Technical University of Munich
TUM School of Engineering and Design

Automated Machine Learning for Applications in Earth Observation

Kalifou René Traoré

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr.-Ing. Yuanyuan Wang

Prüfende der Dissertation:

1. Prof. Dr.-Ing. habil. Xiaoxiang Zhu
2. Prof. Francisco Chicano, Ph.D.

Die Dissertation wurde am 20.11.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 07.08.2024 angenommen.

This page is intentionally left blank.

Acknowledgements

I would like to sincerely thank my supervisor, Xiao Xiang Zhu, for her tremendous support during my PhD. She has always been generous with her time and expertise, as well as trusting and patient throughout my entire PhD. Moreover, I could not have undertaken this journey without the help of Andrés Camero, who has been a fantastic collaborator and mentor in all aspects of my PhD. Working with him has been a formidable human experience.

I am also grateful for the kindness and support of my working group (Deep Learning WG), that has been supportive of my professional growth along the way, by providing me feedback but also joy at work. My thanks should also go to the management of the SiPEO chair of TUM, as well as our department at DLR for enabling a working environment filled with a great spirit of collaboration, which has provided me with plenty of inspiration for my research. I am also thankful for the camaraderie of all my colleagues and office-mates, for contributing to making DLR such a lovely place to work.

Last but not least, I would also like to thank my family, especially my parents, Salif and Regina, my brother, Camille, and friends for their unwavering support in my studies and life, particularly when dealing with the challenging COVID-19 pandemic.

This page is intentionally left blank.

Abstract

Remote Sensing (RS) is a very active field of research, with a high pace of innovation, and developments in numerous areas including the emergence of novel sensors, and the launch of programs for the acquisition of ever-growing volumes of data, to name a few. From the monitoring of environmental changes to the monitoring of urban growth, or the detection of crises and disasters, the available data has the potential to support highly impactful decision-making. In order to make this decision-making efficient and reliable, the research area of Automated Machine Learning (AutoML) provides tools for autonomously designing well-suited Data Science pipelines. The works presented in this dissertation explore the aspects of *explainability*, *efficiency* and *accessibility* of AutoML for tailored decision-making in Earth Observation-based (EO) applications. The objectives and contributions of this dissertation can be summarized as follows:

- *Explainability:* AutoML systems generally consist of an ensemble of components all playing an essential role in the quality of the resulting decision-making solutions. We are interested in understanding the influence of all components in the output of AutoML systems in RS applications. In particular, we propose to analyze Neural Architecture Search (NAS) problems using a bag of features characterizing key optimization aspects explaining search algorithm performances. We provide novel insights on the influence of such elements on search algorithm behavior, and how these could be used to better design or calibrate AutoML search algorithms in EO.

- *Efficiency:* Designing modern decision-making solutions is still mostly performed manually, which can require considerable time investments and provide results of variable quality. In this dissertation, we present methodologies for making such activity time-efficient, and resulting in resource-efficient well-performing solutions for EO applications. To do so, we have developed a methodology helping specific AutoML search algorithms (population-based metaheuristics) to achieve faster convergence and better performances (quality of identified decision-making solutions) on EO applications. This is done by providing a data-driven initialization strategy that uses freely available search space performance evaluations. Besides, we have also introduced a methodology able to find high-performing image classifiers of low complexity, competing with popular and manually-designed baselines of similar complexity.

- *Accessibility:* Given the high computational cost of conducting AutoML experiments, researching the topic with data-intensive domain-specific EO applications is not always an easy endeavour. To contribute to making AutoML more accessible to the EO community, we explore the creation of a NAS Benchmark in EO, helping develop and benchmark AutoML search algorithms without the need for model evaluation on an EO scene classification task.

Zusammenfassung

Fernerkundung ist ein sehr aktives Forschungsgebiet mit einem hohen Innovationstempo und Entwicklungen in zahlreichen Bereichen, darunter das Aufkommen neuartiger Sensoren und die Einführung von Programmen zur Erfassung ständig wachsender Datenmengen, um nur einige zu nennen. Die verfügbaren Daten haben das Potenzial, äußerst wirkungsvolle Entscheidungen zu unterstützen. Diese reichen von der Überwachung von Umweltveränderungen und städtischen Wachstum bis zu der Erkennung von Krisen und Katastrophen. Um diese Entscheidungsfindung effizient und zuverlässig zu gestalten, stellt der Forschungsbereich Automatisiertes Maschinelles Lernen (AutoML) Werkzeuge zur autonomen Gestaltung geeigneter Data-Science-Pipelines bereit. Die in dieser Doktorarbeit vorgestellten Arbeiten untersuchen die Aspekte der *Erklärbarkeit, Effizienz und Zugänglichkeit* von AutoML für maßgeschneiderte Entscheidungen der Anwendung von Erdbeobachtung. Die Ziele und Beiträge dieser Dissertation lassen sich wie folgt zusammenfassen:

- *Erklärbarkeit*: AutoML-Systeme bestehen im Allgemeinen aus einem Ensemble von Komponenten, die alle eine wesentliche Rolle für die Qualität der resultierenden Entscheidungslösungen spielen. Wir sind daran interessiert, den Einfluss aller Komponenten auf die Ausgabe von AutoML-Systemen in FK-Anwendungen zu verstehen. Insbesondere schlagen wir vor, Probleme der Neural Architecture Search (NAS) mithilfe einer Reihe von Merkmalen zu analysieren, die wichtige Optimierungsaspekte charakterisieren und die Leistung von Suchalgorithmen erklären. Wir liefern neue Erkenntnisse über den Einfluss solcher Elemente auf das Verhalten von Suchalgorithmen und wie diese genutzt werden könnten, um AutoML-Suchalgorithmen für die Erbeobachtung besser zu entwerfen oder zu kalibrieren.

- *Effizienz*: Der Entwurf moderner Entscheidungslösungen wird immer noch größtenteils manuell durchgeführt, was einen erheblichen Zeitaufwand erfordert und Ergebnisse unterschiedlicher Qualität liefern kann. In dieser Dissertation stellen wir Methoden vor, um solche Aktivitäten zeiteffizient zu gestalten und zu ressourceneffizienten, leistungsstarken Lösungen für Erdbeobachtungsanwendungen zu führen. Zu diesem Zweck haben wir eine Methodik entwickelt, die bestimmten AutoML-Suchalgorithmen (populationsbasierte Metaheuristik) hilft, schnellere Konvergenz und bessere Leistungen (Qualität der identifizierten Entscheidungslösungen) bei Erdbeobachtungsanwendungen zu erreichen. Dies geschieht durch die Bereitstellung einer datengesteuerten Initialisierungsstrategie, die frei verfügbare Leistungsbewertungen des Suchraums verwendet. Darüber hinaus haben wir eine Methodik eingeführt, mit der leistungsstarke Bildklassifikatoren mit geringer Komplexität gefunden werden können, die mit beliebten und manuell entworfenen Basislinien ähnlicher Komplexität konkurrieren.

- *Zugänglichkeit*: Angesichts des hohen Rechenaufwands für die Durchführung von AutoML-Experimenten ist die Erforschung des Themas mit datenintensiven domänenspezifischen Erdbeobachtungsanwendungen nicht immer ein einfaches Unterfangen. Um dazu beizutragen, AutoML für die Erdbeobachtungsgemeinschaft zugänglicher zu machen, untersuchen wir die Erstellung eines

NAS-Benchmarks für die Erbeobachtung, der bei der Entwicklung und dem Benchmarking von AutoML-Suchalgorithmen hilft, ohne dass eine Modellbewertung für eine Erdbeobachtungszenenklassifizierungsaufgabe erforderlich ist.

This page is intentionally left blank.

Table of Contents

This page is intentionally left blank.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Symbols

Abbreviation	*Description*
AutoML	Automated Machine Learning
AE	Aging Evolution
BILS	Best Improvement Local Search
CNN	Convolutional Neural Network
CV	Computer Vision
EA	Evolutionary Algorithm
EO	Earth Observation
FDC	Fitness Distance Correlation
FLA	Fitness Landscape Analysis
GA	Genetic Algorithm
GNN	Graph Neural Network
GO	Global Optima
HP	Hyperparameter
HPO	Hyperparameter Optimization
LCZ	Local Climate Zone
LHS	Latin Hypercube Sampling
LO	Local Optima
LON	Local Optima Network
LULC	Land Use Land Cover
MMCE	Mean Misclassification Error
NAS	Neural Architecture Search
NLP	Natural Language Processing
NN	Neural Network
RS	Remote Sensing
ResNet	Residual Neural Network
SOTA	State of the art
STN	Search Trajectory Network

Table 1: Table of abbreviations used in the manuscript.

This page is intentionally left blank.

Chapter 1

Introduction

This section provides an introduction to this doctoral dissertation by stating the motivations behind the work being conducted, stating the research problems being addressed, introducing the contributions that have been made, and providing a general outline of the document.

1.1 Motivations and Challenges

With an ever-growing number of satellite programs being launched, the amount of publicly available data is on the rise in the field of RS. This has led to a high demand for technological solutions that can utilise these resources to generate novel insights and support decision-making. Examples of real-world applications that benefit from such technologies are precision agriculture and urban development monitoring, which policymakers rely on for making accurate decisions by complementing data gathered on the ground with remote-sensing data. In practice, many such technologies use modern Computer Vision-based (CV) models to tackle their tasks. An example of this is the use of Convolutional Neural Networks (CNNs). In order to improve the decision-making abilities of such models, the research communities of CV and EO have increasingly worked on developing novel configurations and architectures for the models. However, these innovations have traditionally resulted from manual engineering, a process that requires important time investments and is not guaranteed to succeed. On the other hand, researchers in AutoML have developed methodologies to help automate and optimize the design of data-driven decision-making models. This dissertation aims to investigate AutoML methodologies for helping design solutions that are more efficient and better tailored to solve EO-related applications.

1.2 Problem Statements

Within the scope of this doctoral dissertation, we aim to find answers to the following research questions:

- *RQ1:* Can we expect AutoML search algorithms to perform similarly in the EO domain, as they do in the CV domain?

- *RQ2:* Assuming that there exist similarities across domains of applications, can we leverage domain-specific algorithmic knowledge from a non-EO application, to help an AutoML search algorithm perform better in its EO counterpart?
- *RQ3:* By looking at the characteristics of an AutoML optimization problem (e.g. search space), is it possible to anticipate the behaviour and performances of a search algorithm needed to solve it? Assuming the previous question has a positive answer, what elements would have such an influence on the search algorithm behaviour?

1.3 Contributions

The contributions of this manuscript are summarized as follows:

- *Contribution C1:* Optimizing neural network architectures for a target application is a complex task that requires careful consideration of numerous elements, some critical to the suitable functioning of the resulting solutions. These include identifying the type of architecture to consider, the optimizer to tune the architectures, and the evaluation protocol to measure their fitness. We introduce the framework of the Fitness Landscape Footprint[1] enabling us to quantify the impact of such elements on the landscape of a NAS search problem. We demonstrate its ability to characterize and compare the difficulty of finding good architectures for image classification tasks of two very different domains (CV and EO). This work is introduced in Section 4.1 and associated with the publication P1 (see in section 1.5).
- *Contribution C2:* As a Follow-up work to Contribution 1, we demonstrate the use of the framework for a very practical problem: identifying the most beneficial input-sensor setting when searching for an optimal model in the case of a multi-modal RS image classification task[2]. We quantify the similarities and anticipated benefits of each input for the considered NAS problem. This work is introduced in Section 4.2.
- *Contribution C3:* Following up on the previous studies, we investigate how the theory of fitness landscape analysis can help characterize the influence of performance evaluation metrics on the task of tuning the hyperparameter of decision-making models[3]. More particularly, we discover that a malfunction of the performance metric can make AutoML landscapes harder for optimizers, on a variety of Hyperparameter Optimization (HPO) scenarios and tasks. This work is introduced in Section 4.3 and associated with the publication P4.
- *Contribution C4:* We propose a data-driving initialization technique for population-based search algorithms[4]. We demonstrate the benefit of this method against popular initialization baselines for three established search algorithms. We demonstrate the possibility of initializing a search algorithm deployed on the problem of tuning models for an EO target classification task while using performance data from a different source domain (classification of natural scenes). This work is introduced in Section 5.1 and associated with the publication P2.

- *Contribution C5:* We propose a novel approach for searching efficient and low-complexity scene classification models[5], combining a modular backbone architecture with a complexity-reducing loss. This work is introduced in Section 5.2 and associated with the publication P3.
- *Contribution C6:* We propose a novel NAS database for the EO, providing both real and surrogate performance estimations over a large search space of image classifiers. This work is introduced in Section 6.

1.4 Outline

This manuscript is structured in seven chapters, of which the remaining ones are organized as follows: in Chapter 2, we describe general concepts that are important to understand the state of the art in Data Science for EO, NAS, as well as landscape analysis for AutoML. Then, in Chapter 3, we present relevant works in Machine Learning for EO, AutoML, as well as Landscape Analysis methodologies that are highly relevant in describing combinatorial AutoML optimization problems. Chapter 4 presents our research output on the topic of landscape-aware AutoML. Relevant contributions include a framework for describing generic NAS problems (see Section 4.1), as well as its application to a multi-modal sensor-fusion NAS instance (see Section 4.2). In Chapter 5, we introduce works on the topic of efficient AutoML, with contributions to the subject of the initialization of population-based NAS algorithms (see Section 5.1), as well as to the subject of searching for efficient decision-making models of low-complexity (see Section 5.2). In Chapter 6, we present our contributions to AutoML databases in EO. In Chapter 7, we provide a conclusion to the manuscript by presenting future research plans on the topic of landscape-aware neural architecture search for EO and providing additional discussions on the topics previously tackled. Moreover, in Appendix A, we provide a description of the datasets used in all experiments supporting the manuscript. Last but not least, we provide a list of publications supporting the manuscript, in the Appendix B, followed by the publications themselves.

1.5 Publications

This section presents the publications related to the work introduced in this dissertation.

Publications in Journals

- P1: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang, *Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems*, *Under review* at Journal of Machine Learning Research (JMLR), 2023.
- P2: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2023), *A Data-driven Approach to Neural Architecture Search Initialization*, Annals of Mathematics and Artificial Intelligence, pp. 1-28. Springer Nature, doi: 10.1007/s10472-022-09823-0, ISSN 1012-2443.

Publications in the Proceedings of Conferences

- P3: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2021), *Compact Neural Architecture Search for Local Climate Zones Classification*, In: The 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 06. - 08. Oct. 2021, Online. doi: 10.14428/esann/2021.ES2021-55, ISBN 978287587082-7.

- P4: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2023), *We Won't Get Fooled Again: When Performance Metric Malfunction Affects the Landscape of Hyperparameter Optimization Problems*, In: 6th International Conference on Optimization and Learning, OLA 2023, 1824, pp. 148-160. Springer, Cham, 3-5 May 2023, Malaga, Spain, doi: 10.1007/978-3-031-34020-8_11. ISBN 978-303134019-2, ISSN 1865-0929.

Other publications not included in the cumulative dissertation

- P5: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2022) *Landscape of Neural Architecture Search across sensors: how much do they differ?*, In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 3-2022, pp. 217-224, XXIV ISPRS Congress, 6.-11. June 2022, Nice, France, doi: 10.5194/isprs-annals-V-3-2022-217-2022, ISSN 2194-9042.

- P6: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2021), *Lessons from Clustering of a Search Space: a Data-driven initialization technique to Search*, Workshop on Data Science meets Optimization, International Joint Conferences on Artificial Intelligence (IJCAI), Online.

- P7: Demir, Emre and Traoré, Kalifou René and Camero, Andrés (2024), *Leveraging performance-based metadata for designing multi-objective NAS strategies for efficient models in Earth Observation*, In: The 32nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 09. - 11. Oct. 2024, Brugges, Belgium, doi: 10.14428/esann/2024.ES2024-94, ISBN 978-2-87587-090-2.

This page is intentionally left blank.

Chapter 2

Fundamentals

In this chapter, we introduce the fundamental concepts behind the works presented in this manuscript. These concepts cover the research areas of Data Science for EO, NAS, and Evolutionary Computation (FLA).

2.1 Modern Deep Learning Methodologies for Earth Observation

Deep learning-based methodologies have helped tackle plenty of applications in RS. Examples of use cases span from Precision Agriculture with Crop Monitoring to Disaster Management with systems for Flood Detection. An RS application requiring data-driven decision-making can be classified into various categories of tasks, requiring task-specific methodologies and data. The following paragraphs introduce the fundamental tasks of Semantic Segmentation, Object Detection, as well as Change Detection, to give examples of methods used to tackle popular relevant RS applications.

Semantic Segmentation Semantic segmentation is an important task in RS. Given an image of objects sensed on the ground, it consists of assigning a semantic label to every pixel of the image, a.k.a. pixel-wise classification[6]. As a result, all instances of objects present in a scene usually have associated pixels assigned the same semantic label. Early methodologies for RS semantic segmentation have been highly inspired by works from the Compute Vision community, where various solutions use a Deep Learning-based feature extractor to tackle the task. A particularly recognizable feature of such methods is their use of a neural network architecture that has two main feature processing stages. In the first stage, low-dimensional features are extracted from an input image using a series of (convolutional) layers of decreasing filter size. Then, it is followed by a symmetric structure aiming at constructing an image-like semantic map from the low-dimensional features. This is done using a series of (convolutional) layers of increasing kernel sizes, a.k.a. deconvolution. This two-step structure is present in various architectures such as the Fully Convolutional Network[7] (FCN), SegNet[8], as well as U-Net[9], with variations in the type of layers being used (fully connected etc.), and other layer characteristics (up-sampling etc.). Recent works in RS have focused on tackling domain-specific challenges present in the RS data such as the lack of data annotations to train the models, the importance of pixel-level accuracy, as well as dealing with images from specific sensors (multi-spectral, hyper-spectral), or combination of (sensor fusion).

