significant impact on training time, required infrastructure, and model performance [Hutter et al., 2019]. As model complexity increases, the number of parameters also grows, making the task of determining an optimal combination of hyperparameters an important aspect of building machine learning models. Manually finding a good combination can be a tedious and ineffective job [Yang and Shami, 2020], so the need to find a way to automatically define those parameters has also increased. The Automatic Machine Learning (AutoML) field has a specific area of study dedicated to finding these hyperparameters, known as Hyperparameter Optimization (HPO). Hyperparameter Optimization can improve the performance of a model [Frank Hutter, 2019], but it also has several challenges, as it usually can not make use of traditional optimization algorithms. For instance, its objective function lacks features of traditional optimization problems, such as convexity [Frank Hutter, 2019].

Illustrating this growing increase in the number of parameters and consequently hyperparameters in recent machine learning models, Figure I.1 lists the number of parameters in the main models over the years, even when comparing models with different uses such as VGG16, ResNet-50 for computational vision or BERT and GPT-3 for natural language processing (NLP). Over the years, is possible to note the growing trend to create bigger and more powerful models boosted by more accessible computational power and data availability [Bernstein et al., 2021].

There are many methods, or variations of them, being used to search for an optimal solution in HPO. One can highlight Random Search [Bergstra and Bengio, 2012], Grid Search [Barbero et al., 2007], Bayesian Optimization [Brochu et al., 2010; Snoek et al., 2012], and Covariance Matrix Adaptation Evolution Estrategy (CMA-ES) [Hansen, 2006]. Random Search is an approach that randomly explores the search space, constrained to a limited time and/or resource usage. Grid Search also explores the search space, but exhaustively in a fixed domain of values. On the other hand, Bayesian Optimization works by evaluating the previous value to determine the next one,

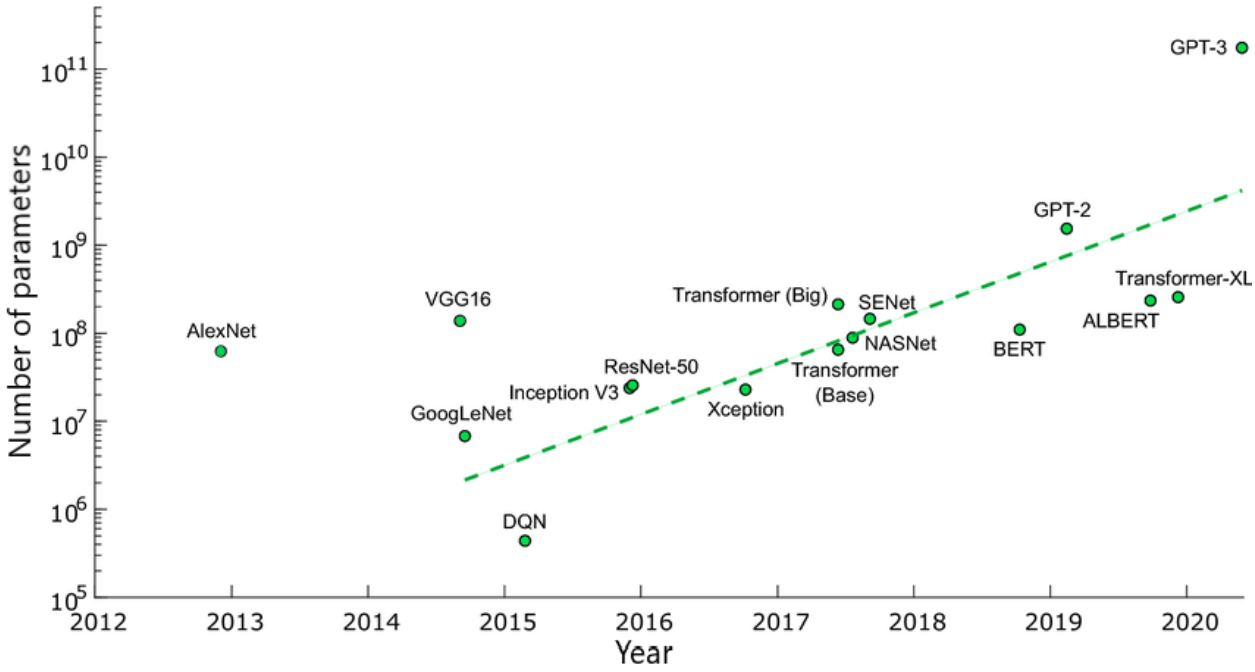

Figure I.1: Number of parameters in recent neural network architectures
Legend: Adapted from Bernstein et al. [2021]

avoiding many unnecessary iterations and finding the best results earlier [Yang and Shami, 2020]. CMA-ES sample solutions based on a multivariate Gaussian distribution updating its parameters according to evaluation values.

In this research, a novel approach based on the method proposed by [Japa et al., 2022] is presented. Our approach is a step further in improving the exploration step by allowing Hybrid Biased Random Key Genetic Algorithm (HBRKGA) to have more than one surrogate function, and in an adaptive manner select which one would yield the best results. Five hyperparameters of a Multilayer Perceptron (MLP), a fully connected Artificial Neural Network (ANN) were optimized through experimentation; the results show that the proposed method generates competitive ANNs, with advantages when compared to traditional models. Our main contributions are the following:

1. The improvement of HBRKGA by adding an adaptive component and allowing for the use of multiple surrogate functions.

2. Providing experimental data that demonstrates the superiority of the proposed method when compared to the baseline techniques like Grid Search, Random Search, Bayesian Optimization, and CMA-ES.

3. Providing further comparisons with other commonly used HPO algorithms.

The development and deployment of a machine learning application address several steps [Russell and Norvig, 2021]. Figure I.2 shows these main tasks enumerated. It all starts with data preparation where data preprocessing activities like data cleaning or feature engineering will be applied to assure

the integrity and quality of data. In sequence, the algorithms' hyperparameters are defined. Next, the algorithm is trained using hyperparameters and training data to create a model aforementioned. In the sequence, the model is evaluated with unseen examples from test data. Normally, these three previous steps are repeated until the stop criteria is met. Finally, with the achievement of minimum requirements established, the model can be deployed to the production environment.

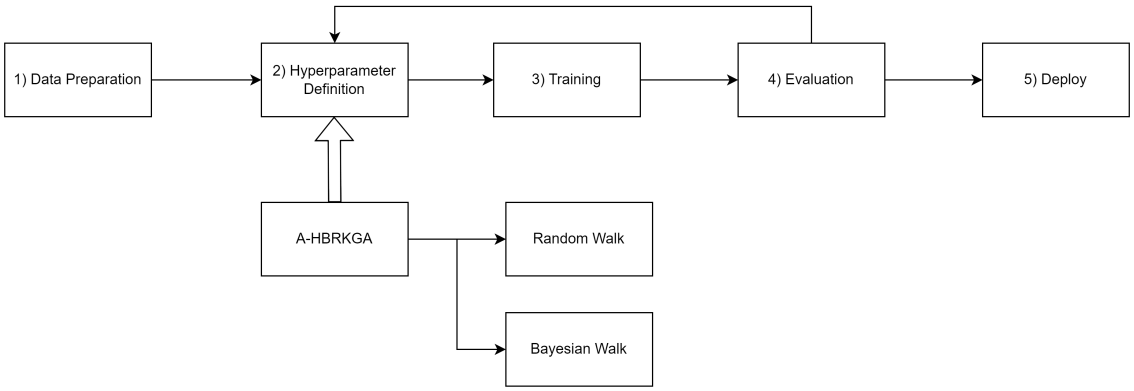

Figure I.2: Machine Learning workflow
Legend: Prepared by the author

The proposed approach, A-HBRKGA, is located at the hyperparameter definition, it's responsible for generating candidate solutions that must improve the overall performance of the model.

The remainder of this document is structured as follows: In Chapter II, we describe important concepts used in this work. In Chapter III, we describe related works in the field of hyperparameter optimization; In Chapter IV we present the proposed method. In Chapter V, the computational experiments are shown. Finally, Chapter VI presents the conclusion and future directions.

Chapter II Background

This chapter presents the fundamental concepts used in this work. Section II.1 describes hyperparameters and some strategies used for their optimization. Section II.2 presents BRKGA, the base for proposed strategy. Section II.3 explains artificial neural networks.

II.1 Hyperparameter and principal optimization strategies

In machine learning exists two types of parameters: model parameters are estimated from a dataset during the training step. After that, these parameters are incorpored to model and used to make predictions or classify new data; hyperparameters are specific to algorithm and not to dataset, so they can't be learned from data and must be set previously [Yang and Shami, 2020].

Hyperparameter Optimization (HPO) is the process of experimenting different values and selecting ones that give best performance. The operation of a hyperparameter search algorithm can be summarized in the iterative process consisting of evaluating the metrics obtained with the current configuration and proposing a new hyperparameter configuration with the potential to improve the metrics. The next subsections presents some of these strategies: Grid Search (Subsection II.1.1), Random Search (Subsection II.1.2), Bayesian Optimization (Subsection II.1.3) and CMA-ES (Subsection II.1.4).

II.1.1 Grid Search

Grid Search it's the most basic HPO method, it performs an exhaustive search considering a manually specified a multi-dimensional grid of hyperparameters [Bergstra and Bengio, 2012]. It is a simple but computationally expensive method for hyperparameter optimization.

Grid Search is negatively impacted by the curse of dimensionality since the required evaluation grows exponentially with the dimensionality of the configuration space [Hutter et al., 2019]. The curse of dimensionality refers to the fact that as the number of dimensions in a dataset increases, the amount of data to train a model effectively also increases [Debie and Shafi, 2019]. Grid Search performance depends on the search space defined, including the step used to change from one hyperparameter to another. Figure II.1 shows the Grid Search procedure explores all search space until return the best hyperparameter solution found in red.

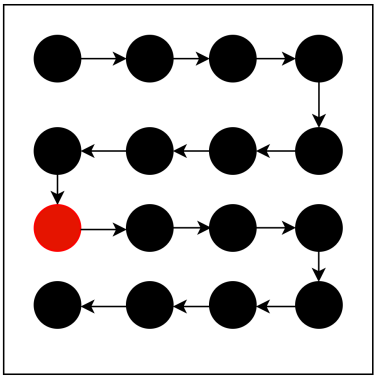

Figure II.1: Grid Search strategy for hyperparameter optimization problem.
Legend: Every point represents a solution of a combination of hyperparameters. This technique finds the best solution (red point) after running a sequential path. Adapted from Bergstra and Bengio [2012]

II.1.2 Random Search

Random Search sample configurations stochastically from a user-defined subspace. Compared to Grid Search it's more computationally efficient to the effect that it doesn't combine all values in a multi-dimensional grid of hyperparameters [Bergstra and Bengio, 2012].

Random Search's main advantages are easy implementation and parallelization since each hyperparameter configuration can be generated and evaluated independently. It's very used as a baseline method because is non-dependent of algorithm [Hutter et al., 2019]. Figure II.2 shows the Random Search procedure until finding the best hyperparameter solution in red after some trials.

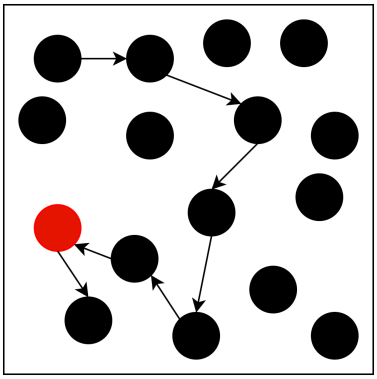

Figure II.2: Random Search strategy for hyperparameter optimization problem.
Legend: Every point represents a solution of a combination of hyperparameters values randomly generated. This technique finds the best solution (red point) after running some random solutions trials. Adapted from Bergstra and Bengio [2012]

II.1.3 Bayesian Optimization

Bayesian Optimization (BO) is an iterative algorithm, unlike Grid Search and Random Search, determines the future evaluation solutions based on previous results [Snoek et al., 2012]. Bayesian Optimization construct a probabilistic model for a function $f(x)$ to search for better solutions than current found.

BO has two key components: a surrogate model, which attempts to fit existing points with the objective function; acquisition function, to determine the next point to be explored balancing exploration and exploitation. Exploration is to sample points in areas that have not been explored yet, while exploitation samples points in a neighborhood that is most likely to contain global optimum [Yang and Shami, 2020].

Figure II.3 exemplifies a run of Bayesian Optimization with four iterations. Starting with two points, at every iteration of the algorithm, the goal is to maximize the acquisition function to determine the next sampling point from the objective function. The acquisition function incorporates the mean and variance of the predictions across the space to estimate the utility of sampling. Subsequently, the objective function is sampled at the location that corresponds to the highest value of the acquisition function, the Gaussian process is updated with this new information, and the entire process is repeated iteratively [Brochu et al., 2010].

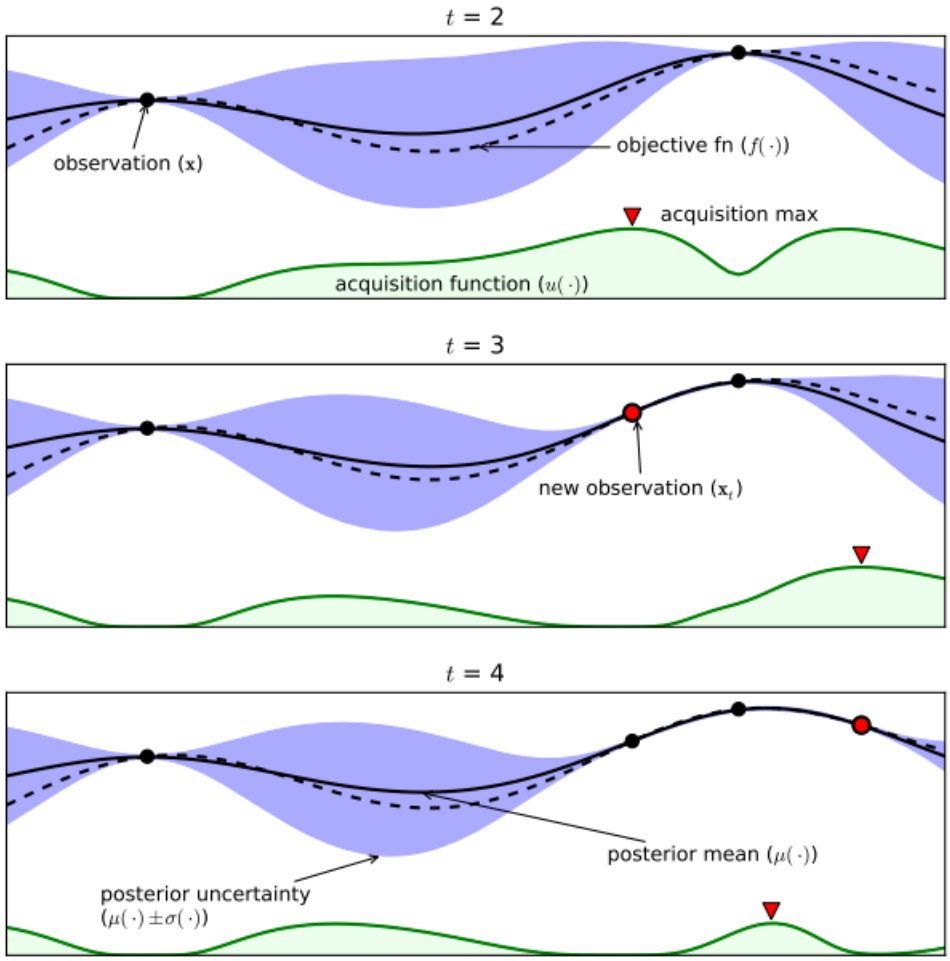

Figure II.3: Bayesian Optimization strategy for hyperparameter optimization problem. Legend: As points are observed, the Gaussian process is adjusted, maing able for the acquisition function to evaluate new solutions. From Brochu et al. [2010]

II.1.4 Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Estrategy (CMA-ES) is an evolutionary algorithm, a population of new search points is generated from normal multivariate distribution sampling. It iteratively evaluates solutions and adjusts the sampling distribution used for the next iteration to give a higher probability to find good solutions [Loshchilov and Hutter, 2016]. It's a derivative-free optimization algorithm, well-suited for high-dimensional problems and problems with a large number of local optima [Loshchilov, 2014].