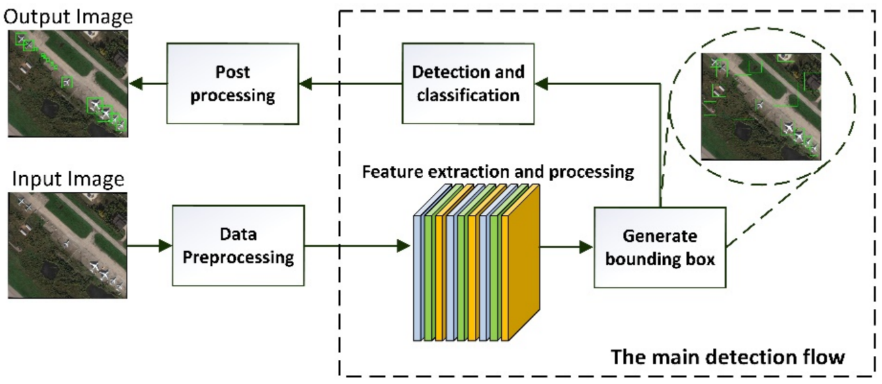


Figure 2.1: Overview of the Framework of Object Detection for Remote Sensing[10].

Object Dectection Moreover, some RS data analysis scenarios may require categorizing remotely sensed objects and locating them geographically. This is the task of Object Detection, which supports real-world applications such as building planning or urban monitoring, where these object locations and characterization play an important role and are crucial[10]. The challenges to Object Detection methodologies lie in the characteristics of the objects to detect (coexistence of objects, size, ambiguity in the category), the complexity of the background of the images being used, and the quality of the labelling of the data. The standard framework of RS Object Detection methodologies is outlined in Figure 2.1. The first step consists of preparing the input data for feature extraction: most techniques consist of data augmentation, as well as clipping, i.e., the slicing of large images into smaller ones in order to reduce the cost of feature extraction. Then, the processed data is passed to a feature-extracting model, which is needed to categorize and locate the objects. Figure 2.2 displays statistics regarding the usage of feature extractors, with the seminal VGGNet and popular ResNet architectures being used 67% of the time. Next is the step of bounding box generation performed using a dedicated algorithm, followed by the steps of object classification and a final prepossessing.

Change Detection RS data analysis also involves dealing with data of various forms, including sequential data, that requires tailored models. For this purpose, RS experts have developed a variety of decision-making models able to deal with time series, with an example of a use case being Change Detection. In RS, Change Detection[11] aims at accurately detecting changes occurring on a remotely sensed object located at the surface of the Earth, using a sequence of measurements taken at various moments in time. This framework supports the monitoring of all types of relevant events for which the effect can be remotely sensed, such as natural disasters (flood detection, etc.) or environmental changes (deforestation), to name a few. Figure 2.3 provides the visualization of an example of change detection[11], where the aim is to detect changes in land use using consecutive optical images (pairs) of the areas of interest. The baselines that are shown are based on data fusion of the elements in the sequence and use an encoder-decoder type of architecture with

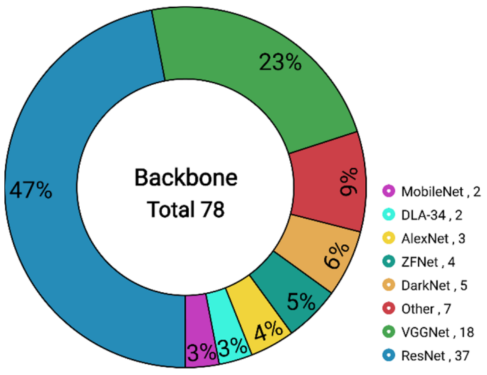


Figure 2.2: Statistics about the distribution of usage in of modern image classification backbones in Object Detection-oriented mythologies in Remote Sensing[10].

data fusion occurring at various stages of the models. The objective is to detect the pixels (and associated geographical locations) where the changes occur in the sequence elements.

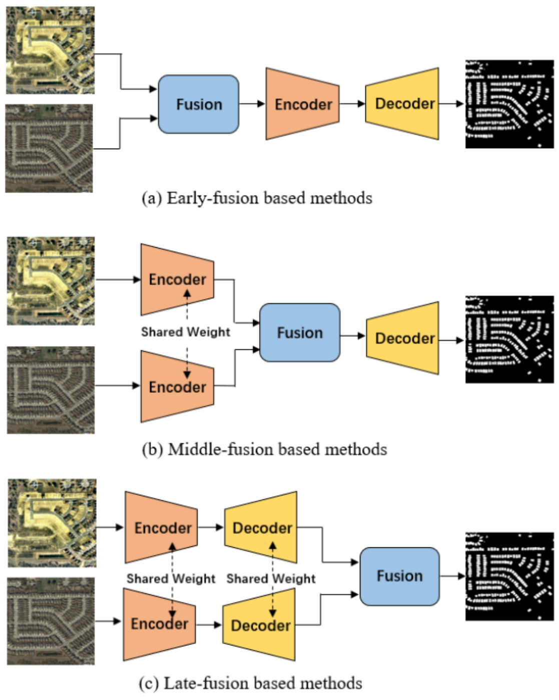


Figure 2.3: Example of change detection baseline using pairs of images from scenes where changes occur from one image to the other[11]. The method used is based on manually designed Deep Learning-based baselines of varying architectures: an auto-encoder with early, middle or late-fusion.

To summarize, the span of EO applications that have benefited from the Deep learning revolution is wide and continuously growing. Most EO applications for which large amounts of data can be gathered to simulate real-world use cases, could be supported by modern, high-performing, but data-hungry decision-making models. However, the design of models that are specific to EO applications is often done manually, an activity that requires a considerable time investment, with no certainty in the optimality of the resulting solutions. This constitutes an opportunity for AutoML and NAS to support EO experts in identifying more reliably suitable solutions.

2.2 Automated Machine Learning

AutoML as a research field, aims to support data scientists in designing robust general machine learning pipelines in a systematic and efficient manner[12]. It is subdivided into three main research areas that are HPO[13], NAS[14] and Meta Learning[15]. Since most modern ML algorithms have many hyperparameters, HPO-related methods focus on providing HP configurations, improving performances over manual tuning while dealing with complex HP spaces and costly fitness estimations by means of Multi-fidelity approximations[13]. On the other hand, NAS encompasses all methods explicitly dealing with the automated design of NN-based decision-making algorithms and overlaps with the field of HPO[14]. Other AutoML research enables algorithms to learn from data about past deployment, i.e. meta-data, such as model evaluations (fitness, training time, etc.), to help deploy optimizers at reduced computational cost[15].

Besides, another important aspect of AutoML research is providing systems working in the real world[12]. An example is auto-sklearn[16], an AutoML system able to autonomously handle the tuning of the hyperparameters of numerous decision-making models, feature and data preprocessing methods. Formally, auto-sklearn uses the framework of Combined Algorithm Selection and Hyperparameter optimization (CASH)[17] in order to jointly find the appropriate decision-making algorithm and a hyperparameter setting yielding optimal performances for the task at hand (e.g. data preprocessing). It has proven its value in real-world scenarios by winning several AutoML competitions against SOTA competing systems. Besides, other works provide an overview of existing open-source frameworks for building ML pipelines automatically and provide a detailed benchmark of their performance[18]. In particular, their analysis considers ML systems from the perspective of the pipeline creation problem, analogous to CASH, which describes systems with variable pipeline structures (i.e. an internal data and feature processing graph of variable complexity).

2.3 Neural Architecture Search

NAS refers to the topic of research dealing with the autonomous design of Neural Network-based (NN) decision-making models. It is a sub-field of AutoML and is closely tied to HPO, with a specific focus on NN architecture-related hyperparameter tuning.

The general framework encompassing most NAS methods is depicted in Figure 2.4. It consists of three main components: a search space, a search strategy (a.k.a. controller), and an evaluation strategy. The search space defines NN configurations that are of interest. The search strategy, a.k.a. controller, aims to find solutions that maximize performances on unseen data. The performance evaluation strategy provides a measurement of fitness for solutions sampled by the search strategy. In NAS problems that can be described with this framework, the search strategy iteratively samples a candidate solution (or set of) from the search space, and then this candidate has its fitness estimated by the evaluation strategy. The fitness measurement is then returned it to the search strategy in order for it to make an informed decision on which candidate to sample next. This iteration is ended once the budget of the search iteration is reached, and the configuration (or set of) having reached the best fitness is returned as the optimal architecture.

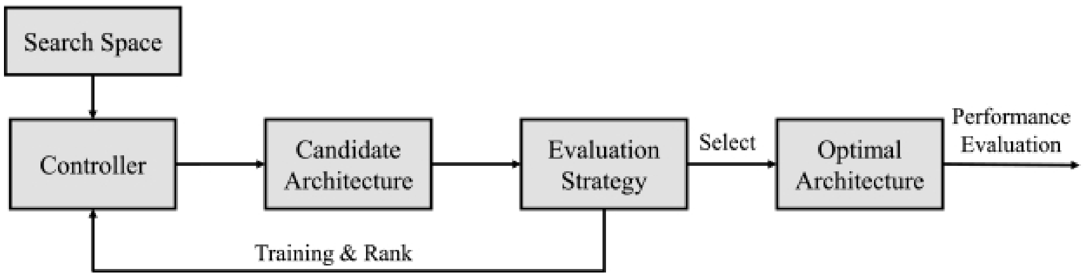


Figure 2.4: Overview of the general framework of NAS, showing the interaction between the components in a NAS pipeline[19].

Search Strategies: Search strategies are tools used to explore search spaces and retrieve competitive solutions, given budgets of computation. Most search strategies can be categorized as black-box optimization algorithms, at the exception of the more recent one-shot NAS strategies. Early NAS methodologies have relied on black-box methods that are based on Evolutionary or Genetic algorithms. More recent methods make use of Reinforcement Learning or Bayesian optimization-based search algorithms. When performance is the only considered metric, search strategies can optimize for a single objective. They can also consider multiple objectives in scenarios where other costs, such as model complexity-based metrics, are of interest. Figure 2.5 illustrates an example of a search strategy, using the framework of Reinforcement Learning[20].

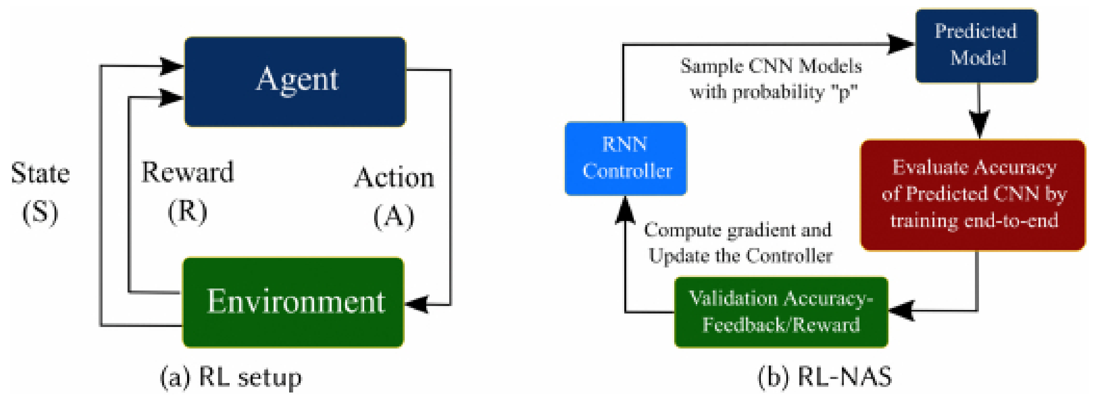

Agent
State (S)
Reward (R)
Action (A)
Environment
(a) RL setup
Sample CNN Models with probability "p"
Predicted Model
RNN Controller
Evaluate Accuracy of Predicted CNN by training end-to-end
Compute gradient and Update the Controller
Validation Accuracy-Feedback/Reward
(b) RL-NAS

Figure 2.5: Overview of a prototypical Reinforcement Learning-based NAS search strategy (right, RL-NAS), based on the standard RL computational framework (left)[20].

Search Spaces: Search spaces are containers for all the solutions that are considered in given NAS problems. They play a crucial role in the NAS framework since the performance of search strategies will, by default, be bound to their definition.

In practice, search spaces are parameterized by a set of HPs that aim to affect the structure of NN candidate solutions, and in turn, their performance. When comparing modern NN architecture configurations with similarities, the level at which their differences occur highlights the nature of search spaces that could express them. For instance, NN architectures that follow the design principle of the popular ResNet architecture and only differ in the number of stages or blocks contained could be generated by *macro search spaces* or *chain-structured search spaces*. Both methodologies help express configurations with variations of macro-level features, with *macro search spaces* allowing for more complex solutions. On the other hand, NN configurations of identical macro-level features, but differences in design choices (operations, variable connections between layers, etc.) that are affecting elementary building blocks, can be expressed by *cell-based search spaces* or even *hierarchical search spaces*. Figure 2.6 provides a visualization of an example of search space that is *cell-based*[19]. Each image classification architecture sampled from it has a fixed chain-like structure to its backbone, that alternates between two types of feature processing modules. The first module is a sequence of *variable*, but repeating units called normal cells. The second module contains a single unit, the reduction cells. The normal cell does the most computationally expensive operations, while the reduction cell aims at extracting higher-level features through dimension reduction.

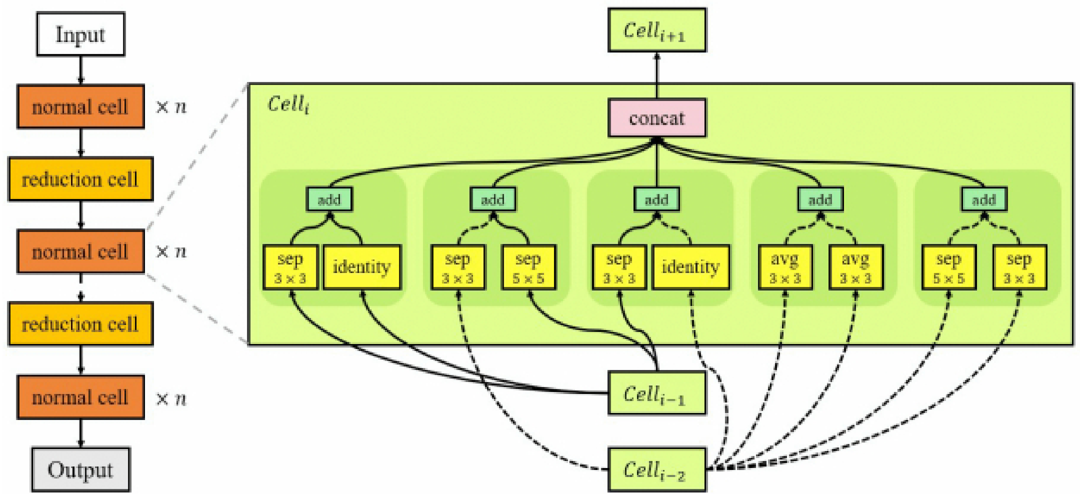


Figure 2.6: Overview of a Neural Network configuration instance, generated from a chain-structured image classification backbone, where the variable hyperparameters are associated to the *normal cell*[19]. This setting falls under the umbrella of *cell-based* search spaces.

Evaluation Strategies Performance evaluation strategies play the role of assessing the quality of solutions sampled by the NAS search strategy from the search space [19,20,14,12]. This is done by direct measurement of fitness using performance metrics while training the models, or indirect measurements with metrics that are approximations of model performance. Other variants of performance estimation strategies might rely on fitness measurements, while reducing the fidelity of some resources, such as reduced training budgets or even training data, as an approximation of the fitness for full fidelity (resources completely available). Figure 2.7 illustrates an example[19] of such, particularly suitable for highly constrained NAS systems where computational efficiency is essential. Indeed, it is based on learning curve prediction that aims at approximating the performances of a model at full training budget fidelity (black dashed line), after only a few iterations of training (back line). Thus, when used for estimating the performance of numerous solutions, it can reduce the entire training computation need by a considerable margin. Learning curve prediction models usually require to be trained using the complete performance evaluation of a training and evaluation set of solutions (red lines).

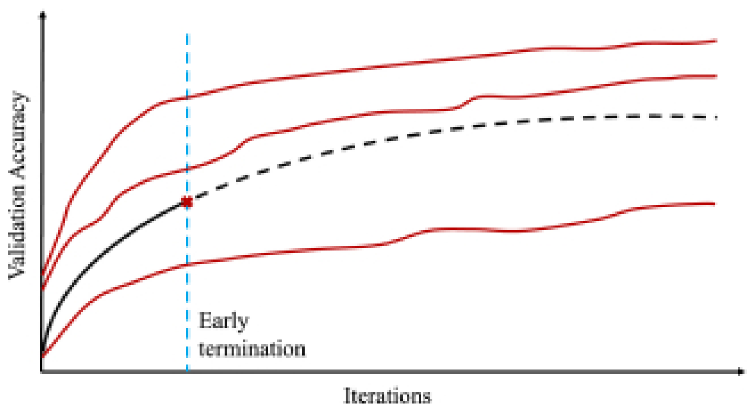


Figure 2.7: Example of performance prediction with an instance of learning curve extrapolation[19]. More precisely, this example shows how a learning curve extrapolation method could select and rank a model for a given budget of training (full fidelity), after only few iterations of training.