The computation of the covariance matrix in CMA-ES is carefully designed to leverage both the variance among selected points within a generation and the correlation across generations. This approach aims to fully exploit the information available in the optimization process. The objective of CMA-ES is to adjust the search distribution, represented by a Gaussian probability distribution, to align with the contour lines of the objective function that is being minimized. By

aligning the search distribution with the objective function's contour lines, CMA-ES improves its ability to explore and exploit the search space effectively [Khan, 2018]. This process is shown in Figure II.4, with the progress of each CMA-ES generation, the parameters of the search distribution, including the mean vector and covariance matrix, are dynamically adjusted. This process aims to align the search distribution with the contour lines of the objective function. As a result, as the optimization continues, the evaluation points become increasingly concentrated in the neighborhood of the minimum of the objective function.

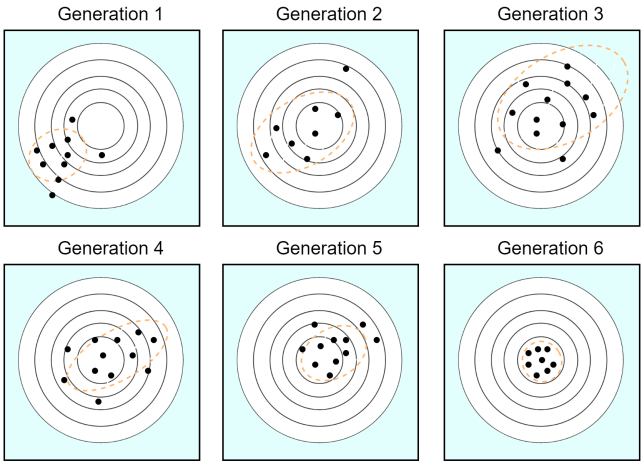

Figure II.4: CMA-ES strategy for hyperparameter optimization problem.
Legend: Adapted from Sentewolf [2023]

II.2 Biased Random Key Genetic Algorithm

Genetic algorithms have Darwin's evolutionary concept as a theoretical foundation regarding the survival of the fittest individuals [Darwin and Knight, 2003]. Genetic algorithms begin with the selection of the best individuals from the initial population according to a metric that evaluates how well that configuration behaved. The best individuals will generate new individuals, that will inherit characteristics that will increase their chances of also being the best individuals of their generation [Sloss and Gustafson, 2020]. Typically, this iterative process is repeated until a configuration is found that exceeds the established limit or the number of generations is reached.

Being an evolutionary algorithm approach, the Random-Key Genetic Algorithm (RKGA) [Bean, 1994] simulates the evolution of a population over generations, where each individual is a candidate solution for the optimization problem. RKGA introduces two fundamental elements to create operators that are independent of the problem: the encoder, responsible for mapping the input to the domain $[0, 1]$, and the decoder, responsible for converting the values back to the original

domain. The Biased Random Key Genetic Algorithm (BRKGA) [Gonçalves and Resende, 2011] is an evolution of the RKGA, introducing a bias in the set of the best individuals of the population in the current generation. During crossing over, these individuals are more likely to pass their genes to their offspring.

II.2.1 Encoder and decoder

In RKGA, a solution γ is represented as a codified vector in the range [0,1] of n elements, and $\bar{\gamma}$ is the solution in the original domain. In our case, each position of γ is a hyperparameter.

Applying a min-max normalization (Eqn. II.1) permits keeping all hyperparameters in the same range of values, γ_i means the i-th component of γ, U_i and L_i specific maximum and minimum values of γ respectively. It's also useful because it can work with different problems just necessitating a particular decoder.

$$\gamma_i = \frac{\overline{\gamma_i} - L_i}{U_i - L_i} \tag{II.1}$$

The purpose of the decoder is to convert a proposed solution to the original domain to be further evaluated by the objective function. This is done by applying Eqn. II.2. Here, the *round* function converts a real number to the closest integer.

$$\overline{\gamma_i} = round(\gamma_i * (U_i - L_i) + L_i) \tag{II.2}$$

The encoder and decoder transformations are schematically represented in Figure II.5. The original domain X is represented on the left side, in this case, each hyperparameter is a component. The right side represents the image set Y, the codified vector. In short, the encoder is a function $f : X \to Y$ and the decoder is the inverse function f^{-1}.

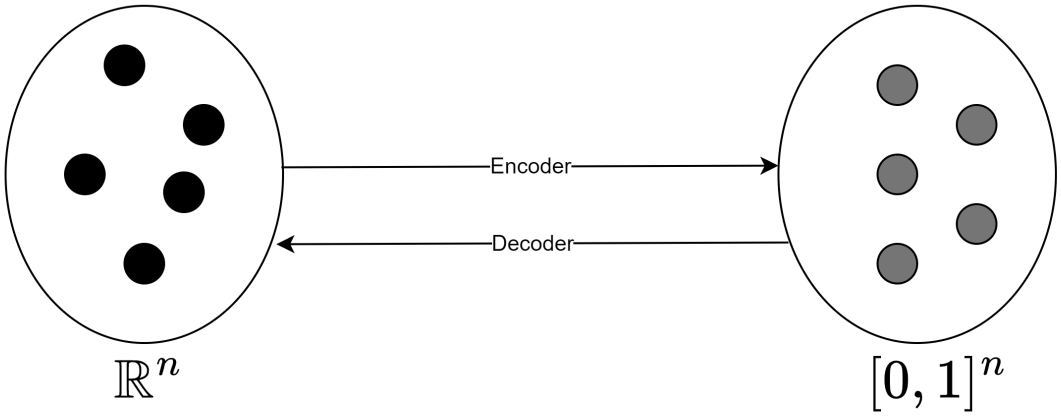

Figure II.5: BRKGA hyperparameters representation
Legend: A real-valued combination of n hyperparameters can be mapped to the $[0,1]^n$ space by the encoder, and be brought back by the decoder. Prepared by the author

II.3 Artificial Neural Networks

Computers are able to learn without being explicity programmed using machine learning [Mitchell and Mitchell, 1997], differently from what occurs in a conventional algorithm where rules are strictly written.

Natural computing, a line of computer science research, develops algorithms inspired by nature or biology as a source of inspiration, for example the ant colony algorithm used in routing problems on a graph Brabazon et al. [2015]. Besides trying to reproduce intelligent behavior, the structure of the solutions can be inspired by humans, for example in Artificial Neural Network (ANN) that are motivated by how the brain works and how synapses occur between neurons. They are composed from diverse layers consisting of simple processing nodes called neurons, each neuron has a set of connections to others neurons. Each layer process the input and pass to next layer until the last layer called output.

ANN has inumerous hyperparameters to be optimized, for example: number of layers, determines the depth of network and it's capacity to model complex patterns from data; number of neurons, number of neurons in each layer, more neurons can allow model learn complex patterns from data but a excessive amount can make model overfit [Srivastava et al., 2014]; learning rate, size of step of gradient algorithm; regularization rate, prevents model to overfit controlling the regularization term added to the cost function [Aggarwal et al., 2018].

Chapter III Related Work

In Florea et al. [2019], an enhanced version of Random Search was developed to optimize the hyperparameters of machine learning algorithms. Unlike traditional methods, their version used a probability of change linked to each hyperparameter. In this new method, at each iteration, only the hyperparameter with the highest probability of change is changed, instead of all of them.

Showing that pure Random Search is still a reasonable approach, Farag et al. [2021] used it to optimize hyperparameters of two networks, ResNet and Xception Net, to diagnose COVID-19 from chest X-ray images.

Following Farag et al. [2021], Shahin et al. [2021] also focused on a solution to identify COVID-19, but using Electrocardiogram data in several Convolutional Neural Network (CNN) models. In one of them, VGG16, they applied Grid Search, successfully enhancing its results, increasing the accuracy, and decreasing the loss. In Lee et al. [2021] work, a genetic algorithm is used to optimize a Convolutional Neural Network (CNN) architecture, in a dataset used for Alzheimer's disease diagnostics. A genetic algorithm is used to optimize a Ling-Short Term Memory (LSTM) to predict the next word in a Nature Language Processing (NLP) domain [Gorgolis et al., 2019].

When talking about Bayesian Optimization, Atteia et al. [2022a] used it to optimize the hyperparameters of a Convolutional Neural Network (CNN) designed to detect leukemia in its early stages. Still in the health sector, Şahin et al. [2022] also Bayesian Optimization to enhance a CNN, designed to identify skin cancer.

Loshchilov and Hutter [2016] propose the use of CMA-ES to optimize hyperparameters of deep neural networks. Comparing against Bayesian Optimization and some variations, the results show comparable results with a lower computational cost.

Genetic Algorithm is the core of our method and also was used by Mohan and Badra [2023] to improve their model, outperforming other models using fewer computational resources. Liashchynskyi and Liashchynskyi [2019] compares the three most popular algorithms for hyperparameter optimization (Grid Search, Random Search, and Genetic Algorithm) to construct a Convolutional Neural Network (CNN) using a neural architecture search approach. The genetic algorithm was the most successful approach in experiments realized.

The research carried out by Japa et al. [2022], from which this work is developed, also uses Genetic Algorithm. A version called HBRKGA is presented with the addition of an exploitation

method at the end of each evolutionary step.

Table III.1 summarizes the related works according to each hyperparameter optimization technique covered in the study. Some works present the study of one or more techniques separated, this research and Japa et al. [2022] cover all cited strategies to comparision.

Table III.1: Summary of related works by strategy

Author	**Grid Search**	**Random Search**	**Bayesian Optimization**	**CMA-ES**	**Genetic Algorithms**
Florea et al. [2019]		✓			
Farag et al. [2021]	✓				
Atteia et al. [2022b]			✓		
Şahin et al. [2022]		✓			
Loshchilov and Hutter [2016]				✓	
Lee et al. [2021]					✓
Mohan and Badra [2023]					✓
Liashchynskyi and Liashchynskyi [2019]	✓	✓			✓
Japa et al. [2022]	✓	✓	✓	✓	✓
This dissertation	✓	✓	✓	✓	✓

Chapter IV Metodology

This chapter presents the methodology developed by the authors. In Section IV.1, Adaptive HBRKGA is presented, having its exploration step further explained in Section IV.2. Section IV.3 presents the use of CMA-ES to improve final solutions.

IV.1 Adaptive HBRKGA

In Japa et al. [2022], the authors proposed an HBRKGA, a strategy using an exploitation step to improve BRKGA performance after each generation. The main idea is to explore the neighborhood of individuals in a population changing some individuals. A variation of HBRKGA to optimize hyperparameters is proposed, where multiple methods can be used in the exploitation step. For each method in the exploitation step, a probability of being called is defined, which is used to select which method is called at a given time. These probabilities are updated following the adaptive step presented in Boudia et al. [2007].

The proposed method's pseudo-code is shown in Algorithm 1. The inputs are: disturbance ratio ϵ, the number of exploitation steps, the available exploitation methods β and k, which determines every how many generations the probabilities of the exploitation methods to be chosen will be updated. Lines 1-6 are initialization steps, such as generating the initial population and computing the fitness function for all individuals. The core loop starts at line 7, where each iteration represents one generation of the genetic algorithm. The selection of best individuals, i.e. selection of individuals from the elite set, occurs at line 8. The remainder individuals of the population are generated at lines 9-10 with crossover and mutation operators. The loop starting at line 12 computes the fitness of every individual of the current population by applying the objective function $\mathcal{F}$. Our main contribution happens at lines 15, 16 and 18 where an exploitation method is selected, executed and updated, respectively. Details of these steps are presented in Algorithm 2. Further, the current population is refreshed and at the final, the highest individual found is returned.

This approach involves using these two strategies listed above to improve the results of HBRKGA. Instead of applying the same exploitation step after each generation of BRKGA, is introduced an adaptive probability of choice based on fitness obtained by each exploitation strategy. In the beginning, a uniform distribution is used and, based on the candidate's solutions found, each exploitation

Algorithm 1 Adaptive HBRKGA

Input: ϵ, *steps*, β, k
Output: Best found γ

```
1: Generate initial population P with p randomly generated γ's of n keys each;
2: Iter ← 0;
3: for γ in P do
4:    γ.score ← F(γ̄);
5: end for
6: Sort γ's in P by their score;
7: while stopping criteria not satisfied do
8:    P* ← Select some individuals γ;
9:    P_c ← Crossover individuals from P*;
10:   P_m ← Generate mutants from P*;
11:   P* ← P* ∪ {P_c} ∪ {P_m};
12:   for γ in P* do
13:      γ.score ← F(γ̄);
14:   end for
15:   E_m ← Select exploitation from β;
16:   P* ← exploitation(P*, ε, E_m, F(∗));
17:   if Iter mod k = 0 then
18:      β ← Update probabilities from β;
19:   end if
20:   P ← P*;
21:   Sort γ's in P by their score;
22:   Iter ← Iter + 1;
23: end while
24: return γ with highest score in P;
```

strategy improves their probability to be chosen, as shown in Algorithm 2, which input parameters are: exploitation methods set β, best fitness γ^*, a list of scores obtained using each exploitation method γ and a variational factor θ.

The loop starting at line 2 will compute the best average cost by exploitation method using fitness, score, count and the variational factor. Finally, the line 9 will update each exploitation method probability to be selected after the next generation of BRKGA.

Algorithm 2 Update exploitation method probabilities

Input: $\beta, \gamma^*, \gamma, \theta$

```
1: σ ← 0;
2: for β_i in β do
3:    score(β_i) ← score(β_i) + γ(β_i)
4:    avg(β_i) ← score(β_i)/counter(β_i);
5:    Q(β_i) ← (γ*/avg(β_i))^θ;
6:    σ ← σ + Q(β_i);
7: end for
8: for β_i in β do
9:    P(β_i) ← Q(β_i)/σ;
10: end for
```

IV.2 Exploitation step

The exploitation step is a crucial component in the proposed method as it aims to improve the performance of the algorithm by exploring the neighborhood of current solutions in search of better ones. In this research, the authors have implemented two different exploitation methods: the Random Walk and the Bayesian Walk.

The Random Walk method, as the name suggests, explores the neighborhood of the current population by making random movements. In this method, after receiving the current population γ, a new population $\bar{\gamma}$ is generated by stochastically exploring the neighborhood defined in the interval $(0, \gamma_i(1+\epsilon)]$, where ϵ determines how far from the current population the exploration will take place.

The Bayesian Walk, on the other hand, is based on Bayesian optimization and uses a probabilistic model of a surrogate function, in this case, a Gaussian Process. The main idea behind this method is to use all the information available from previous evaluations to generate the most promising individuals to explore, rather than simply relying on gradients or Hessian approximations to improve its performance [Snoek et al., 2012].