2.4 Fitness Landscape Analysis

Fitness Landscape Analysis (FLA) is an area of research at the intersection of various fields dealing with Optimization[21]. Researchers in Evolutionary Computation invented it to help them better understand the Optimization problem they were tackling. Recent FLA-related methodologies aim at providing a quantitative appraisal of search difficulty through the use of various sampling strategies and metrics. The resulting features aim to characterize various aspects of the search processes (fitness, ruggedness, evolvability, epistasis) that may correlate with the behaviour and outcome of the solvers being used to tackle the problem at hand. The following paragraphs introduce a few popular features, helping describe search difficulty in generic optimization problems. These features also apply to applied optimization problems, such as NAS problems, with dedicated search spaces and solvers.

Fitness Distance Correlation The Fitness Distance Correlation (FDC) is often considered an established way to assess the difficulty of optimization problems. It measures the correlation between the fitness of solutions in the search space and their distance to the *global optimum*. Besides, it is challenging to interpret. Instead of considering its numerical quantity, we propose to visualize it, and we formulate in Equation 2.1 as :

$$FDC(x^*, \Omega, f) = \{(d(x^*, y), f(y)), \forall y \in \Omega\} \tag{2.1}$$

where $d(\cdot)$ is a distance operator, and $x^* \in \Omega$ the global optimum.

Ruggedness Another way to analyze an optimization problem's difficulty is by identifying how irregular the associated landscape is. Indeed, a high level of irregularities could affect local search-based algorithms in reaching global optima. Examples of effective indicators of irregularity in a landscape are the *ruggedness* and *multi-modality*. We focus here on a method to assess the ruggedness.

The ruggedness is a *local descriptor* of optimization landscapes: it does not require a knowledge of all the whole search space, but can instead be estimated using *local sampling strategies*. In practice, an appropriate sampling strategy consists of deploying and evaluating several random walks over the search space, always considering randomly selected and distinct starting points. In this way, one can obtain global appreciation of the ruggedness by aggregating its value at the local level in various regions on the landscape. Then, the next is to compute the serial correlation $\rho(\cdot)$ of the fitness of the walks, as introduced in Equation 2.2:

$$\rho(k) = \frac{\mathrm{E}[(f(x_i) - \bar{f})(f(x_{i+k}) - \bar{f})]}{Var(f(x_i))}, \forall i \in \mid \mathrm{W} \mid \tag{2.2}$$

where for a given random walk $\mathrm{W} \in \Omega$, x_i and x_{i+k} are elements of the walk at an interval of k steps (lags), f is the functions evaluating the fitness of these elements, and $\rho(k)$ is the serial correlation considering k lags of interval. Then, the ruggedness is obtained as the inverse of the serial correlation at $k = 1$ lags $\tau = \frac{1}{\rho(1)}$: it is high when the fitness at a local level is highly variable (low $\rho(k = 1)$), and low in the case of a relatively continuous local fitness (large $\rho(k = 1)$).

Local Optima Another relevant descriptor is the estimation of the number of local optima (LO) existing in the search landscape. We consider local maxima for NAS scenarios, given that we use an objective that we maximize (classification accuracy). We also consider problems that fall into the category of combinatorial optimization. Here, we describe a series of steps to approximate the number of existing LO in an optimization landscape. First, we can retrieve a local optimum at the individual level using a local search-based solution. In the case of local maxima, we can use the procedure of *Best Improvement Local Search* (BILS), described in Algorithm 1.

Algorithm 1: Best-improvement local-search (BILS)

```
Choose an initial solution $x \in \Omega$;
repeat
    $x^* = x$;
    for $i = 1$ to $\mid N(x^*) \mid$ do
        Choose $y_i \in N(x^*)$;
        if $f(y_i) < f(x)$ then
            $x = y_i$;
        end
    end
until $x = x^*$;
```

In order to provide an enumeration of all LO in the landscape, one would ideally run the `BILS(·)` several times until no new solutions are reached. However, such a procedure would possibly be exceptionally computationally and time-costly since the duration of execution of `BILS(·)` algorithm (stochastic algorithm) and the number of LO are both variable[22]. The least costly but reliable procedure would be based on the solution to the *Birthday problem*[23].

Algorithm 2: Analytics of the birthday problem

```
Let $T$ the total number of trials of enumeration;
for $i = 1$ to $T$ do
    Choose $M$ distinct random starting points in $\Omega$;
    Iteratively collect the $M$ Local Optima using ;
    Let $k_i$ the number of Optima at first duplication;
end
Let $k_{mean}$ the average number of Optima at duplication;
Derive $N$ the number of Optima using $k_{mean}$ and Eq. 2.3;
```

The estimation is based on large-scale deployment `BILS(·)`, as described in Algorithm 2. First is defined T, the number of trials used in the estimation. Then, for each trial i, we randomly select M distinct starting points in the search space Ω. Then, using the sample, the `BILS(·)` procedure is used to collect M LO. While the retrieval of LO is being performed in an *iterative* manner, we identify k_i, the number of optima for which the first duplicate occurs. Once the T trials are performed, we compute the quantity k_{mean}, the average number of optima that can be collected until a first duplication occurs. Last but not least, we use k_{mean} together with Equation 2.3, in order to estimate the total number of LO N in the search space Ω:

$$N \approx \frac{k_{mean}^2}{-2 * ln(1 - P_D)} \tag{2.3}$$

where N approximates the number of LO given an average number k_{mean} of optima retrieved until duplication, and a fixed probability of duplication P_D in the search space.

Chapter 3

Related work

This chapter presents an overview of the literature relevant to this dissertation, with works regarding the topic of Deep Learning for decision-making in EO, NAS, as well as the general understanding of search algorithms.

3.1 Deep Learning in Earth Observation

Recent advances in RS missions have enabled the collection of large amounts of data that help support various applications, where aerial imaging can contribute to more reliable and systematic decision-making[24]. Examples of this include data in the context of global land-cover mapping[25], the monitoring of environmental changes using change detection[26], or even precision agriculture with the monitoring of crops overtime[27]. This evolution in the increasing availability of curated EO datasets is well captured in Figure 3.1.

In order to take advantage of the opportunities brought by the newly available data, researchers in the RS community have developed plenty of decision-making models in order to cover tasks such as LULC classification, the semantic segmentation of Land-covers, object detection in Edge-detection scenarios, or time-series analysis in change detection-related applications. Besides, the particularities of these EO data sets (e.g., multimodal, geolocated, time-variable data, massive) have also provided an opportunity to develop more domain-specific models, a challenge for solutions highly inspired by the Computer Vision (CV) community.

3.2 Neural Architecture Search

The topic of NAS has been investigated since early 90s[28,29], first under the name of Neuro-evolution, popularized with methods such as NEAT[30] already showing prowess evolving architectures to adapt to complex tasks such as vision-based decision-making for video games[31]. More recently, other evolutionary methods have reestablished the SOTA in CV-based decision-making, with methods such as the Aging Evolution[32], helping find the very competitive families of image classifiers AmoebaNets.

Simultaneously, many competing NAS methods have emerged[14], with search strategies classified as part of very different families of optimization algorithms. For instance, authors have used Reinforcement Learning-based approaches[33] to au-

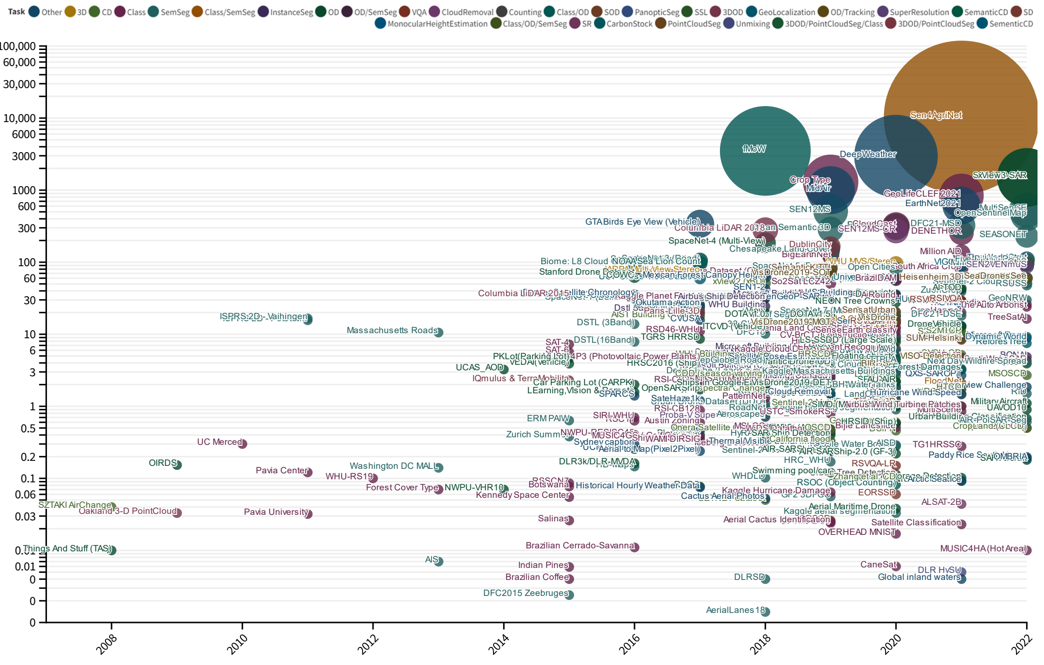


Figure 3.1: Visualization of the datasets published in the field of RS, by time of publication, volume, and application being tackled[24]. It appears that the pace of publication and their volume are considerably increasing.

tonomously design image classification architectures. Other authors have developed Bayesian-based search strategies[34] striving for sample efficiency, helping reduce the computational cost of NAS. Alternatively, continuous Bi-level optimization-based methods have also shown tremendous success in NAS[19], with methods such as DARTS[35] and GDAS[36], helping find SOTA image classifiers while requiring small budgets of computation. They have been the topic of extensive studies, such as works exploring their instability and proposing fixes for more robustness[37].

As an attempt to look beyond the single lens of performance, NAS researchers have also explored methods for helping design more efficient decision-making models, in particular by taking into account hardware capabilities and constraints, a.k.a. Hardware-aware NAS[20]. Such methodologies focus on elements susceptible to improving the efficiency of the models w.r.t. to the hardware, such as the design of better search spaces, search strategies, optimizing the numerical precision of the models, as well as the considering model evaluation metrics that are hardware or complexity-related, as additional objective functions.

Besides, recent contributions to the field of NAS have provided efficient search strategies and search spaces. However, the diversity in implementations and the cost of deployments remain significant barriers when it comes to the adequate reproducibility of the methods[38]. NAS Benchmarks[39] aim at alleviating this problem by providing databases with free-of-cost evaluations of NN configurations, enabling anyone to benchmark and prototype any NAS method. These benchmarks come in the form of databases with exhaustive NN, evaluations or a mix of real[40] and predicted evaluations[41] (surrogate NASBenchmarks). They usually cover a wide range of applications[42] (classification, segmentation, object detection, etc.), and domains[43,44] (CV, Speech, NLP, etc). Besides, more recent benchmarks investigate model efficiency, such as Hardware-aware NAS Benchmarks, or even hybrid settings, such as joint HPO and NAS[45]. Figure 3.2 shows an overview of such benchmark, with the particular case of NAS-Bench-Suite[42].

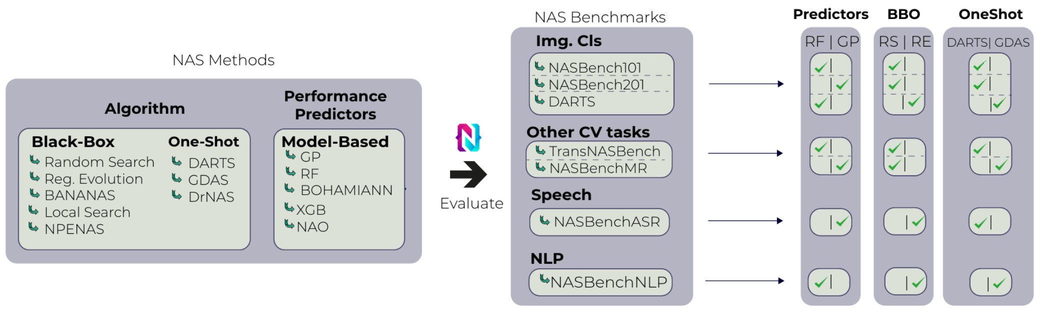


Figure 3.2: Overview of NAS-Bench-Suite, a collection of benchmarks helping develop and evaluate numerous search algorithms, on various search spaces, and domains (Speech, Natural Language Processing, Computer vision)[42].

Most modern CV models have been designed following design patterns that involve a sequential repetition of modular units to build the backbones of CV models. The benchmarking of *Deep* models has shown the importance of *depth* in increasing their capacity to fit datasets, resulting in better decision-making. However, instead of following such heuristics, some authors have proposed an innovative NAS approach based on randomly wiring the elementary units within the desired models. They demonstrate competitive results against SOTA baseline, designed using more conventional principles[46]. Other works have later explored similar ideas, emphasizing the design of search spaces for better model properties. Instead of providing tunable neural architecture backbones, a recent study proposes formulating design spaces. Design space would provide specifications for generating search spaces with desirable properties such as computational efficiency and high performance. The approach is called designing design spaces[47]. Similarly, another study proposes a search for models through scaling pre-existing CV models and empirically identifying a formula considering several factors such as the model depth, width, and resolution. The proposed approach achieves a target efficiency level by enabling high performance and low complexity[48]. According to these recent findings, various ways of dealing with model architecture design have emerged, all being very promising in terms of future outlook.

The quest for a better understanding of the factors that influence model performances is also supported by researchers interested in qualitative analysis of existing AutoML systems. For instance, ArchExplorer was introduced to enable users to visually analyze NAS search spaces to get insights into design principles that influence architecture performance and complexity. ArchExplorer is based on clustering the search space that uses structural similarity (edit distance) to define local and global relationships between the solutions[49]. Similarly, the authors introduce PipelineProfiler, a tool for visually inspecting end-to-end AutoML systems. It enables us to identify how primitives used in the systems affect their performances, simply debugging or even comparing several systems when deployed on Data Science tasks of any kind[50]. Similar studies have followed in the particular context of NAS for vision-based decision-making[51], helping establish best practices for benchmarking, comparing and designing NAS methodologies with an increased awareness of the role played by each component in their systems.

3.3 Understanding the Behavior of Search Algorithms

Aside from contributing to the development of search algorithms and their application to the real world, the field of Optimization has also investigated the more fundamental problem of understanding already existing algorithms and how their inner components contribute to the successes or failures of the whole. This research activity, referred to as Fitness Landscape Analysis (FLA)[21], has been applied to various optimization problems in order to help better design search algorithms. The general idea behind this novel paradigm is to study and model the structure of the landscape of optimization problems and identify how the underlying structures explain and relate to algorithm performances.

For instance, authors have studied the relationship between funnels present in the landscape of quadratic assignments' optimization problems and the performance of search heuristics[52]. They identify critical elements in the structure of funnels (depth) that might affect the difficulty in solving the optimization problems. Other examples appear in the case of dynamic optimization problems, where authors have performed the first analysis of dynamic vehicle routing problems (DVRPs) using static fitness landscape analysis and identify how the temporal or dynamic aspect of the studied problems is captured (or not) using the tools[53]. Similarly, another study has tackled the instances of the dynamic capacitor arc routing problem (DCARPs) to identify how dynamic hyperparameters contribute to the problem difficulty[54].

Furthermore, authors have recently introduced the concept of Search Trajectory Networks (STNs) aiming at modelling the deployment process of search heuristics using a graph-based approach. Their method is compatible with population-based or single-point search heuristics in discrete or continuous search settings. STNs are generated using data collected during several independent runs of algorithms. Thus, STNs can reveal the search space's topology in terms of LO's connectedness and how different regions are navigated (or shared) by a group of competing algorithms[55]. An example of STN is visualized in Figure 3.3.

Alternatively, it is also possible to model fitness landscapes using the concept of Local Optima Networks (LONs)[56,57], highlighting the distribution of LO and their connectedness in the search space. Examples of applications of LONs to analyze optimization problems include the study of the Assisted Seismic History Matching problem[58], where a preliminary study of the problems using LONs helps prepare design an accurate strategy to solve this computationally expensive industrial problem.

Moreover, a novel framework is introduced for analyzing the fitness landscapes of (continuous) optimization problems. They propose a bag of features to gather information about the local aspects of the landscapes, as well as their evolvability, and have shown to be valuable predictors of algorithm run-time (ERT) on classical optimization benchmarks (BBOB).[59]. Another seminal work demonstrates how global and local landscape features, in the context of MO optimization problems, can help reliably quantify problem hardness and predict algorithm performance[60].