IV.3 CMA-ES

CMA-ES is a black box optimization algorithm that uses a multivariate normal distribution to generate candidate solutions and updates its parameters across generations to sample solutions from regions with the potential to contain good solutions [Nomura et al., 2021].

Genetic algorithms simulate the evolution of a population over generations [Goldberg, 2002]. With the selection of the fittest individuals, naturally, the final population consists of good individuals. The CMA-ES strategy was used after the evolution step in an attempt to enrich the individuals of the last population found by BRKGA. The main idea behind this approach is present a population that evolved across a couple of generations and make a warming start in CMA-ES improving the results obtained previously. In Nomura et al. [2021], a similar approach related to CMA-ES uses prior knowledge to create an enhanced version.

Chapter V Experimental Evaluation

In this chapter, an overview of the validation process for the proposed hyperparameter optimization strategy is provided. Six publicly accessible datasets widely used from various application domains were utilized to conduct the experiments.

V.1 Datasets

A total of six datasets of classification tasks were used in experiments, MNIST [LeCun, 1998], Rectangles [Larochelle et al., 2007], COSMOS [Scoville et al., 2007] and MNIST variants (MNIST-IB, MNIST-RanB, MNIST-RotB) Larochelle et al. [2007]. A short explanation of these datasets is provided below.

1. MNIST: it is a set of handwritten digit image data, having 70,000 examples between the training set and the test set [LeCun, 1998]. The images in the dataset amount to 784 features, each image having a size of 28x28 pixels.

2. Rectangles: it's a dataset with images containing rectangles drawn in them. The objective of this dataset is to discriminate between tall and wide rectangles in images of 28x28 pixels dimensions.

3. COSMOS (Cosmic Evolution Survey): it's a dataset with information about more than 500000 astronomical objects and 90 attributes with their photo-metric measures. The same dataset and pre-processing steps as cited in [Machado et al., 2016] was used.

4. MNIST with image background (MNIST-IB): MNIST variant with background produced with piece of internet images Larochelle et al. [2007].

5. MNIST with random background (MNIST-RanB): MNIST dataset with random background producing noise in the digits Larochelle et al. [2007].

6. MNIST with rotation and background (MNIST-RotB): MNIST-RandB with rotated digits Larochelle et al. [2007].

V.2 Objective function $\mathcal{F}(*)$

In general terms, the objective of a supervised classification problem is to minimize a function J_Θ using a function also called hypothesis, that receives an input (x_i, y_i) and evaluates $h(x_i)$. Using a loss function $C(h(x_i), y_i)$ the classifier's performance is evaluated and parameters Θ can be updated. Cross-entropy was used as loss function, softmax activation function as the output layer and ADAM as optimizer [Kingma and Ba, 2015] for training the neural network.

At every execution of $\mathcal{F}(*)$, the model is trained using the hyperparameters passed as input, and the value of the F-measure is returned as a measure of the model's quality.

It should be noted that a simple architecture of a neural network may not be effective for complex data, but it is sufficient to evaluate the impact of hyperparameters on the model's performance, allowing for the identification of poor and good hyperparameter settings.

V.3 Experimental settings

The experiments ran on a computer with an AMD Ryzen$^{\text{TM}}$ 5 5600X 3.70 GHz processor and 32GB RAM, also equipped with a GeForce RTX 3060 GPU with 12GB of RAM. BRKGA's Python implementation developed in Andrade et al. [2021] was used to develop A-HBRKGA. The *PyTorch* framework [Paszke et al., 2019] was used to build the artificial neural network (ANN) models.

To validate the approach, a multilayer perceptron was employed as the neural network, fully connected with three hidden layers. The hyperparameters selected to be optimized were the number of neurons in each layer, the learning rate, and the regularization rate. The range of values used for the experiments was based on previous studies and is summarized in Table V.1. The parameters used to parametrize BRKGA were also summarized in Table V.2. The stopping criteria was defined as 9 generations, each population had 6 individuals, the number of elites and mutant set was set to 2 and 1 respectively. Bias, the probability of an offspring inheriting a gene from an elite parent, was set to 0.7, exploitation steps are limited to three iterations and the perturbation ratio was set to 0.15, which is used to limit the neighborhood explored at the exploitation step.

The proposed approach, A-HBKRGA, was compared to several baseline optimization strategies, including Grid Search and Random Search implemented from scratch, Bayesian Optimization[1] and CMA-ES[2] which are publicly available. To ensure fairness in the comparison, the total number of evaluations for each method was approximated.

[1] https://github.com/fmfn/BayesianOptimization
[2] https://github.com/CyberAgentAILab/cmaes

Table V.1: Range of values of hyperparameters used by dataset

Dataset	**Neurons in 1st layer**	**Neurons in 2nd layer**	**Neurons in 3rd layer**	**Learning rate**	**Regularization rate**
Cosmos	[5, 15]	[5, 30]	[5, 45]	$[10^{-6}, 10^{-1}]$	$[0, 10^{-3}]$
Rectangles	[1000, 2000]	[2000, 4000]	[2000, 6000]	$[10^{-6}, 10^{-1}]$	$[0, 10^{-3}]$
MNIST and variations	[1000, 2000]	[2000, 4000]	[2000, 6000]	$[10^{-6}, 10^{-1}]$	$[0, 10^{-3}]$

Table V.2: Parameters used in the experiments.

Parameter	**Value**
Max. number of generations (stopping criteria)	9
Population size p	6
Number of elites p_e	2
Number of mutants p_m	1
Bias ρ	0.7
steps	3
Perturbation ratio ϵ	0.15

V.4 Experimental results

In this section, the performance of A-HBRKGA is compared to other optimization techniques through various evaluation metrics. Subsection V.4.1 presents the evaluation metrics used to compare the results, Subsection V.4.2 compares model performance, the score obtained after optimization, in Subsection V.4.3 we explore the time performance, ie., time taken to run each technique. Additionally, an ablation study is presented in Subsection V.4.4 to provide further insight into the results.

V.4.1 Evaluation Metrics

In a classification problem, there are four possible situations to label an instance:

- True positive (TP): classifier gives the positive label correctly;
- False positive (FP): classifier gives the positive label incorrectly;
- True Negative (TN): classifier gives the negative label correctly;
- False Negative (FN): classifier gives the negative label incorrectly.

In this evaluation, the performance of the classifiers is measured using metrics that provide information about the accuracy of the classifiers [Sokolova and Lapalme, 2009]. The metrics used were precision, recall and F-measure.

Precision is calculated as the ratio of correctly classified positive examples to the total number of examples labeled as positive (Eq. V.1).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{V.1}$$

Recall is calculated as the proportion of correctly classified positive examples to the total number of actual positive examples (Eq. V.2).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{V.2}$$

F-measure or F_1-score is a metric that combines precision and recall. Its values range in the closed interval $[0, 1]$ and the closer to 1, the better the result. It was chosen as the main evaluation metric because all datasets we used have multiple classes, and it is not affected by imbalanced classes. The F-measure is calculated for each class present and the mean is used as the final evaluation metric.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{V.3}$$

V.4.2 Model performance comparison

As one can observe from Table V.3, the proposed method, A-HBRKGA, achieved the highest scores in three of datasets tested and in the two remaining the scores are very close. Specifically, A-HBRKGA outperformed other methods in terms of F_1-score, with a stable standard deviation which, proves that the results were consistently better.

In the Cosmos dataset, Bayesian Optimization achieved the highest score with other methods with very near results. This is probably the impact of simplest dataset used in experiments. A-HBRKGA outperformed competitors in Rectangles dataset followed by Bayesian Optimization. In MNIST and MNIST-RotB datasets, A-HBRKGA come behind Bayesian Optimization with a difference of only 0.0035 and 0.0055 respectively. In MNIST-IB and MNIST-RanB datasets, the same behaviour was observed, A-HBRKGA gather highest scores followed by Bayesian Optimization. This is notable, as it highlights the adaptability of A-HBRKGA to different types of datasets, comparing to its non-adaptive counterpart.