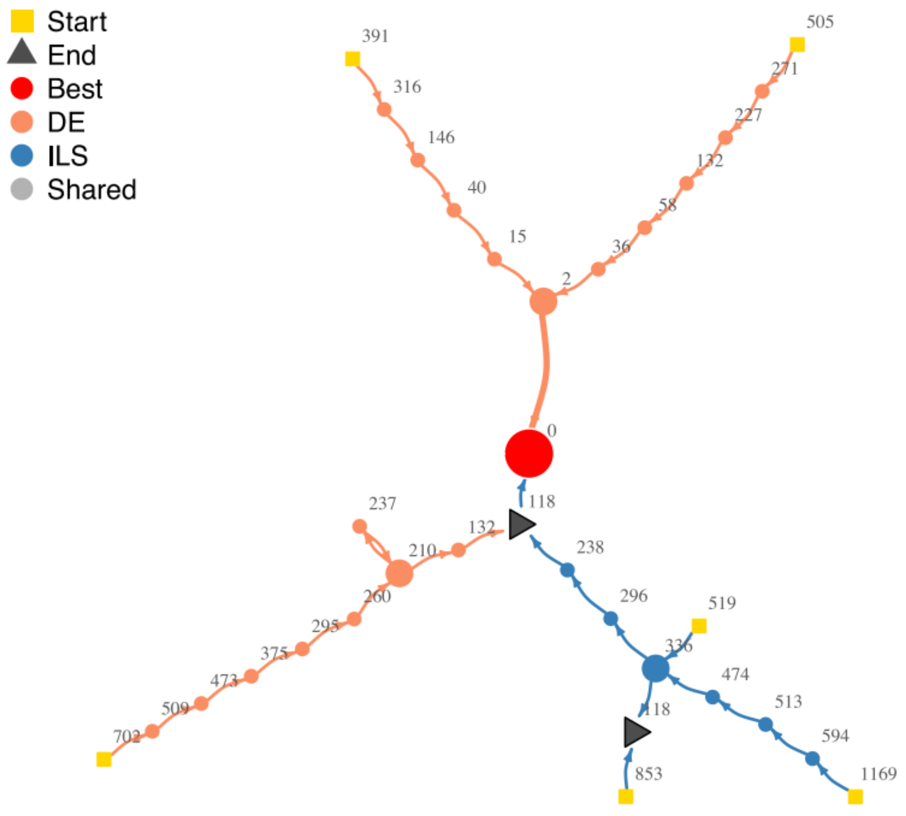


Figure 3.3: Example of Search Trajectory Network generated for two search baselines, the Iterated Local Search (ILS) algorithm, and Differential Evolution (DE)[55].

More recently, authors have started researching how tools from FLA could help better understand AutoML[61] and NAS search problems[62,63]. Part of the work conducted during this thesis carries such research objectives, in particular, search problem understanding in the context of NAS for EO-related decision-making tasks.

This page is intentionally left blank.

Chapter 4

Landscape-aware Search for Automated Machine Learning

This chapter is dedicated to the methodological contributions of this thesis aiming at better characterizing and designing efficient AutoML search algorithms. It introduces the contributions of *C1, C2* and *C3*, while tackling all research questions.

4.1 Toward Landscape-aware Neural Architecture Search

The following section introduces a framework for quantitatively describing the difficulty of generic NAS search problems.

Peer-reviewed publication:: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang, Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems, Under Review, Journal of Machine Learning Research, 2023.

Research questions tackled: *RQ1, RQ2, RQ3.*

Related contributions: *C1.*

4.1.1 Motivations

With the growing availability of AutoML solutions helping Data Scientists better design decision-making pipelines, deciding what search methodology to use, or even hyperparameters to tune among all possible options, becomes harder. Indeed, given the complexity of modern pipelines in their number of components, how does one ensure that the resulting system remains optimal for a given task? This work aims to investigate ways to better understand the influence of the components of an AutoML pipeline, on the behaviour and performances of a search algorithm. Indeed, it aims to do so by highlighting key characteristics in AutoML problems and relating such characteristics to algorithmic performance. The objective is to design an approach that is problem-agnostic, i.e. independent of the AutoML scenario, and task-agnostic, i.e. independent of the decision-making task being tackled (dataset). This should enable us to study, in a systematic manner, any AutoML problems on any scenario and applications in EO. Besides, the approach should also enable us to quantitatively relate the characteristics of a given AutoML problem to alternative scenarios or tasks. Recent advancements in Evolutionary Computation, AutoML and optimization have been of high relevance, in particular work at their intersection, i.e. landscape analysis of AutoML problems, aiming to relate search problem hardness and search performances. We propose a framework based on recent advances in the fields described in Chapter 3.

4.1.2 A Framework of General-purpose Features for NAS Problems

This section introduces the proposed framework for characterizing generic NAS problems. First, we introduce the feature of *Persistence*, helping characterize the temporal dynamic in the performances of solutions in a search space. Then, we present the proposed concept as a *bag of features* tackling several aspects of NAS combinatorial optimization.

Persistence: In order to capture aspects specific to NAS, we introduce metrics aiming at capturing the dynamics in the fitness of a population of solutions in the search space. We consider the situation where each solution $x \in \Omega$, is a neural network trained and evaluated for a given budget of iteration (or epochs) t_n, with additional fitness measurements collected at intermediate time steps in the form of a list $\mathrm{M} = \{f(t_0), f(t_i), ..., f(t_n)\}$. Then, we consider a ranking operator `Ranking(`N`,` t_i`, Ω)`, which returns the set of solutions in the search space Ω that rank N, when considering the evaluation time step t_i, and the fitness operator f. Then, we formulate the metric of *Persistence* Π, outlined in Equation 4.1 as:

$$\Pi(Ranking(\cdot), N) = \frac{P(\cap_{t_i \in \mathrm{R_{all}}} Ranking(N, t_i))}{P(Ranking(N, t_0))} \tag{4.1}$$

where the numerator estimates the probability P that solutions in Ω keep their rank N across all evaluation time steps $t_i \in R_{all}$. Given that the numerator estimates the probability that solutions have the rank N at the initial time step t_0, the *Persistence* measures the chances for solutions initially at rank N, of keeping their rank throughout the training and evaluation procedure. Then, we formulate the *positive* and *negative Persistence*, outlined in Eq.4.2:

$$\begin{aligned} \Pi_{Positive}(N) &= \Pi(TopRank(\cdot), N) \\ \Pi_{Negative}(N) &= \Pi(BottomRank(\cdot), N) \end{aligned} \tag{4.2}$$

where the $TopRank(\cdot)$ (resp. $BottomRank(\cdot)$) is the ranking function retrieving the *Top* (resp. *Bottom*) N *percentile* performers in Ω. Then, the positive (resp. negative) Persistence measures the chances for solutions in Ω to remain *Top-N* (resp. *Bottom-N*) performers. Then, we also consider the area under the curve of the *Persistence* $AuC(\Pi(\cdot, N))$, outline in Eq.4.3 as:

$$AuC(\Pi(\cdot, N)) = \int_1^{N_{max}} \Pi(\cdot, k)\, dk \tag{4.3}$$

Fitness Landscape Footprint: a bag of features enabling a comparative analysis Furthermore, we introduce a novel framework, *fitness landscape footprint*. It gathers the optimization landscape descriptors introduced in Section 2.4, as well as the *Persistence*, in the form of a bag of features. It aims to help characterize a NAS optimization problem by considering the various aspects of its landscape, captured by the selected features. It is best visualized in the form of a radar plot, with an axis for each feature. Aside from describing individual NAS problems, it also enables the comparison of several instances. This is possible by a simple overall pair-wise comparison of the *Footprints* of the different instances. The bag of features contains the following metrics introduced previously:

- *Overall Fitness*: an estimation of the fitness potential of the search space Ω, by measuring the mean and variance in the fitness f.
- *Ruggedness*: a first descriptor of the irregularity of the landscape associated with the given NAS problem.
- *Enumeration of Local Optima*: an additional descriptor of the multi-modality of the landscape.
- *Persistence*: a feature informing on the temporal consistency in the fitness of solutions in Ω. Four quantities are associated with this metric: the positive and negative Persistence, as well as the associated areas under the curve.

4.1.3 A Landscape Analysis of NAS in Various Domains

Next, we describe FDC results, as shown in Figure 4.1, for the classification instance CIFAR-10. The bottom left and right-hand corner figures show the FDC, respectively, when using training budgets of 36, and 108 epochs.

We find a relatively low negative correlation between randomly picked solutions in the search space and the global optimum (GO). More precisely, after 36 epochs of training, the gain in fitness per hamming distance to the optima is positive and about 1.66 percentage point in fitness.

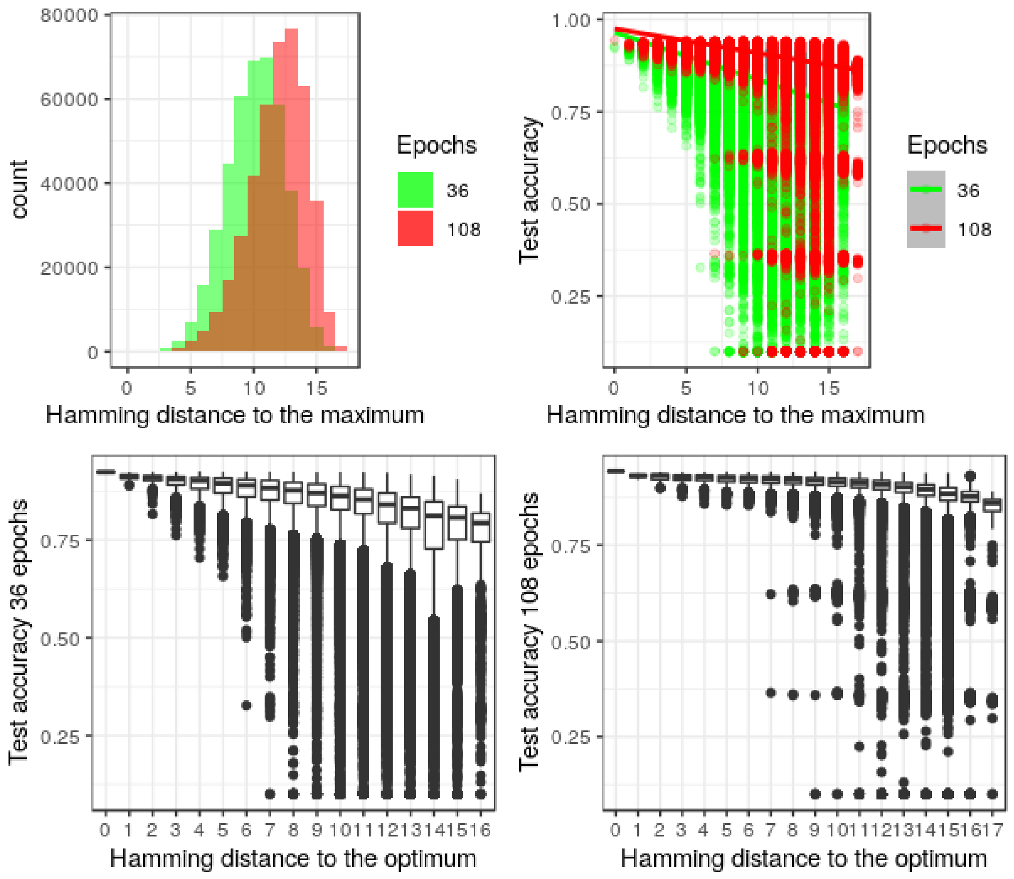


Figure 4.1: Fitness distance correlation for the classification instance CIFAR-10, using the search space of NASBench-101.

After 108 epochs of training, we find that the previously observed negative correlation is slightly decreased: the gain in fitness per reduced hamming distance is lower. Moreover, the overall fitness centers at much more significant values (∼91%).

The existence of a negative correlation for both training regimes suggests the existence of favourable trajectories for local search-based strategies in finding the GO, from randomly picked solutions in the search space. The decrease in the correlation and increase in the average performance of solutions suggests that when training for 108 epochs, the absolute gain in fitness obtained by such an optimizer is decreased for the current search space.

Next, we analyze the landscape from the perspective of the ruggedness. Figure 4.2 provides the visualization of the evaluation of a random walk using CIFAR-10 (green), and the So2Sat LCZ-42 (blue), as input data. Overall, we find that

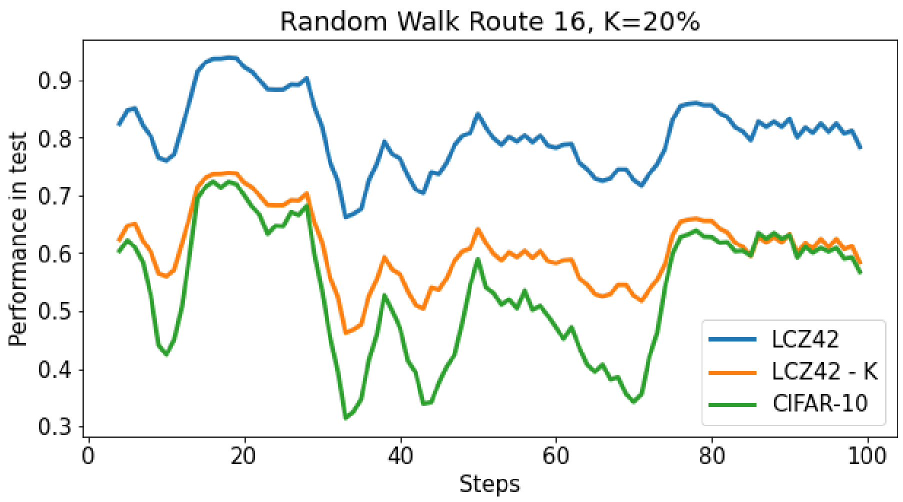


Figure 4.2: Visualization of a random walk route being evaluated on both classification instances CIFAR-10 (green) and So2Sat LCZ42 (blue). The fitness of the solutions in the route is measured on the test set after 36 epochs of training.

the route is of relatively large ruggedness, as the fitness of consecutive solutions appears to fluctuate a lot, for both instances. Surprisingly, we observe a similar visual appearance of the routes for both instances, with what appears to be a constant absolute difference K in fitness between the two routes (blue for LCZ-42, and green for CIFAR-10). We identify $K = 20\%$, such that when removed from the fitness of solutions in the route obtained for the instance LCZ-42, we can retrieve a route close to the one of CIFAR-10.

To summarize, even though the route evaluated using the So2Sat LCZ-42 reaches a larger average fitness, the route obtained using a sensor from another domain (LCZ-42) provides similar curvature, with absolute distance in fitness relatively constant between both. This suggests that there are areas of the search space that provide with the same ruggedness, for landscapes generated using instances from different domains.

Trial	*Avg. Step*	*Avg. Improvement(%)*	*First Repeat k*	*Cardinal*
1	2.90	4.53	94	6373
2	2.89	5.48	57	2343
3	2.78	4.76	26	487
4	3.02	5.55	58	2426
5	2.86	5.06	38	1041
6	3.01	4.5	195	27429
7	2.83	5.77	52	1950
8	2.80	4.78	129	12003
9	2.70	4.79	197	27994
Summary	**2.86**	**5.03**	**94**	**6373**

Table 4.1: Results of enumerating local optima (maxima) for the CIFAR-10 classification instance.

Next, we look at the multi-modality of the landscapes. Table 4.1 displays the enumeration results of local-optima for CIFAR-10. The estimation is performed

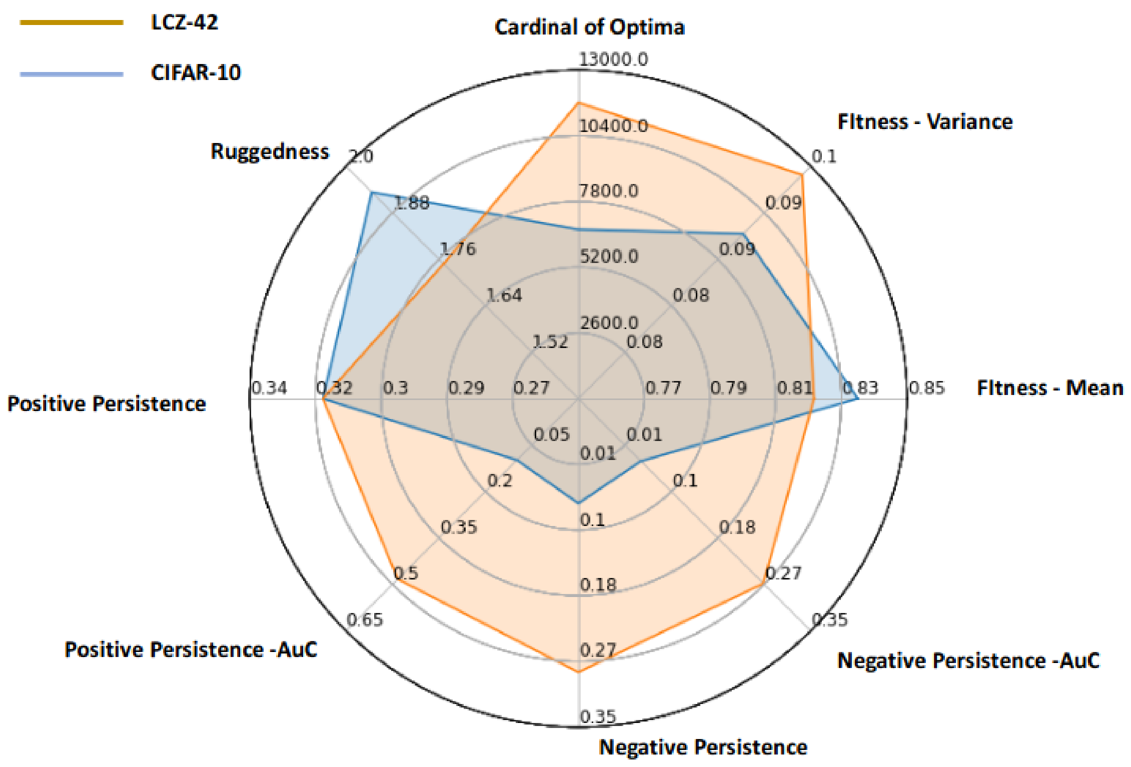


Figure 4.3: Fitness landscape footprint describing the classification instances CIFAR-10 and So2Sat LCZ42, using the NASBench-101 search space.

considering $T = 9$ trials. We find that for BILS(·), the average improvement in fitness until reaching a local maximum is of 5.03 percentage point in accuracy. In average, we manage to deploy BILS(·) consecutively for $k_{mean} = 94$ times, until a LO is found twice. Overall, we estimate the number of LO 6373, which appears relatively large, considering the low number of steps needed to reach one.

Next, we analyze the *fitness footprint* obtained for the instances CIFAR-10 and So2Sat LCZ-42, as visualized in Figure 4.3, respectively, in blue and orange. We find that there is a larger number of O on the landscapes of the So2Sat LCZ-42 NAS instance with 11153 LO against 6373, for CIFAR-10. This suggests more chances for local search-based strategies to get stuck in suboptimal solutions, when using LCZ-42 as training data, as opposed to CIFAR-10. Moreover, the overall fitness is similar for both datasets, with a slightly larger mean fitness for CIFAR-10 of about 83% of overall accuracy after 36 epochs, and slightly larger variance for LCZ-42 of about 10%. This suggests that the overall performance might be similar when doing NAS using either dataset. Besides, we find that the ruggedness is more prominent when using CIFAR-10 (1.93 against 1.75), suggesting a more challenging landscape for local search-based NAS heuristics. Overall, the persistence is more considerable using LCZ-42 than CIFAR-10 (positive and negative). This indicates more predictability in the performances of models from early in their training. This suggested that performance estimation strategies such as learning curve extrapolation could benefit more when performing NAS using LCZ-42 than CIFAR-10.

To summarize, we have successfully provided a quantitative appraisal of key elements of the landscapes of two different NAS classification instances using the bag of features included in the fitness footprint. The following section presents ideas that could be explored in the future to best take advantage of the current framework or extend it.

Note: Several of our FLA experiments capture results with respect to a GO. When using CIFAR-10, the GO (global maximum) consists of the best solution within an exhaustively evaluated search space, while for So2Sat LCZ-42, it is approximated as the best solution within a set of LHS sampled solutions. We find that the trends observed in the FLA using both estimations do not significantly differ[1].

4.1.4 Outlook

Next, we discuss areas of research that could benefit from the current research, as well as ideas of extensions for the current work.

Enabling Comparisons: First, is about exploring the ability to compare search problems. An example of a question that could be tackled with this tool is the following. *How does the AutoML problem change when key elements of the ML setting are changed?*. Answers to these questions could be a quantitative appraisal, based on the features gathered in the *footprint*, of the changes generated by comparing the landscapes of the different problems. A potential topic of interest for such comparison would be *multi-fidelity* in NAS, i.e. NAS search problems in the context of approximations in the fitness measurements based on varying experimental settings. This is explored in the form of multi-modal decision-making, i.e. *sensor-fidelity*, in Section 4.2. Other aspects of *multi-fidelity* in NAS could also be relevant, such as data-fidelity or training-fidelity.

Beyond Classification: Classification is at the basis of CV and remote-sensing-based decision-making. Besides, many modern applications have spanned from it, with as examples, Object Detection, image and scene segmentation, or even generative tasks such as image super-resolution and cloud-removal. These are getting widely adopted by RS experts and practitioners, solving real-world use cases. For these reasons, we find that investigating NAS for the respective scenarios and instances could bring novel insights that would benefit the respective research communities. In practice, researching the application of the footprint to such a problem would involve some research and development on the specification of the landscapes. For instance, this would require identifying relevant search space definitions, architecture solution representation and neighbourhood operators.

Beyond NAS: As a sub-field of AutoML, NAS mainly specializes in designing optimal architectures for data-driven decision-making. While this aspect plays a vital role in improving the capabilities of the overall system, tuning other aspects of a pipeline could also benefit this purpose. Thus, we propose to consider not only NAS AutoML optimization problems, but go beyond and consider the generic problem of HPO[13], or even the the problem of jointly optimizing a target architecture and hyperparameters relevant to it. More precisely, we propose to study the landscape of problems where the fitness function, the search space, as well as the neighborhood operator involved, are dedicated to the respective HPO, or joint HPO and NAS optimization problems. Section 4.3 explores this direction.

4.2 Enabling Sensor-aware NAS for Multi-modal Remote Sensing Scene Classification

This current chapter tackles the topic of *sensor-fidelity* in NAS, for tasks where identifying the most beneficial sensor, or combination of several, is important for improving decision-making abilities.

Research questions tackled: *RQ2, RQ3.*

Related contributions: *C2.*

4.2.1 Motivations

Modern Deep Learning-based decision-making models have positively impacted the development of RS technologies, addressing the need for more accurate and versatile models. While the community initially focused on single-modality-based models, the limitations of such solutions led to investigating means of incorporating more modalities[64]. Indeed, the ability to sense the object of the ground highly depends on the type of sensor being used. Thus, leveraging the complementary of several sensors in their coverage of the information being retrieved would be a desirable feat for a data-driven model. Furthermore, being able to provide with optimal multi-modal data-driven model by means of automation and optimization (NAS), could lead to additional breakthroughs in the field.

For this purpose, we propose to see multi-modal decision-making NAS search instances, as instances of multi-fidelity optimization. More precisely, sensor-fidelity NAS, i.e., a setting where the amount of sensor used during the search can vary. This is could happen, for instance, in the context of the EO classification tasks where several sources of input data are available, and combining several sources of input comes at an increased computational cost. Then, sensor-fidelity NAS would help to find an optimal classifier at a reduced computational cost. We propose to study the influence of this fluctuation in input sensors on the behavior of search, using the *fitness landscape footprint* introduced in Section 4.1.

4.2.2 A Landscape Analysis of NAS using Various Sensors

In this section, we present the results of landscape analysis in the context of multiple data-fidelity-oriented neural architecture optimization. Experiments are conducted using the same experimental protocol and input data as in Section 4.1.3. More precisely, the task being tackled is classifying Local Climate Zones using the So2Sat LCZ-42 dataset, considering both Sentinel-1 and Sentinel-2 sensors. The landscape analysis is mainly conducted using the *fitness landscape footprint*, and aims at comparing the *effect* of the input sensor on the NAS optimization process and performances.

Ruggedness Figure 4.4 displays the results of a random walk route (N=100 steps) being evaluated on the instance of the So2Sat LCZ-42, using various sensors as input. In blue, yellow, green, and red are the results, respectively, for using Sentinel-1, Sentinel-2, both sensors and Cifar-10. The route is the same for all sensors, is obtained using the search space of NASBench-101, and the solutions in the route are all trained for 36 epochs using the same hyperparameters. We find that using Sentinel-2 provides the largest mean fitness, closely followed by the case of using both sensors. Additionally, the routes have a very similar curvature, with LO appearing to be at the same steps in the walk. When using Cifar-10, the observed mean fitness is much lower. It is at its lowest when using Sentinel-1. Given that the search space and route have displayed good fitting capabilities (e.g., when using Sentinel-2), this suggests that fitting Sentinel-1 and Cifar-10 are much more challenging tasks. The route might benefit from a longer training budget in order to improve its fitness for these two sensors.

Besides, we find that the better the fitness, the lesser the irregularities in the routes, measured by a larger ruggedness. On the other hand, the lower the fitness, the more irregularities in the routes, and the larger the ruggedness.

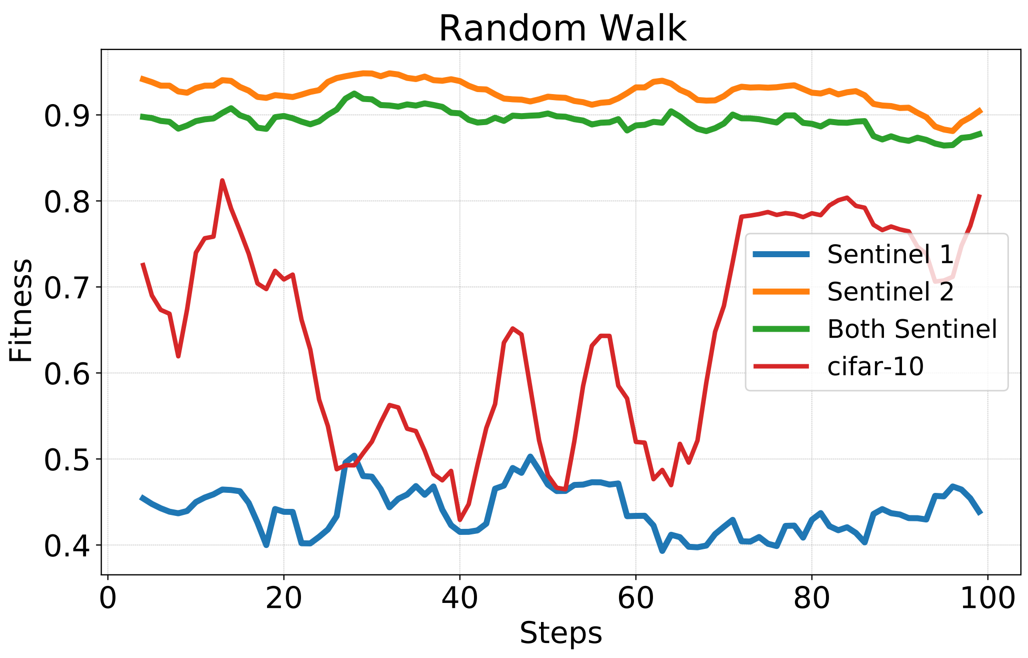


Figure 4.4: Results[2] of a random walk evaluated using various input sensors, on the search space of NASBench-101. All solutions are trained for 36 epochs, and the routes are $N = 100$ steps long.

Fitness Footprint Next, we show how the *Fitness Landscape Footprint* is used to make a quantitative, summarized description of the sensor-aware NAS problem that is being tacked.

First, let's consider the overall fitness, captured in the *Footprint* with the `Average Fitness`. As discussed before, we found that the NASBench-101 search space achieves a larger mean fitness in this task when using the Sentinel-2 sensor, followed by input-level fusion and, lastly, the Sentinel-1 sensor only. This phenomenon is possibly explained by the fact that fitting the So2Sat LCZ-42 using SAR imagery might be harder than optimal imagery (Sentinel 2), due to the lack of information in the sensor relevant for distinguishing certain classes. Then the sensor fusion might yield intermediate performances for the same reasons. When it comes to Cifar-10, we know from other studies that the NASBench-101 search space is able to properly fit the task. The poorer fitness could be improved with a larger training budget.

Besides, the larger the fitness of a given sensor, the fewer irregularities in the landscape, as seen in the random walk routes. This is captured by measuring lower `ruggedness` scores, as well as a lower `standard deviations` in the overall fitness.

Regarding the `persistence` of the fitness, the scores are reported in our study published in the annals of ISPRS in 2022[2]. Overall, we found that the `persistence`, whether positive or negative, is larger in the case of searching with a sensor that is easier to fit (larger overall fitness), and with a lower ruggedness.

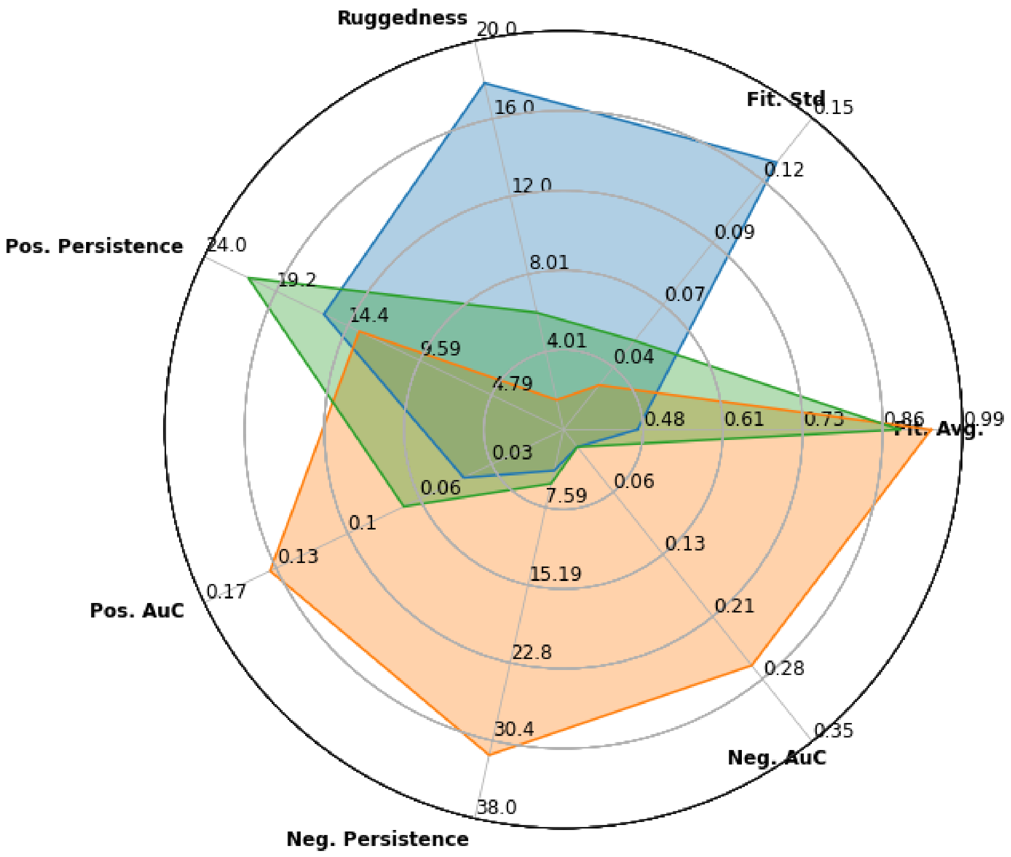


Figure 4.5: Fitness landscape footprint applied to the NASBench-101 search scenario, given 36 epochs of training. In blue, yellow, and green are the results respectively for using as sensor Sentinel-1, Sentinel-2, and an input-level fusion of Sentinel-1 and Sentinel-2.

To summarize, this comparative analysis of the *Footprints* generated from using the search space in the different sensor settings suggest an easier landscape when using Sentinel-2 only, followed by the use of the input-level fusion. This demonstrates the ability of this tool to successfully support the analysis and design of sensor-aware AutoML problems.

4.2.3 Outlook

Next, we discuss aspects of the project that deserve to be explored more in the future. An instance of the fruitful topic is *data-fidelity* in NAS, already explored in the current project in the form of *sensor-fidelity*, i.e., researching if a NAS search problem difficulty on a given dataset changes based on the sensor used to tackle it (including *sensor fusion*). We identified that the landscape of the search problem would have a similar feature regardless of the sensor being used, in particular when training budgets were large. We speculate that this would hold true even on problems where the number and variety of input modalities are much larger[65]. Future work might research the validity of the hypothesis and how insights gained on the topic could help improve NAS search strategies by means of performance prediction for expensive multi-modal problems, approximated by the model performance of a lesser *sensor-fidelity* (i.e. less expensive modalities).

4.3 Beyond Search Spaces: how do other important components affect autoML problems ?

This chapter explores the idea of a landscape analysis of AutoML problems for investigating the role of sometimes overlooked elements (evaluation pipelines) in the success of AutoML search strategies.

Peer-reviewed publication: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang, We Won't Get Fooled Again: When Performance Metric Malfunction Affects the Landscape of Hyperparameter Optimization Problems, International Conference on Optimization and Learning, CCIS, 2023.

Research questions tackled: *RQ3.*

Related contributions: *C3.*

4.3.1 Motivations

Previous research at the intersection of landscape analysis and AutoML has mainly focused on providing novel insight into the target optimization problems using the search space as the main factor of influence for the search difficulty and performance. For instance, authors have researched the correlation between the metric of neutrality in the optimization landscapes of hybrid end-to-end AutoML systems[66], with the fitness. Other authors have analyzed the landscapes associated with optimizing graph-based decision-making models (GNNs)[67], providing an appraisal of the general difficulty of the respective NAS problem. However, to our knowledge, very few works have gone beyond the analysis of the effect of this particular component on the search process. This project is an attempt at using FLA and the *footprint*, introduced in Section 4.1, to characterize generic HPO problems. In particular, it researches the effect of performance estimation pipeline definition on HPO search difficulty.

4.3.2 Tracking the Impact of Evaluation Metric Failures on the Easiness of HPO Problems

We propose to analyze the HPO problems using a variety of tools[68]. First, we consider the landscape of HPO problems $\mathcal{L} = (S, f, N)$, where S denotes the HP configuration space, f the fitness evaluation function, and N a neighborhood operator. Note that N designed to handle HP configuration space of very heterogeneous nature (i.e. mixed types), when attributing neighbors to a solution. Since we aim to understand the effect of the fitness function on the HPO optimization landscapes, we propose to analyze the problems on interest using the FDC[68]. Indeed, by investigating the correlation between the fitness values in a configuration space, and the distribution of distances of HP configurations to the GO, we expect to notice any side effects in fitness evaluation functions. In particular, any misattribution in fitness to an HP solution might be easily identifiable in the FDC. Second is an analysis of the local relationship between solutions and their neighbors, in terms of fitness. Similarly, pathological fitness evaluation functions might be associated with an altered relationship between those quantities. Third, related to the locality, is an analysis of the neutrality in the landscapes. It defines as $N_d(x) = |\{x' \in N(x) \mid\mid f(x') - f(x) \mid < \epsilon\}|$, where the *neutrality degree* N_d of a solution x corresponds to the number of neighbors of similar fitness it has. This quantity would also be affected in case of misattributed fitness.

4.3.3 The Case of the MMCE Metric

The following section introduces results obtained considering the HPO benchmark DS-2019[69]. It consists in the scenario of tuning the HPs of CNNs on ten image classification instances. For all instances, HP configurations were evaluated considering the mean misclassification error (`MMCE`) as a performance evaluation function. Additional experiments were conducted using the YAHPO benchmark[70], encompassing up to 119 classification instances, restricted in our case on the tuning HPs of CNNs, considering the predictive accuracy (opposite of `MMCE`) or `Log Loss`.