The Grid Search method, however, performed poorly with the exception of the Cosmos dataset. This is likely due to the pre-defined search space, which may not be optimal for all datasets.

Summarizing, the results presented in Table V.3 demonstrate the effectiveness of A-HBRKGA as a powerful method for hyperparameter optimization, achieving the highest scores across most datasets tested. It highlights the adaptability of A-HBRKGA and its ability to explore the hyper-

parameter space effectively, thus yielding better results than other methods such as Grid Search, Bayesian Optimization and CMA-ES and HBRKGA.

Table V.3: Results of the experiments ran for every method

Method	Dataset					
	Cosmos	**Rectangles**	**MNIST**	**MNIST-IB**	**MNIST-RanB**	**MNIST-RotB**
	$avg \pm std$	$avg \pm std$	$avg \pm std$	$avg \pm std$	$avg \pm std$	$avg \pm std$
Random Search	0.9853 ± 0.0009	0.9778 ± 0.0070	0.9444 ± 0.0109	0.6178 ± 0.1037	0.6484 ± 0.1121	0.3050 ± 0.0324
Grid Search	0.9845 ± 0.0012	0.7590 ± 0.0176	0.7097 ± 0.0114	0.3852 ± 0.0205	0.3118 ± 0.0235	0.2085 ± 0.0134
BO	$\mathbf{0.9857 \pm 0.0009}$	0.9807 ± 0.0077	$\mathbf{0.9549 \pm 0.0064}$	0.6943 ± 0.0266	0.7061 ± 0.1041	$\mathbf{0.3228 \pm 0.0263}$
CMA-ES	0.9853 ± 0.0014	0.9320 ± 0.0667	0.9018 ± 0.0840	0.4466 ± 0.1516	0.4454 ± 0.2080	0.2378 ± 0.0558
HBRKGA	0.9853 ± 0.0010	0.9682 ± 0.0128	0.9431 ± 0.0108	0.6076 ± 0.1272	0.6782 ± 0.1123	0.2940 ± 0.0406
A-HBRKGA	0.9856 ± 0.0012	$\mathbf{0.9830 \pm 0.0052}$	0.9514 ± 0.0084	$\mathbf{0.7060 \pm 0.0160}$	$\mathbf{0.7334 \pm 0.0425}$	0.3173 ± 0.0406

Legend: For every method, the average F_1 score (avg) and standard deviation (std) are presented. Every experiment has been repeated a total of 30 times.

V.4.3 Time performance comparison

Table V.4 presents the average execution time for 30 runs of each method, with the best (lowest) results highlighted in bold for each dataset. Looking exclusively to time performance, no one method dominates all others. In Cosmos and MNIST datasets, CMA-ES was the fastest method. Grid Search was very efficient in MNIST-RanB dataset. In Rectangles dataset, A-HBRKGA superates the concorrents. Further, in MNIST-IB and MNIST-RotB datasets, HBRKGA obtained the best average time.

Genetic algorithms do not rely on all previous data and can be executed in constant time Lan et al. [2022]. In contrast, Bayesian Optimization requires access to all previous data in order to take informed decisions about the next step in the optimization process. This characteristic makes A-HBRKGA more efficient than Bayesian Optimization for this situation, which explains the difference between Bayesian Optimization and A-HBRKGA.

Regarding the difference between HBRKGA and A-HBRKGA, the explanation lies in the exploitation methods used by each methodology. While HBRKGA relies only on Random Walk for all generations, in A-HBRKGA this is not fixed. The adaptive mechanism of A-HBRKGA allows for the selection of different exploitation methods, such as Bayesian Walk, which can greatly reduce the execution time. The main difference between Random Walk and Bayesian Walk is the number of runs required for each member of the population. In Random Walk, the algorithm runs multiple times, one for each member of the population. On the other hand, in Bayesian Walk, the whole population is fed to the Bayesian Optimizer, which in turn returns a new population.

The implementation of the adaptive mechanism in A-HBRKGA allows for the selection of the most appropriate exploitation method for each generation, based on the current state of the opti-

mization process. This flexibility in the exploitation method can greatly improve the efficiency of the optimization process and explains the difference in time between HBRKGA and A-HBRKGA. The details of how time differs from both exploitation mechanisms will be discussed further in the ablation study of the next subsection.

The results obtained demonstrate the effectiveness of using genetic algorithms as a starting point for exploring hyperparameter optimization. A-HBRKGA, in particular, proves to be a powerful improvement, showing a significant reduction in execution time compared to other methods. These findings highlight the potential of genetic algorithms as a robust and efficient approach for solving complex optimization problems.

Table V.4: Time performance of the experiments ran for every method

Method	Dataset											
	Cosmos		**Rectangles**		**MNIST**		**MNIST-IB**		**MNIST-RanB**		**MNIST-RotB**	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
Random Search	00:03:26	00:00:02	00:18:42	00:00:33	01:48:09	00:02:36	01:03:20	00:05:40	01:17:37	00:08:35	01:00:40	00:02:41
Grid Search	00:03:45	00:00:01	00:16:13	00:00:28	01:28:25	00:03:18	00:52:24	00:01:41	**00:50:30**	00:01:38	00:51:40	00:01:24
Bayesian Optimization	00:03:41	00:00:07	00:16:48	00:02:12	02:20:19	00:18:43	01:03:48	00:06:12	01:24:28	00:13:22	01:03:33	00:04:47
CMA-ES	**00:03:21**	00:00:02	00:12:46	00:00:39	**01:15:39**	00:06:24	00:50:30	00:03:21	01:00:10	00:04:21	00:49:22	00:02:15
HBRKGA	00:04:09	00:00:02	00:13:20	00:00:55	01:25:28	00:07:28	**00:46:31**	00:06:43	00:53:42	00:05:39	**00:45:11**	00:04:38
A-HBRKGA	00:04:25	00:00:09	**00:11:17**	00:01:37	01:17:03	00:11:26	00:48:39	00:07:10	00:55:31	00:07:59	00:46:12	00:06:11

Legend: For every method, we show the average time in format h:m:s (avg) and standard deviation (std). Every experiment has been repeated a total of 30 times.

V.4.4 Ablation study

The purpose of an ablation study is to evaluate the contribution of individual components or features of a system by removing or modifying them and then measuring the impact on performance Reddy [1974]. In this study, we conducted a comparison of A-HBRKGA against variants of the BRKGA method. The entries with a "+" sign (such as BRKGA+BW and BRKGA+RW) indicate that the traditional BRKGA method has been modified to include the specified exploitation method (Bayesian Walk or Random Walk) as a surrogate function. Following the same principle, when a "+" sign is followed by CMA-ES, indicates that it was used as final action to optimize hyperparameters.

Table V.5 presents the results of the ablation study. The results are presented in terms of average F_1 score (avg), standard deviation (std), and average time (time). The results are based on 30 executions using the same parameters as shown in Table V.2. The highest F_1 scores and the lowest times for each dataset are highlighted in bold for easy comparison.

As we can see from Table V.5, our proposed method, A-HBRKGA+CMA-ES, consistently produces the highest F1-score among all the variants of BRKGA, while traditional BRKGA always has the lowest execution time. This is to be expected, as the inclusion of an additional surrogate function in every iteration of BRKGA leads to an increase in execution time.

When comparing the execution time of BRKGA, BRKGA+BW and BRKGA+BW+CMA-ES, we can see that the inclusion of the Bayesian Walk surrogate function and CMA-ES has only a minimal impact on execution time. For all datasets, BRKGA+BW and its enhanced version, BRKGA+BW+CMA-ES, consistently improves F1-score, even though it increases execution time, indicating that it is possible to achieve high-quality solutions by sacrificing a small amount of time.