Fitness Distance Correlation Next, we analyze the landscapes from the perspective of the FDC. Figure 4.6 shows results of FDC for the instances of CIFAR-10, FLOWER, SCMNIST and SVHN. These findings are representative of those made across all the other instances in the DS-2019 dataset. Overall, we find that there is little correlation between the two variables of interest: the fitness of a given HP configuration does not always improve with its proximity to the Global Optimum (GO). Note that we approximate the GO as being the HP configuration with the highest fitness (predictive accuracy) in the configuration space. Besides, we observe that the distributions of distances of HP samples to the GO are all wide and appear uniform. This suggests that there is diversity among the sampled HPs and possibly across the HP configuration space. Regarding the distributions of fitness of the samples, we make two different types of observations. First, for the classification instance CIFAR-10, we find that the distribution is large and uni-modal, with a mode at large values (80% of accuracy).

Second, for the instances FLOWER, SCMNIST and SVHN, we find an unexpected mode at odd values in the distributions of fitness. There are observed for the around 22% (FLOWER), 65% (SCMNIST) and 20% (SVHN) of fitness. Moreover, the distances to the GO of the solutions having this unexpected performance cover about 50% of the possible range in most cases. This suggests that the solutions in the set exhibit this behaviour irrespective of their HP configurations.

After analyzing the dataset, we find that they have an imbalance in their distributions of labels, and that these numerical values are close to the prevalence of the majority class. This suggests that these solutions are naive classifiers, i.e. majority class predictors.

Average Fitness in the Neighborhood Next, we analyze the instance from the perspective of the locality, in order to generate additional insights about the effects on the respective landscapes. Figure 4.7 introduces the result for the same instances. We find that in the case of the nonexistence of naive classifiers (CIFAR-10), the correlation between the two variables of interest seems strong. More precisely, for solutions exhibiting a fitness in any of the possible ranges of values, the average

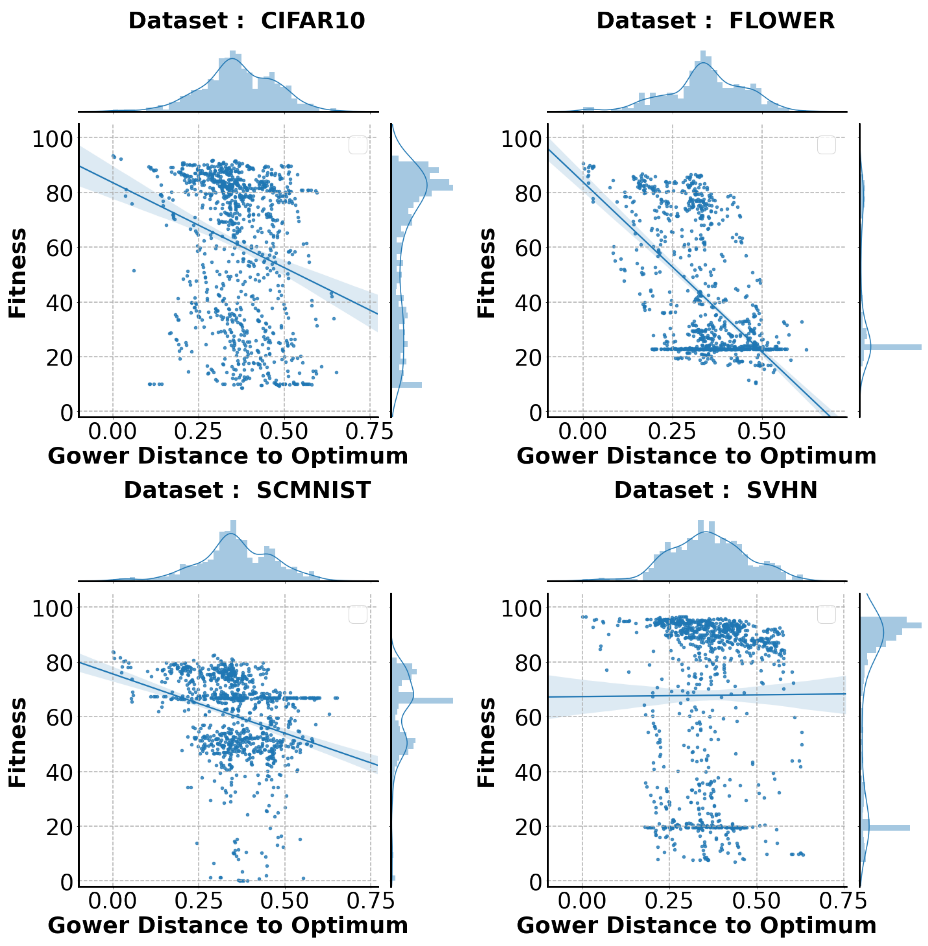


Figure 4.6: Fitness Distance Correlation (FDC) for the HPO instances of CIFAR-10, FLOWER, SCMNIST and SVHN. All plots obtain considering a sample size of $N = 1000$ of hyperparameter configurations, for the problem of tuning the HPs of a ResNet-18 image classifier[3].

fitness of their neighbors will be similar. This could be interpreted as the landscapes having little irregularities.

On the other hand, we find that the correlation is worse, when the landscapes have multi-modal distributions of fitness, with potentially many unexpected naive classifiers. In these cases, there are many ranges of fitness for which the average fitness in the neighborhood is dissimilar but close to the value associated with the prevalence of the predominant class.

This also suggests that the landscape is has ruggedness, with possibly many local optima that are in fact sub-optimal naive classifiers.

Neutrality Next, we complement the previous analysis with results covering the aspect of the neutrality of the landscapes. Figure 4.8 introduces these results for the same instances. We find that whenever the correlation between the fitness of a current solution and its neighbors is low, the neutrality degree is also relatively low.

To summarize, the FDC enables us to detect a way in which the vulnerability of the performance metric affects the landscape of the HPO instances being tackled. More precisely, in the case of the predictive accuracy, the vulnerability is the possibility of turn any classifier into a naive classifier, a sub-optimal behaviour ran-

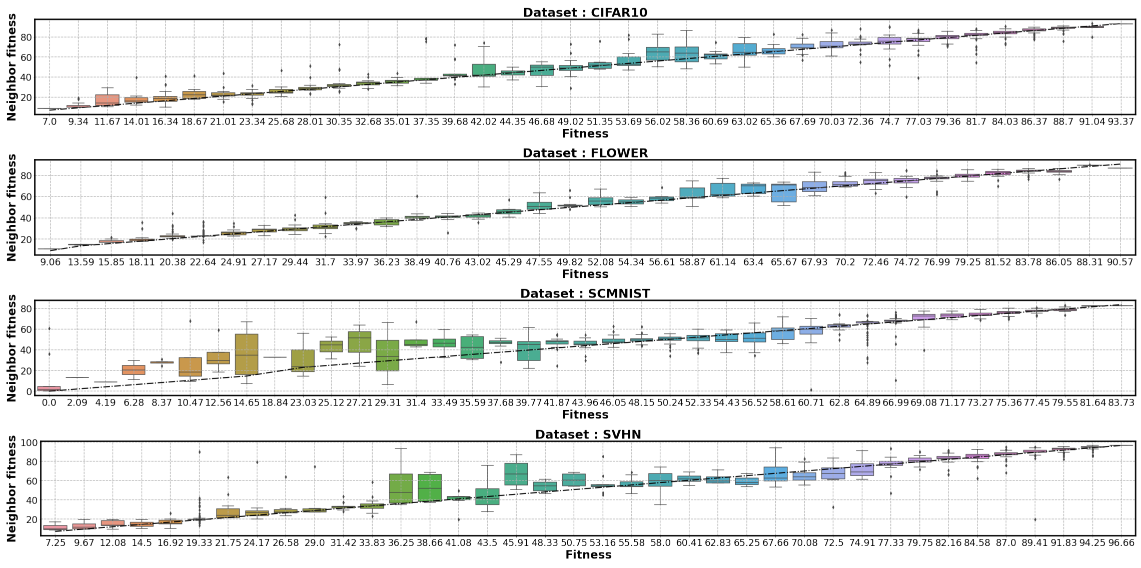


Figure 4.7: Average neighbor fitness as a function of the observed fitness, for the instances of CIFAR-10, FLOWER, SCMNIST and SVHN. All plots are obtained considering a sample size of $N = 1000$ of hyperparameter configurations for the problem of tuning the HPs of a ResNet-18 image classifier[3].

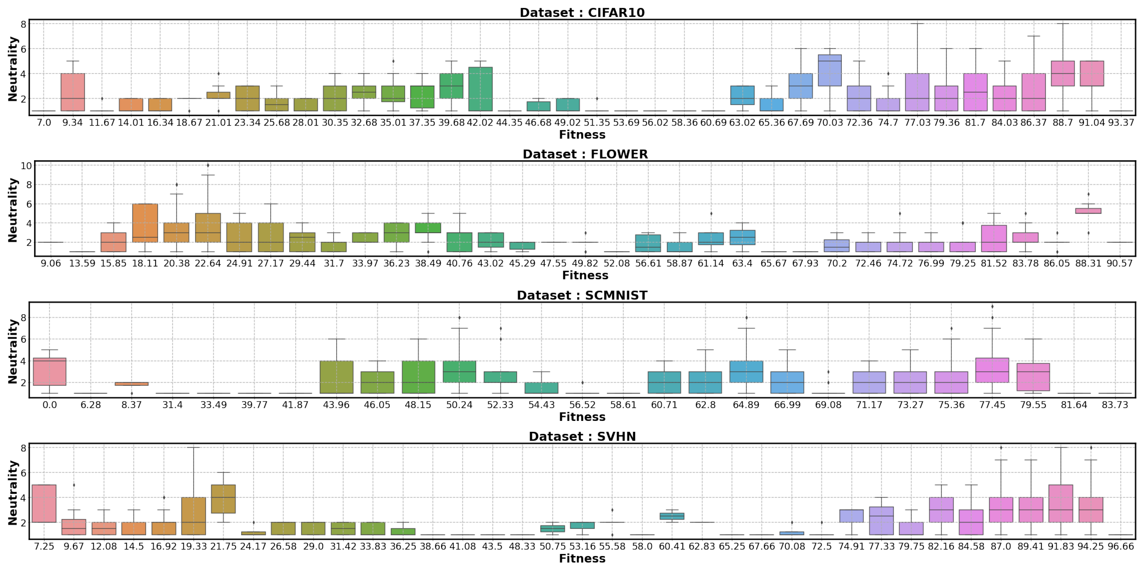


Figure 4.8: Neutrality scores for the instances of CIFAR-10, FLOWER, SCMNIST and SVHN. All plots are obtain considering a sample size of $N = 1000$ of hyperparameter configurations, for the problem of tuning the HPs of a ResNet-18 image classifier[3].

domly arising during training. This would happen irrespective of the models' HP configuration, but instead in relationship with the nature of the class distribution (imbalance) of the dataset, for the instance being tackled.

Analyzing the relationship between solutions and their neighbors, for the same instances, suggests that the existence of naive classifiers is associated with more rugged landscapes. Results on the neutrality degree of the landscapes support the claim. Indeed, areas with a low correlation of fitness between solutions and their neighbors are associated with lower neutrality degrees, indicating more irregularities (local optima) in the landscapes.

Similar results are observed regarding the influence of sub-optimal models generated when using the Log Loss for classification.

4.3.4 Outlook

Next, we discuss directions in which this research could be extended. First is regarding the aspect of quality control of benchmarks. Creating of AutoML benchmarks, whether specific to HPO[71], NAS[40,44], or for hybrid[72,45] search scenarios, requires investing a large number of computational resources in order to evaluate *partially*[40], or *exhaustively*[41] large search spaces. Our findings suggest that some design choices, e.g. performance metrics, can have adverse side effects, e.g. diverse HPs turned into naive classifiers. These side effects are hard to track but might have an impact on the quality of the benchmarks, by biasing and altering the underlying search landscapes. Future research might investigate how to use FLA more systematically in order to build robust AutoML benchmarks (see Section 6).

Second, is about understanding how the results obtained transfer to the domain of EO. Given the scarcity in the occurrence of certain phenomenon or classes of interest in EO-related applications, such as the unbalance in the distribution of population across the globe[65], EO datasets designed for data science-based analysis tend to have a class imbalance. Future research might investigate how the choice in evaluation metrics could affect the success of AutoML algorithms used to solve such EO classification tasks. In particular, what is the predominance of naive classifiers among all sampled and evaluated solutions? What fix to the evaluation metrics could help reduce it?

This page is intentionally left blank.

Chapter 5

Mechanisms for Efficient Search in AutoML

The current chapter introduces contributions to the topic of Efficient AutoML. First, we describe our work dealing with search initialization and its connection to the efficiency of NAS search algorithms. Second, we introduce our research on using differentiable search for finding compact and efficient classifiers.

5.1 A Data Science-based Approach to Improving AutoML Search Algorithms

This section introduces our first contribution to the topic of mechanisms for *search efficiency* in AutoML for EO. More particularly, it revolves around improving the initialization of NAS search algorithms using a data-science-based approach.

Peer-reviewed publication: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang, A Data-driven Approach to Neural Architecture Search Initialization, Annals of Mathematics and Artificial Intelligence, Springer 2023.

Research questions tackled: *RQ1, RQ2.*

Related contributions: *C4.*

5.1.1 Motivations

Initialization search algorithms in AutoML are traditionally based on a (random) sampling of solutions in a manner agnostic (independent) of the task. Besides, the recent growing availability of databases in all areas of Data Science has encouraged the AutoML research community to explore the benefits of databases and platforms specific to that area of expertise. This has resulted in the creation of databases and open-source platforms gathering performance data of all kinds (e.g., HPO, NAS solution configurations) in order to ease the prototyping and benchmarking and support reproducibility in methodological research in the field. An example of this, NASBench-101[40], provides the exhaustive evaluation of a NAS search space covering about distinct 500k NN configurations. In this context, the following questions arise: can we benefit from such data to help better initialize AutoML search algorithms? Would a better initialization help improve short and long-term search algorithm performance? The research conducted on this topic aims at answering these questions.

5.1.2 Data-driven Initialization for Population-based NAS Heuristics

Next, we introduce the proposed initialization method. First, we describe the overall procedure visualized in Figure 5.1. Then, we present the feature encoding designed to represent solutions in the following experiments.

Objective: The proposal aims to help search algorithms reach better short- and long-term performance improvements. The type of search algorithm we target is a population-based search algorithm. For instance, meta-heuristics, Bayesian-based approaches, etc.

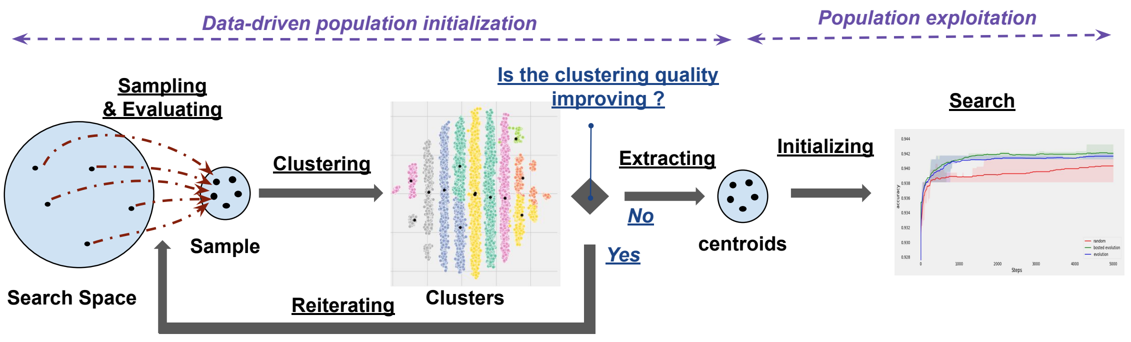


Figure 5.1: Flowchart of the data-driven initialization methodology[4]

Main Algorithm Algorithm 3 introduces the algorithm behind the proposed procedure. In order to operate, it requires the following inputs: μ, the size of the population used to operate the search strategy, a list of sample sizes *list_sample_sizes*, the feature encoding type ε, a clustering algorithm α, a list' M of metrics to evaluate the clustering, and the search space Ω.

Then, it iterates over all sample sizes in *list_sample_sizes*, in order to find the value helping reach the best clustering quality. For each value of sample size, it will collect an appropriate sample from the search space Ω, using the procedure `RandomSampling(·)`. Then, it evaluates the fitness of the sample using `EvaluateModel(·)`. Then, it proceeds with retrieving the proper representation for the sample using the feature encoding ε and the procedure `EncodeModel(·)`. Moreover, we find that the computational complexity of the several steps of the algorithm depends on the size of the feature representation used. In order to alleviate this aspect, we proceed with reducing the dimensions of the encoded sample using (`ReduceDimensions(·)`). In order to obtain the corresponding clustering labels, `ClusterFeatures(·)` is applied. The quality of the current sample is measured using the set of metrics M, labels, and the function `Assess(·)`. Finally, the clusters are obtained as the combination of the encoded samples and their clustering labels,

Algorithm 3: Data-driven Initialization Technique[4]

```
input: The desired population size $\mu$, a list of sample sizes list_sample_sizes
  (decreasing order), the feature encoding type $\varepsilon$, a clustering algorithm $\alpha$, the list
  of metrics to evaluate the clustering quality $M$, and a search space $\Omega$.
CLUSTER_QUALITY ← $\emptyset$
for SAMPLE_SIZE ∈ LIST_SAMPLE_SIZES do
    SAMPLE ← RandomSampling($\Omega$, SAMPLE_SIZE)
    SAMPLE ← EvaluateModel(SAMPLE)
    ENCODED ← EncodeModel(SAMPLE, $\varepsilon$)
    REDUCED ← ReduceDimensions(ENCODED)
    LABELS ← ClusterFeatures(REDUCED, $\alpha$, $\mu$)
    NEW_CLUSTER_QUALITY ← Assess(REDUCED, LABELS, $M$)
    if CLUSTER_QUALITY > NEW_CLUSTER_QUALITY then
        break
    end
    CLUSTER_QUALITY ← NEW_CLUSTER_QUALITY
    CLUSTERS ← (ENCODED, LABELS)
end
CENTROIDS ← ExtractCentroids(CLUSTERS)
INIT_ARCHITECTURES ← DecodeModel(CENTROIDS, $\varepsilon$)
return INIT_ARCHITECTURES
```

Once an appropriate sample size is identified, the centroids are extracted from the corresponding clusters (`ExtractCentroids(·)`). The initial population is obtained after retrieving the solutions in the search space associated with the encoding of the centroids (`DecodeModel(·)`). The procedure then returns the initial population to the search strategy.

Encoding of the Data: The initialization algorithm relies on a cluster analysis of the search space. In order to be successful (high clustering quality based on metrics M), it collects a sample that is encoded using specific feature representation, followed by a dimension reduction and clustering. This section focused on describing the custom feature representation we design for the purpose.

It aims at constructing an encoding that helps represent all models from the search space compactly. The desired properties are for it to be generic, i.e., task agnostic, to contain information about model structure and performance in train and validation, at least. Then, the output is to be provided to any search algorithm and help it retrieve models based on the relevant information it contains (model structure and performance).

There are two versions of the encoding: the first is referred to as *Original, or Short encoding*, and the second, *Binary, or Long encoding.* Both feature representations combine the following elements identifying solutions in the search space as shown in Table 5.1: the adjacency matrix, the list of operations, and the list of all measured fitness. The representation mainly differs in the way they allow to express the combination made in the selection of constituents in each NN solution, as shown in Figure 5.2. The *Original (Short) encoding* is obtained by flattening (Major row) of the original adjacency matrix, concatenating it to the list of operations in the selected solution, also concatenated with the list of fitness of the solution, in the order described in Table 5.1. The *Binary (Long) encoding* is obtained by the same series of operations, except it makes use of an alternative representation of the solution in the expanded adjacency matrix, that is visualized in Figure 5.2.

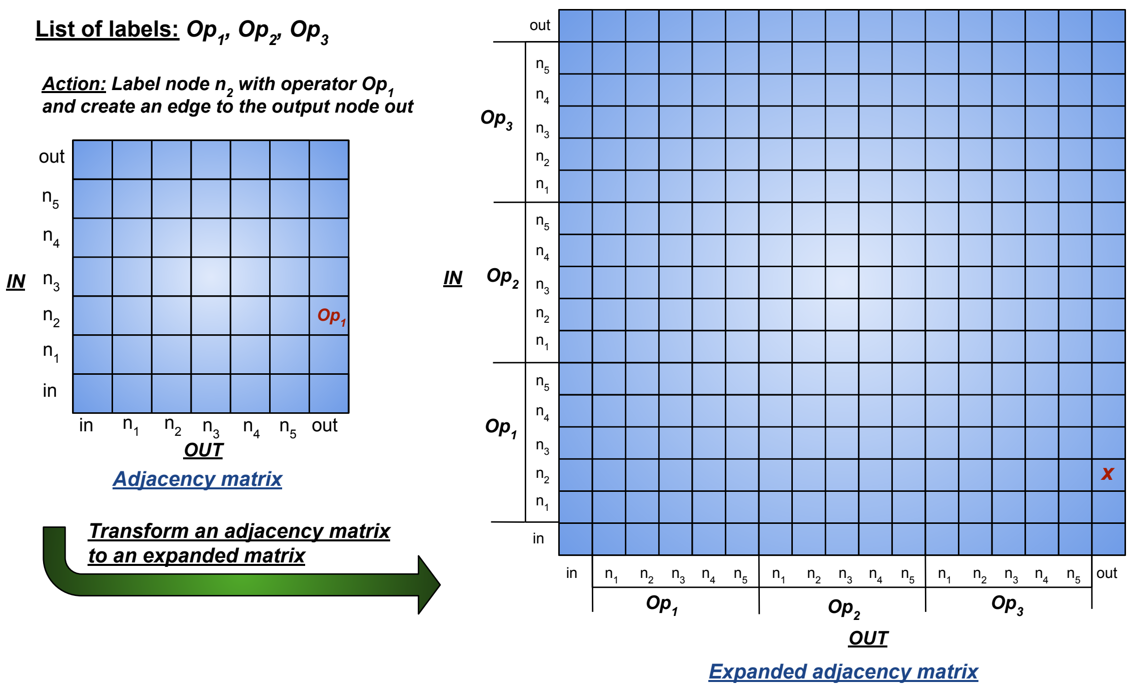

List of labels: Op1, Op2, Op3
Action: Label node n2 with operator Op1 and create an edge to the output node out
out
n5
n4
IN n3
n2
n1
in
Op1
in n1 n2 n3 n4 n5 out
OUT
Adjacency matrix
Transform an adjacency matrix to an expanded matrix
IN
Op3
Op2
Op1
x
Op1 Op2 Op3
OUT
Expanded adjacency matrix

Figure 5.2: Description of the `DAG` attributes used to represent a solution sampled from the cell-based search space: a pair of adjacency matrix and list of operations, or a unique identifier, i.e. the custom adjacency matrix[4].

Feature representation	*Components*
Short	Adjacency matrix + operations + fitness
Long	Expanded adjacency matrix + operations + fitness

Table 5.1: Description of the custom encodings (`Short` and `Long`) designed in order to perform a cluster analysis of the search space.use Each encoding is in the form of a vector. The symbol `+` stands for the concatenation operator.

Baseline Search Algorithm: In order to perform an experimental benchmark evaluating the proposed initialization approach, we select three baseline algorithms to initialize: a Genetic Algorithm (GA), an Evolutionary Algorithm (($\mu + \lambda$)EA) and the *Aging Evolution* (EA). These algorithms are population-based meta-heuristics inspired by natural computation, which evolve a population of solutions using various mechanisms. GA uses selection, crossover, mutation, and replacement operations to completely replace a population of fixed size, after a generation. On the other hand, $(\mu + \lambda)$ EA evolves a population of size μ by replacing it with the best μ individuals of the current population concatenated to a set of λ offsprings. Last but not least, the *Aging Evolution* is similar to GA, except that it discards the solution that remained the longest in the current population, i.e. the aging individual, at the each generation. A detailed description of each algorithm is provided in Section *2.3 (Baseline algorithms)* of publication *P2* [4], also available in the Appendix of this manuscript.

Initialization of an Algorithm: In order to initialize a search baseline, we first collect a set of points retrieved using a desired initialization technique. Then, we provide this set as the initial population of size *initial_size*, to the population-based search algorithm (e.g. GA). Given a set of hyperparameters, including the budget of iteration, the algorithm then returns the population of solutions obtained at the last iteration, as well as the history of the solutions encountered. Given this history, we can retrieve the best-encountered solution. The performance of the search baseline is then measured using the mean fitness of the population in validation, as well as best-reached fitness in validation.

Figure 5.3 displays the results of a benchmark of algorithm performance, using various initialization methods. The top (resp. bottom) row displays results given a budget of 36 (resp. 108) epochs of training. The algorithms in the benchmark, from left to right, are GA, $(\mu + \lambda)$ EA, and *Aging Evolution*. They were all deployed 100 different times, each time considering a budget of 2000 search evaluations. The red, green, and blue curves are for initializing using, respectively, random sampling (`rand`), `centroids` and Latin hypercube sampling (`lhs`). The centroids ($N = 19$) were extracted from a clustering using the Bayesian Gaussian Mixture of models (`BGM`), with the Short Encoding as feature representation. In bold is the mean fitness of the current population, and lighter colours are the fitness ranges (min/max).

We find that for all the search baselines, the centroids provide the largest mean fitness for the initial population. Indeed, we report a difference in fitness with `lhs` and `rand` of up to 20 percentage points in validation accuracy.

Besides, EA is the algorithm that takes the best advantage of this initial *boost*: it converges faster and maintains an improved mean fitness over `lhs` and `rand`, independently of the training budget of the evaluations. Moreover, additional results [4] on the performance of the search algorithms after 2000 evaluations, indicate that `EA` and `GA` benefit from significant performance improvement using the `centroids`, over initializing with `lhs` or `rand`.

Transferring from CV to EO: Next, we investigate the possibility of a transfer of performance from the Computer Vision domain to the Earth Observation domain. Based on previous studies, we know that using the same search space for different instances, i.e., different classification datasets, might result in similar model performances and behavior. Then, we ask ourselves, is it possible to use a population of solutions collected from the CV domain [40], to initialize a search algorithm to be deployed on an EO classification instance, using the proposed method ?

We compare the convergence of search baselines using the scenario of Nasbench-101, considering the classification instance So2sat LCZ-42. Figure 5.4 displays the results for the initialization benchmark using random sampling (`rand`) and our approach (`centroids`). Note that the `centroids` are collected on NASBench-101, using the data-driven initialization taking into account fitness evaluation on CIFAR-10, while `random sampling` is task agnostic. The algorithm deployed is EA ($\lambda = 19$), considering a budget of 4 epochs of training, 100 evaluations, and 5 independent runs. Note that the number of independent runs of the experiments is limited to $N = 5$ due to the computationally expensive nature of measuring a fitness eval-

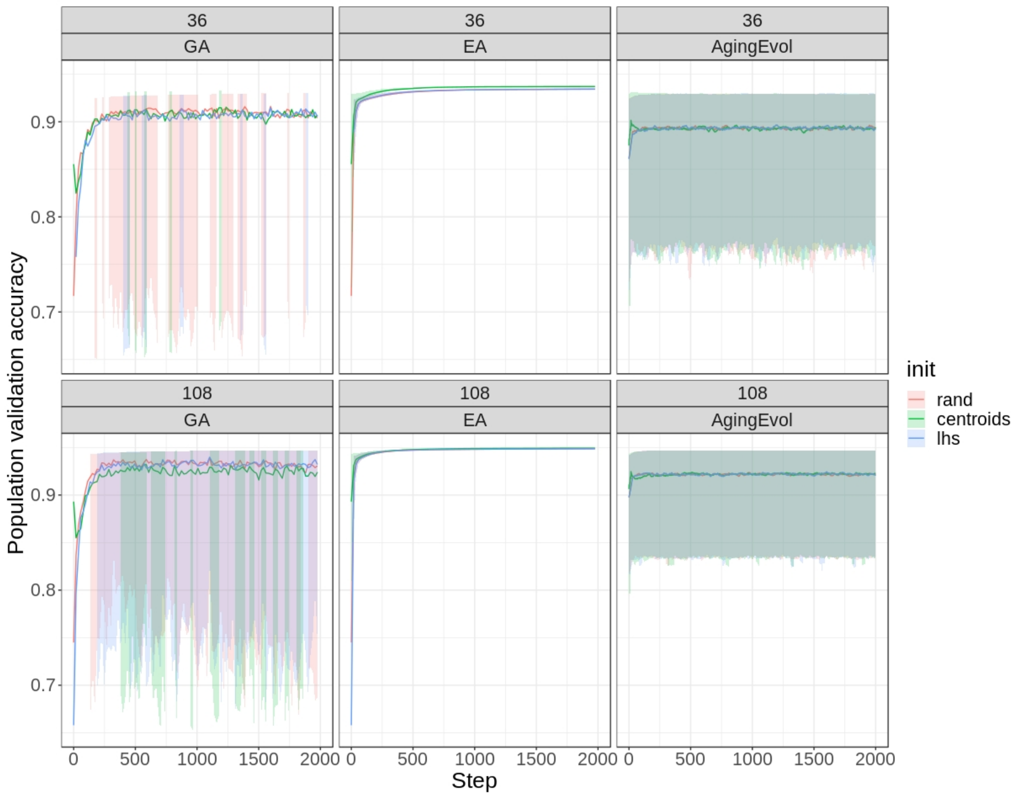


Figure 5.3: Benchmark of NAS algorithm performances, using various initialization methods[4]. The search is performed either when training solutions for 36 (top) or 108 (bottom) epochs. The data-driven initialization techniques involve the Short encoding.

use

uation on the instance So2sat LCZ-42. Indeed, a single full evaluation requires 10 to 15 hours using a NVIDIA A100 GPU accelerator. We find that initializing `EA` using the `centroids` provides a much larger initial mean test accuracy (about 14 percentage points), as well as a maintained higher performance over time.

Our early results regarding *domain and instance transfer* in the context of search algorithm initialization indicate that collecting an initial population on NASBench-101 using the proposed approach also enables faster convergence on another dataset (So2Sat LCZ-42). This is the case even considering the low training budget for the evaluations.

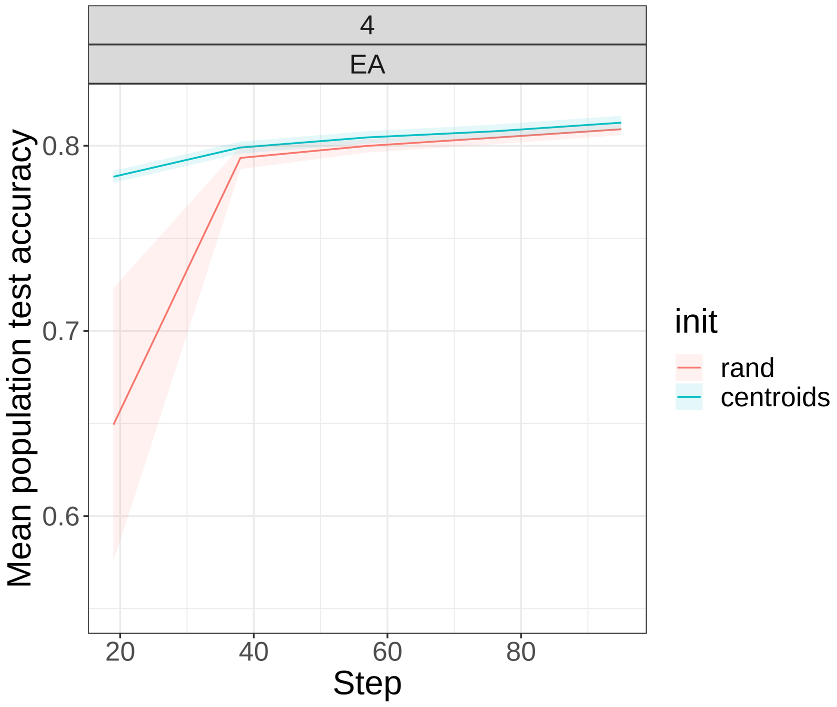


Figure 5.4: Benchmark of initialization[4] (5 seeds), considering the search scenario of NASBench-101, the scene classification instance of the So2Sat LCZ-42. The search baseline is $(\mu + \lambda)$EA, with a budget of 100 evaluations, 4 epochs of training, and $\mu = \lambda = 19$.

Analysis of Solutions Found Next, we analyze the obtained solutions to identify potential model archetypes based on the initialization baselines used to deploy a search strategy. We analyse solutions found using various initialization baselines on the task of searching for classification models on the NASBench-101 search environment. We look for patterns emerging and see if they are specific to the (induced by the) initialization method used. This is done by comparing the distributions of extended adjacency-matrix, i.e., encoding of the retrieved models.

Figure 5.5 introduces results of solutions found using the *Short Encoding*. More precisely, it shows the distribution of activation in the adjacency matrices of solutions gathered based on the initialization and the search algorithm used. Table 5.2 provides the legend to interpret the adjacency matrix representing a solution. Each letter is associated with the following information: the node label (given operation, input, or output node), and its index in the `DAG` representing the solution. Figures

5.5a (top) and 5.5b (bottom) are respectively for solutions found using a budget of training of 36 and 108 epochs. The first, second and third rows are, respectively, for initializing using a random sample (`rand`), the proposed method (`centroids`) and `LHS`. The first, second and third columns are, respectively, for initializing GA, EA, and Aging Evolution.

We find that the activations in the adjacency matrices are distributed in the form of more widespread clusters, the longer the training budget allowed for evaluating solutions. Besides, the type of algorithm used tends to have a slight influence on the results: AE provides solutions with activations gathered in smaller clusters than GA and Aging evolution. Similarly, the initialization procedure is also of importance: there appears to be more diversity (widespread activations) in the solutions obtained using LHS or Random Sampling than with the centroids.