HBRKGA and HBRKGA+CMA-ES follows a similar pattern, where it outperforms traditional BRKGA in terms of F1-score but has a higher execution time. The difference in time between Random Walk and Bayesian Walk is more pronounced, making the trade-off less favorable.

A-HBRKGA and A-HBRKGA+CMA-ES, on the other hand, are able to effectively balance this trade-off by producing high-quality solutions while maintaining a moderate increase in execution time. Its execution time is never as fast as traditional BRKGA, but the solutions produced are consistently better among all variants.

Our results demonstrate that using genetic algorithms as a starting point for exploring hyper-parameter optimization can be a robust and efficient approach. The ablation study highlights the effectiveness of incorporating different exploitation methods and implementing an adaptive mechanism to select the most appropriate method, as we were able to achieve significant improvements in the F_1-score compared to traditional BRKGA by trading-off a bit of execution time. Our study confirms that exploitation methods can significantly improve the performance of BRKGA and that our adaptive strategy further enhances the effectiveness of HBRKGA (BRKGA+RW) as reported in Japa et al. [2022]. Additionally, the addition of a final optimization step using CMA-ES was beneficial by increasing the average score. This behavior was consistently observed in all BRKGA variants and datasets.

Table V.5: Results of the experiments ran for every method in the ablation study

Method	Dataset											
	Cosmos		Rectangles		MNIST		MNIST-IB		MNIST-RanB		MNIST-RotB	
	avg ± *std*	*time*	*avg* ± *std*	*time*	*avg* ± *std*	*time*	*avg* ± *std*	*time*	*avg* ± *std*	*time*	*avg* ± *std*	*time*
BRKGA	0.9839 ± 0.0012	**00:00:40**	0.8970 ± 0.1043	**00:03:04**	0.8673 ± 0.1361	**00:23:36**	0.2729 ± 0.1463	**00:14:28**	0.3730 ± 0.1924	**00:17:41**	0.1744 ± 0.0451	**00:12:51**
BRKGA + CMA-ES	0.9846 ± 0.0016	00:00:51	0.8958 ± 0.0919	00:03:52	0.9276 ± 0.0288	00:26:14	0.3271 ± 0.1519	00:20:25	0.3400 ± 0.1769	00:20:30	0.2063 ± 0.0651	00:14:55
BRKGA + BW	0.9850 ± 0.0012	00:04:23	0.9823 ± 0.0095	00:11:32	0.9514 ± 0.0086	01:12:08	0.6980 ± 0.0114	00:47:02	0.7453 ± 0.0107	00:54:39	0.3250 ± 0.0349	00:45:50
BRKGA + BW + CMA-ES	0.9854 ± 0.0011	00:04:59	0.9847 ± 0.0052	00:12:50	0.9534 ± 0.0058	01:17:15	0.6998 ± 0.0123	00:55:26	0.7479 ± 0.0139	01:01:51	0.3287 ± 0.0127	00:51:22
HBRKGA	0.9853 ± 0.0010	00:04:09	0.9682 ± 0.0128	00:13:20	0.9431 ± 0.0108	01:25:28	0.6076 ± 0.1272	00:46:31	0.6782 ± 0.1123	00:53:42	0.2940 ± 0.0406	00:45:11
HBRKGA + CMA-ES	0.9848 ± 0.0012	00:04:44	0.9673 ± 0.0155	00:15:02	0.9489 ± 0.0071	01:33:51	0.6489 ± 0.0752	00:53:34	0.6743 ± 0.1059	01:03:08	0.3091 ± 0.0327	00:49:42
A-HBRKGA	0.9854 ± 0.0012	00:04:25	0.9830 ± 0.0052	00:11:17	0.9514 ± 0.0084	01:17:03	0.7060 ± 0.0160	00:48:39	0.7334 ± 0.0425	00:55:31	0.3173 ± 0.0406	00:46:12
A-HBRKGA + CMA-ES	**0.9856** ± 0.0010	00:04:57	**0.9853** ± 0.043	00:12:56	**0.9573** ± 0.0064	01:20:06	**0.7102** ± 0.0110	00:56:25	**0.7561 ± 0.0160**	01:00:52	**0.3317 ± 0.0139**	00:50:58

Legend: For every method, we show the average F_1 score (avg), F_1 score's standard deviation (std) and average time in format h:m:s (time). Every experiment has been repeated a total of 30 times.

To verify that results are statistically significantly, Wilcoxon non-parametric test [Rey and Neuhäuser, 2011] was used to determine if there is a significant difference between paired observations from the two groups analyzed in each comparision. The significance level α was set to 0.05. The techniques presented in Table V.3 and Table V.5 were compared against the results of A-HBRKGA and A-HBRKGA+CMA-ES and resulting p-values are shown in Table V.6 and Table

V.7 respectively. The cases where no statistically significance difference between distributions was observed are highlighted in boldface.

In Table V.6 and Cosmos, the simplest dataset used in the experiments, there is insufficient evidence to reject the null hypothesis in 7 of 10 comparing techniques. Looking to other datasets, the methods that use Bayesian Optimization achieve near scores. In MNIST-RanB, MNIST-RotB p-values are very close to significance level α that minor parameter modification in input may take A-HBRKGA achieve better results and consequently change the outcome of the Wilcoxon test.

Table V.6: p-values resulting from applying the Wilcoxon test to compare techniques to A-HBRKGA

Dataset	**RS**	**GS**	**BO**	**CMA-ES**	**BRKGA**
Cosmos	**0.2429**	0.0026	**0.3492**	**0.7000**	0
Rectangles	0.0021	0	**0.1687**	0	0
MNIST	0.0026	0	**0.1473**	0.0001	0
MNIST-IB	0	0	0.0175	0	0
MNIST-RanB	0.0001	0	**0.2367**	0	0
MNIST-RotB	**0.0523**	0	**0.9193**	0	0

Dataset	**BRKGA + CMA-ES**	**BRKGA + BW**	**BRKGA + BW + CMA-ES**	**HBRKGA**	**HBRKGA + CMA-ES**
Cosmos	0.0384	**0.1460**	**0.7611**	**0.3818**	**0.0577**
Rectangles	0	**0.7231**	**0.2382**	0	0
MNIST	0	**1**	**0.3184**	0.0066	**0.2054**
MNIST-IB	0	0.0221	**0.1579**	0	0.0001
MNIST-RanB	0	**0.0879**	**0.0879**	0.0040	0.0013
MNIST-RotB	0	**0.1094**	**0.3492**	0.0035	**0.2621**

In Table V.7, the interpretation of results of A-HBRKGA+CMA-ES is very similar to the one done previously to A-HBRKGA. In MNIST and MNIST-IB datasets, the p-values obtained from comparison of BRKGA+BW and BRKGA+BW+CMA-ES lower than α suggests powerful evidence to reject the null hypothesis and come to the conclusion that there is a significant difference between the groups. This is an interesting result, in addition to the lower time required to execute, A-HBRKGA+CMA-ES achieved better results compared to Bayesian Optimization. Compared to the remaining techniques, the p-values results confirm that A-HBRKGA and A-HBRKGA+CMA-ES achieved competitive hyperparameter optimizations.

Table V.7: p-values resulting from applying the Wilcoxon test to compare techniques to A-HBRKGA+CMA-ES

Dataset	RS	GS	BO	CMA-ES	BRKGA	BRKGA + CMA-ES
Cosmos	**0.2621**	0.0009	**0.4771**	**0.7303**	0	0.0087
Rectangles	0.0002	0	0.0036	0	0	0
MNIST	0	0	**0.2449**	0	0	0
MNIST-IB	0	0	0.0047	0	0	0
MNIST-RanB	0	0	0	0	0	0
MNIST-RotB	0.0003	0	**0.1731**	0	0	0

Dataset	BRKGA + BW	BRKGA + BW + CMA-ES	HBRKGA	HBRKGA + CMA-ES	A-HBRKGA
Cosmos	**0.0577**	**0.5028**	**0.4161**	0.0262	**0.6554**
Rectangles	**0.1225**	0.9468	0	0	**0.0695**
MNIST	0.0026	0.0201	0	0.0002	0.0173
MNIST-IB	0.0002	0.0043	0	0	0.2231
MNIST-RanB	**0.2129**	**0.2894**	0	0	0.0327
MNIST-RotB	**0.8236**	**0.2206**	0	0.001	**0.1460**

Chapter VI Conclusion and Future Work

In this research A-HBRKGA, an improved version of the HBRKGA algorithm was proposed by incorporating an adaptive exploitation component after each evolutionary step, the adoption of an additional optimization step with CMA-ES was explored too. Experiments were conducted to evaluate the performance of this method on classification problems using artificial neural networks and the results were compared with those of traditional optimization approaches such as Grid Search, Random Search, Bayesian Optimization, and CMA-ES. The focus of the study was to develop solutions that optimize hyperparameters while balancing performance and execution time. The results showed that the proposed method was effective in reducing the time spent while achieving better average scores, making it a viable strategy for optimizing hyperparameters and improving the performance of artificial neural networks.

Possible future work includes testing other exploitation techniques, such as using alternative surrogate functions for Bayesian Optimization and other acquisition functions. As the addition of CMA-ES as an additional optimization step had a positive impact, other techniques can also be explored. The use of multiple populations and refinement of the best individuals can also be considered.

References

Aggarwal, C. C. et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

Andrade, C. E., Toso, R. F., Gonçalves, J. F., and Resende, M. G. The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *European Journal of Operational Research*, 289(1):17–30, 2021.

Atteia, G., Alhussan, A. A., and Samee, N. A. Bo-allcnn: Bayesian-based optimized cnn for acute lymphoblastic leukemia detection in microscopic blood smear images. *Sensors*, 22, 2022a.

Atteia, G., Samee, N. A., El-Kenawy, E. S. M., and Ibrahim, A. Cnn-hyperparameter optimization for diabetic maculopathy diagnosis in optical coherence tomography and fundus retinography. *Mathematics*, 10, 2022b.

Barbero, Á., Lázaro, J., and Dorronsoro, J. *Finding Optimal Model Parameters by Discrete Grid Search*, volume 44, pages 120–127. Springer Berlin Heidelberg, 2007.

Bean, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6:154–160, 1994.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

Bernstein, L., Sludds, A., Hamerly, R., Sze, V., Emer, J., and Englund, D. Freely scalable and reconfigurable optical hardware for deep learning. *Scientific Reports*, 11, 2021.

Boudia, M., Louly, M. A., and Prins, C. A reactive grasp and path relinking for a combined production-distribution problem. *Computers and Operations Research*, 34:3402–3419, 2007.

Brabazon, A., O'Neill, M., and McGarraghy, S. *Natural computing algorithms*, volume 554. Springer, 2015.

Brochu, E., Cora, V., and Freitas, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

Darwin, C. and Knight, D. *The Evolution Debate, 1813-1870: On the Origin of Species*. Evolution debate, 1813-1870. Routledge, 2003.

Debie, E. and Shafi, K. Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses. *Pattern Analysis and Applications*, 22(2):519–536, 2019.

Farag, H. H., Said, L. A., Rizk, M. R., and Ahmed, M. A. E. Hyperparameters optimization for resnet and xception in the purpose of diagnosing covid-19. *Journal of Intelligent and Fuzzy Systems*, 41:3555–3571, 2021.

Florea, A. C., Andonie, R., Florea, A.-C., and Andonie, R. Weighted random search for hyperparameter optimization, 2019.

Frank Hutter, Lars Kotthoff, J. V. *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer International Publishing, 1st ed. edition, 2019.

Goldberg, D. E. *The design of innovation: Lessons from and for competent genetic algorithms*, volume 1. Springer, 2002.

Gonçalves, J. F. and Resende, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17:487–525, 2011.

Gorgolis, N., Hatzilygeroudis, I., Istenes, Z., and Gyenne, L.-G. Hyperparameter optimization of lstm network models through genetic algorithm. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–4. IEEE, 2019.

Hansen, N. The CMA evolution strategy: A comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

Hutter, F., Kotthoff, L., and Vanschoren, J. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

Japa, L., Serqueira, M., Mendonça, I., Bezerra, E., Aritsugi, M., and González, P. H. A conjugated evolutionary algorithm for hyperparameter optimization. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9, 2022.

Khan, N. A parallel implementation of the covariance matrix adaptation evolution strategy. *arXiv preprint arXiv:1805.11201*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, pages 1–13, 2015.

Lan, G., Tomczak, J. M., Roijers, D. M., and Eiben, A. Time efficiency in optimization with a bayesian-evolutionary algorithm. *Swarm and Evolutionary Computation*, 69:100970, 2022.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lee, S., Kim, J., Kang, H., Kang, D.-Y., and Park, J. Genetic algorithm based deep learning neural network structure and hyperparameter optimization. *Applied Sciences*, 11(2):744, 2021.

Liashchynskyi, P. and Liashchynskyi, P. Grid search, random search, genetic algorithm: a big comparison for nas. *arXiv preprint arXiv:1912.06059*, 2019.

Loshchilov, I. A computationally efficient limited memory cma-es for large scale optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 397–404, 2014.

Loshchilov, I. and Hutter, F. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

Machado, E., Serqueira, M., Ogasawara, E., Ogando, R., Maia, M. A., da Costa, L. N., Campisano, R., Guedes, G. P., and Bezerra, E. Exploring machine learning methods for the star/galaxy separation problem. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 123–130. IEEE, 2016.

Mitchell, T. M. and Mitchell, T. M. *Machine learning*, volume 1. McGraw-hill New York, 1997.

Mohan, B. and Badra, J. A novel automated superlearner using a genetic algorithm-based hyperparameter optimization. *Advances in Engineering Software*, 175, 2023.

Nomura, M., Watanabe, S., Akimoto, Y., Ozaki, Y., and Onishi, M. Warm starting cma-es for hyperparameter optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9188–9196, 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani,

A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Reddy, R., editor. *Speech Recognition: Invited Papers Presented at the 1974 IEEE Symposium*. Academic Press, 1974.

Rey, D. and Neuhäuser, M. Wilcoxon-signed-rank test. In *International encyclopedia of statistical science*, pages 1658–1659. Springer, 2011.

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*, chapter 19. Pearson, 4th edition, 2021.

Scoville, N., Aussel, H., Brusa, M., Capak, P., Carollo, C. M., Elvis, M., Giavalisco, M., Guzzo, L., Hasinger, G., Impey, C., et al. The cosmic evolution survey (cosmos): overview. *The Astrophysical Journal Supplement Series*, 172(1):1–8, 2007.

Sentewolf. Concept of directional optimization in cma-es. `https://commons.wikimedia.org/wiki/File:Concept_of_directional_optimization_in_CMA-ES_algorithm.png`, 2023.

Shahin, I., Nassif, A. B., and Alsabek, M. B. Covid-19 electrocardiograms classification using cnn models. volume 2021-December, pages 448–452. Institute of Electrical and Electronics Engineers Inc., 2021.

Sloss, A. N. and Gustafson, S. 2019 evolutionary algorithms review. *Genetic programming Theory and practice XVII*, pages 307–344, 2020.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Sokolova, M. and Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yang, L. and Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *CoRR*, abs/2007.15745, 2020.

Şahin, N., Alpaslan, N., and Hanbay, D. Robust optimization of segnet hyperparameters for skin lesion segmentation. *Multimedia Tools and Applications*, 81:36031–36051, 2022.