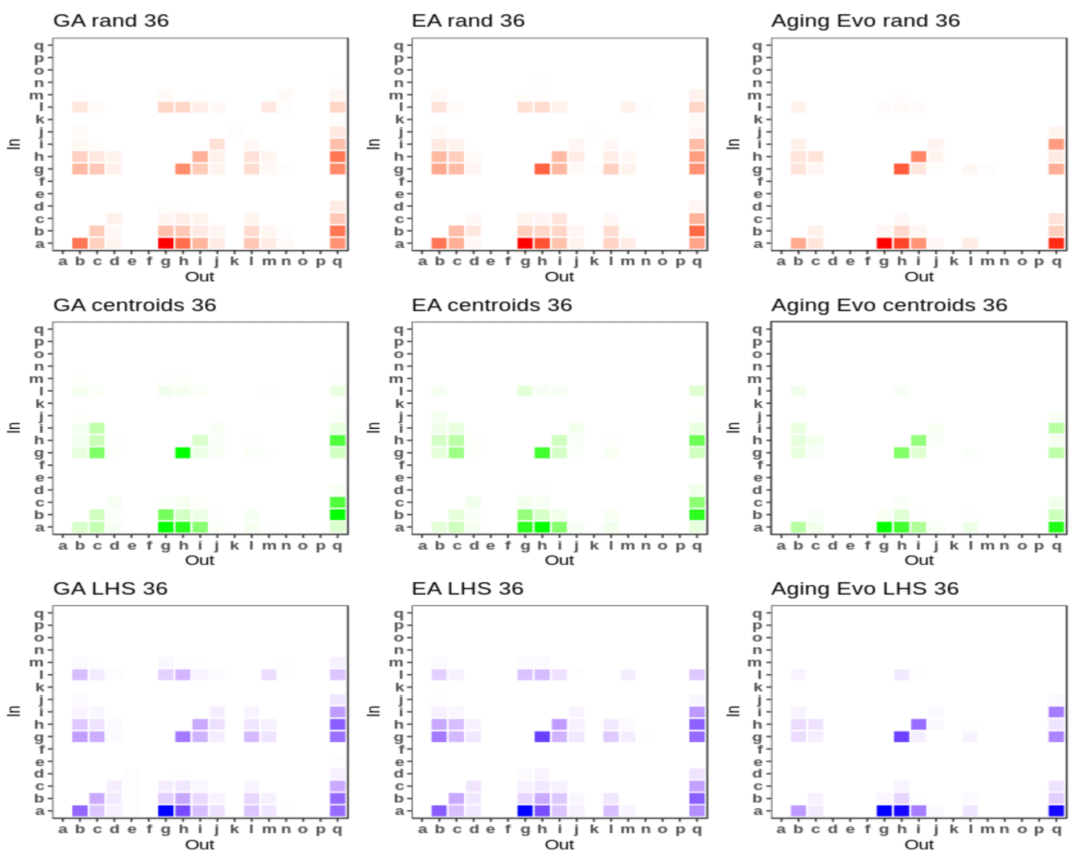
To summarize, we find that the diversity in the solutions found is influenced by various factors such as the training budget, as well as the initialization method used. The longer the training, the more diverse the solutions: this suggests that the flatness in the search landscapes that comes with a more considerable training time enables the retrieve of local optima in various regions of the search space. Regarding the initialization, the centroids might already be local optima; thus, using them as an initial population might limit the exploration of the search space and result in less diversity.

Figure label	*Node Label*	*Node index*
a	input	0
b	Conv 1x1 + BatchNorm + Relu	1
c	Conv 1x1 + BatchNorm + Relu	2
d	Conv 1x1 + BatchNorm + Relu	3
e	Conv 1x1 + BatchNorm + Relu	4
f	Conv 1x1 + BatchNorm + Relu	5
g	Conv 3x3 + BatchNorm + Relu	1
h	Conv 3x3 + BatchNorm + Relu	2
i	Conv 3x3 + BatchNorm + Relu	3
j	Conv 3x3 + BatchNorm + Relu	4
k	Conv 3x3 + BatchNorm + Relu	5
l	MaxPool 3x3 + BatchNorm + Relu	1
m	MaxPool 3x3 + BatchNorm + Relu	2
n	MaxPool 3x3 + BatchNorm + Relu	3
o	MaxPool 3x3 + BatchNorm + Relu	4
p	MaxPool 3x3 + BatchNorm + Relu	5
q	output	6

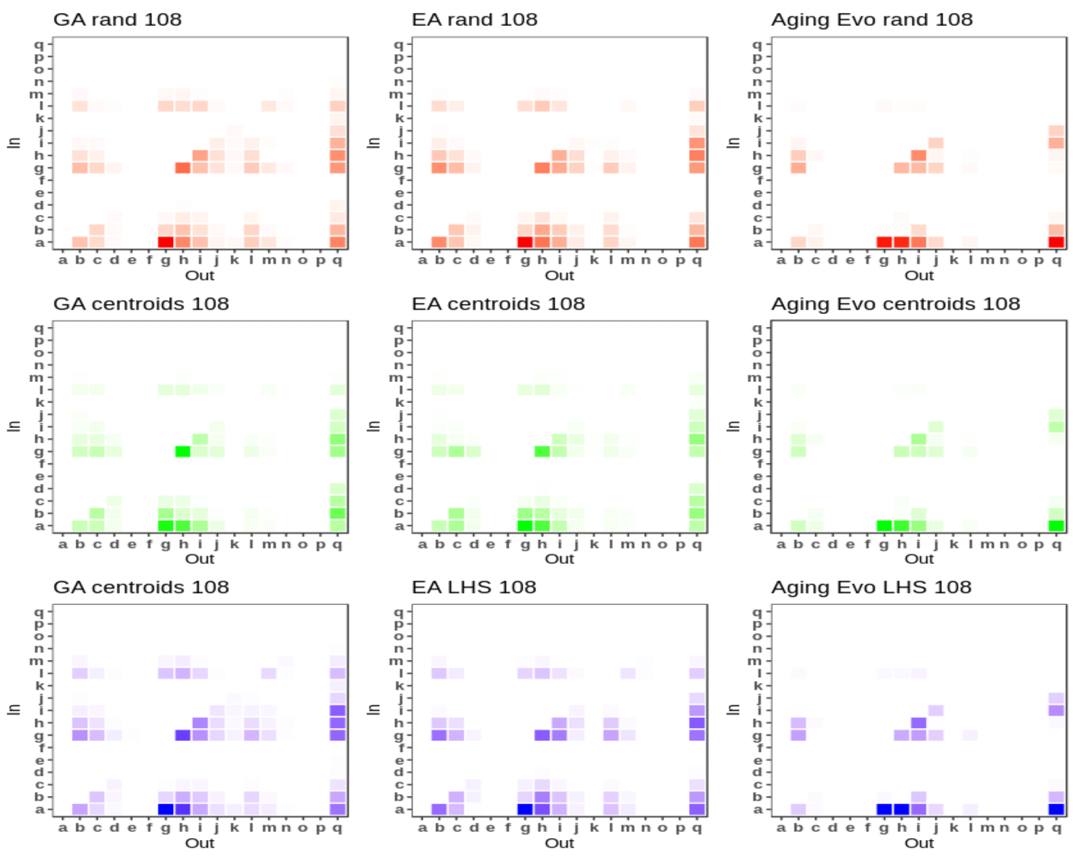
Table 5.2: Legend for understanding Figure 5.5 : a character used to label the axis of the adjacency matrix of a solution is associated with a pair of node label and index, appearing in the formal DAG representation of a solution.

5.1.3 Outlook

Next, we discuss future research directions and spin-off research products that could extend the current research project. First is regarding using other NAS Benchmark databases as input to the data-driven initialization method. This is when targeting classification instances. The idea is to explore the robustness of the *task-agnostic* initialization technique to using alternative data sources (other than `NASBench-101`),



(a) With a budget of 36 epochs



(b) With a budget of 108 epochs

Figure 5.5: Visualization of $N = 100$ solutions obtained with the benchmark[4].

for initializing population-based search strategies. In practice, the challenge would be providing suitable sampling and encoding functions to the alternative search spaces. In the case of model encoding, it would consist of designing a function mapping the initial model representation (adjacency matrix, list of operations, etc.) in the source database to the target binary vector format. Then, the general procedure described in Figure 5.1 and Algorithm 3 could be applied and benchmarked with the new data source.

Second, is regarding the initialization of other families of population-based search strategies. Indeed, we found that population-based *meta-heuristics* such as `GA` and `EA` can benefit from the proposed approach. In particular, `(`$\mu + \lambda$`) EA` takes advantage of very fit initial populations to achieve faster convergence and larger final performances than with sampling-based initialization baselines. We propose to explore the benefit of the data-driven approach for alternative population-based methods, such as those relying on *Reinforcement Learning* or even *Bayesian Optimization*. Once again, the proposal would benchmark various initialization baselines again the proposed one and identify the settings enabling those alternative families of search strategies to take advantage of the proposal.

5.2 Searching for Efficient and Compact Classification Models

Next, we describe an approach addressing the problem of finding high-performing but resource-efficient solutions for classification tasks in EO. The motivation behind this research is to answer the growing need for computationally efficient models that can run on resource-constrained environments such as satellite constellations. An example of use would be the processing of newly acquired remote sensing data and the early decision-making onboard as a component of the software stack embedded in the target resource-constrained hardware.

Peer-reviewed publication: Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang, *Compact Neural Architecture Search for Local Climate Zones Classification*, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2021.

Research questions tackled: *RQ3.*

Related Contributions: *C5.*

5.2.1 Complexity-aware Differentiable Neural Architecture Search

The proposed search strategy aims at finding an optimal neural architecture configuration for classification tasks in EO. The search strategy is based on a flavour of continuous bi-level optimization, known as *differentiable search*[35,36]. Figure 5.6 provides a visual description of the search space of classification models we search for: a model consists of a downsampling module (*head*), followed by a sequence of N units, combining a *block* and *reduction cell* (or Softmax). Each *block* i consists in a sequence of length δ_i of repeating a given element, the *normal cell*. This *normal cell*, which can be represented as `DAG` of operation processing features generated from an EO sensor, is variable. An architecture α is then identified by two elements: a neural *cell* configuration α_{cell}, and a vector Δ encoding the *depth* of all N blocks.

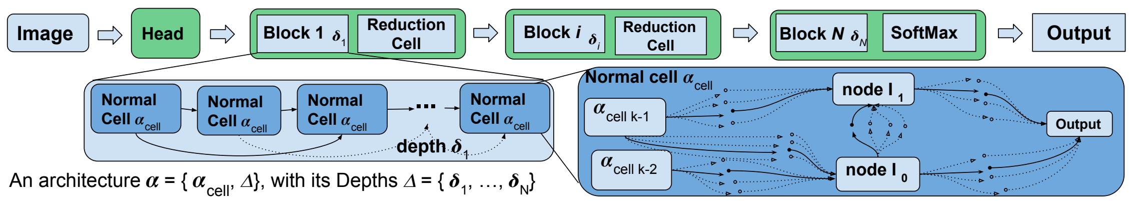


Image
Head
Block 1 δ_1
Reduction Cell
Block i δ_i
Reduction Cell
Block N δ_N
SoftMax
Output
Normal Cell α_{cell}
Normal Cell α_{cell}
Normal Cell α_{cell}
Normal Cell α_{cell}
depth δ_1
Normal cell α_{cell}
$\alpha_{cell\ k-1}$
$\alpha_{cell\ k-2}$
node I_1
node I_0
Output
An architecture $\alpha = \{\alpha_{cell}, \Delta\}$, with its Depths $\Delta = \{\delta_1, ..., \delta_N\}$

Figure 5.6: Visualization of an architecture α, optimized using Algorithm 4[5].

In practice, the search strategy aims at finding the optimal neural architecture by finding a tensor $\mathbb{A} = \{\mathrm{A}_{cell}\} \cup \{\Omega_{depth}\}$, which encodes the distributions of possible cells and depth configurations to sample from, and a set of trainable weights $\mathbb{W}$ of the model, providing the best performances in validation and training splits, respectively. Algorithm 4 provides a general description of the *gradient-based* search procedure. Once an architecture α is found, it is re-trained from scratch and evaluated on the test split of the data to assess its final performance.

Algorithm 4: Searching for α_{cell} and Δ[5]

```
Input: Two disjoint sets $\mathbb{D}_T$ and $\mathbb{D}_V$, randomly initialized $\mathbb{A}$ and $\mathbb{W}$, batch size;
while not at convergence do
    Sample a batch $\mathbb{D}_t = \{(x_i, y_i)\}_{i=1}^n$ from $\mathbb{D}_T$
    Calculate $L_T = \sum_{i=1}^n L(x_i, y_i)$
    Update $\mathbb{W}$ by gradient descent: $\mathbb{W} = \mathbb{W} - \nabla_{\mathbb{W}} L_T$
    Sample a batch $\mathbb{D}_v = \{(x_i, y_i)\}_{i=1}^n$ from $\mathbb{D}_V$
    Calculate $L_V = \sum_{i=1}^n L(x_i, y_i)$
    Update $\mathbb{A}$ by gradient descent: $\mathbb{A} = \mathbb{A} - \nabla_{\mathbb{A}} L_V$
end
Derive the final architecture $\alpha = \{\alpha_{cell}, \Delta\}$ from $\mathbb{A}$;
Optimize $\alpha = \{\alpha_{cell}, \Delta\}$ on the whole training set for future inference on the test
  set.
```

Method	ResNet08	ResNet32	ResNet110	S	S+C	S+D	S+C+D
OA (d1)	60.25	64.85	67.60	64.05	64.25	64.96	64.00
AA (d1)	47.86	52.24	53.34	50.39	51.63	54.02	52.20
Kappa (d1)	0.565	0.615	0.645	0.61	0.61	0.62	0.61
OA (d2)	96.18	96.89	98.59	98.15	97.81	98.00	97.74
AA (d2)	88.91	89.75	91.91	91.61	91.12	91.44	91.08
Kappa (d2)	0.958	0.966	0.984	0.980	0.975	0.980	0.975
Size (MB)	0.08	0.27	1.73	0.224	0.165	0.14	0.130
FLOPs (M)	13.53	41.85	253.89	30.37	24.10	20.05	18.42
Block's Depth	NA	NA	NA	3x3	3x3	1x3	1x3

Table 5.3: Benchmark[5] of model performance in test (5 seeds), on the classification instance So2Sat LCZ-42 (Sentinel-2), and two data distributions (d1 and d2).

5.2.2 Results

This section presents the main results obtained using the proposed approach. Table 5.3 presents a benchmark of the performance in test for the architecture solutions found using the instance So2Sat LCZ-42. In addition, it also compares them to state-of-the-art classification baselines of similar complexity `ResNet08`, `ResNet32`, and `ResNet110`. The complexity of the models is measured in size and `FLOPs`. The performance is assessed using the metrics of Overall Accuracy (`OA`), Average Accuracy (`AA`), and the Kappa Cohen score (`Kappa`). With respect to the solutions found, they are identified by the search setting from which they are resulting. Indeed, `S` refers to optimizing a *single* (normal) cell, `C` stands for the use of *complexity* reducing loss, and `D` for searching for the appropriate architecture *depth.* Then `'+'` stands for combining different search mechanisms.

We find that on the data distribution $d1$, `S+D` and `S+C` positively improve against only using `S`, considering all metrics of accuracy (`OA`, `AA` and `Kappa`). `S+D` brings the most improvements in performance while enabling to find a model of lower complexity in `size` and `FLOPs`. `S+D` is competitive, i.e. `TOP-2` in performance and complexity, and outperforms the manually designed `ResNet08` and `ResNet32` baselines of comparable model complexity. Combining `S+C+D` does improve the Average Accuracy over the setting `S` and `S+C`, but not over `S+D`.

When using the data distribution $d2$, we find that combining the complexity-aware mechanisms (`S+D`, `S+C`, `S+C+D`) does not improve over the conventional single cell search (`S`). However, `S+D` helps reach the best trade-off for performance and low complexity. Its performance scores are 98.00% OA, 91.44% AA, and 0.98 for the Kappa Cohen score, which is a respective difference of 0.58% OA, 0.47% AA, and 0.004 Kappa from `ResNet110`, the best performing model. Note that it does so while being about 92% more compact in `Size` and `FLOPs`.

To summarize, we have developed a search strategy that helps find *efficient* classification models of high-performance and compactness, competing with *popular* and *state-of-the-art manually* designed baselines.

5.2.3 Outlook

Moreover, we identify connections between the proposed approach and other scientific topics of relevance and importance to *AutoML*. First, the topic of *model pruning*[73] comes to mind. It encompasses all methods enabling the transform of an *overparametrized* model with good performances into a more *spare* model without loss in performance. Given that the proposed method can *tremendously* reduce the complexity in `Size` and `FLOPS` while improving or preserving model performances, we argue that it can be interpreted as a form of *model pruning* or *model sparsification.* Let us consider the particular case (`S+D`) of searching for a normal cell in combination with searching for an optimal architecture depth. We speculate that optimizing the blocks' depth might result in sparsification by reducing stages in an over-parametrized initial backbone architecture. To confirm this idea, future work might investigate the effectiveness of the search strategy in finding efficient but compact architectures based on the complexity of the starting solutions, the backbone architecture.

Besides, the topic of *data-driven search initialization* appears of relevance to the current research project. Indeed, we have identified in Section 5.1 that population-based search strategies could benefit from a data-driven selection technique for their initial population, helping them converge faster and reach better long-term performances. Despite not being tested in the case of non-population-based search strategies, future work might investigate how the use of benchmark data, in particular, the use of data related to distributions of cell configurations, could help bias the initialization of the architecture weight matrix $\mathbb{A}$ (tied to the configuration of the *normal cells* and *depth*), in order to achieve better convergence. For this purpose, a complementary data source could be the distributions of optimal solutions found in Section 5.1, Figure 5.5.

This page is intentionally left blank.

Chapter 6

Benchmark for AutoML in Earth Observation

This last chapter introduces contributions to the benchmarking of AutoML algorithms in EO applications. More precisely, the works conducted here deal with the creation of a platform aiming at making it easier to prototype and develop NAS algorithms, on an EO use-case.

Research questions tackled: RQ3.

Related contributions: *C6.*

6.1 Motivations

This project aims to provide a tool enabling researchers and practitioners of the EO community to research and develop AutoML methodologies with *ease*. According to the best practices and recommendations[38] in the AutoML community, NAS benchmarks should be reproducible, use standard solutions and components (search space, evaluation pipeline) and be open-source. For this purpose, we have prepared a database[74] in the form of NAS Benchmark providing freely available architecture performance evaluations for an EO application[75]. This database builds on top of previously published work, in particular for the NAS search scenario (*cell-based search space with a fixed classification backbone)*[40] and the concept of surrogate benchmarks[41].

6.2 Database

The database tackles the search scenario of NASBench-101[40], which consists in finding an optimal solution in the search space of the ResNet-like image classification architecture backbone. The backbone is fixed but contains an elementary unit, a.k.a. *cell*, in the form of a DAG of neural network features and operations, then repeating following a specific pattern (several *blocks*, or sequence fixed length of cells). We chose this search scenario because there is evidence[40] that the search space can express a variety of architectures in terms of configurations (includes *Inception-Net*), but also in terms of complexity (e.g. number of trainable parameters), and performance.

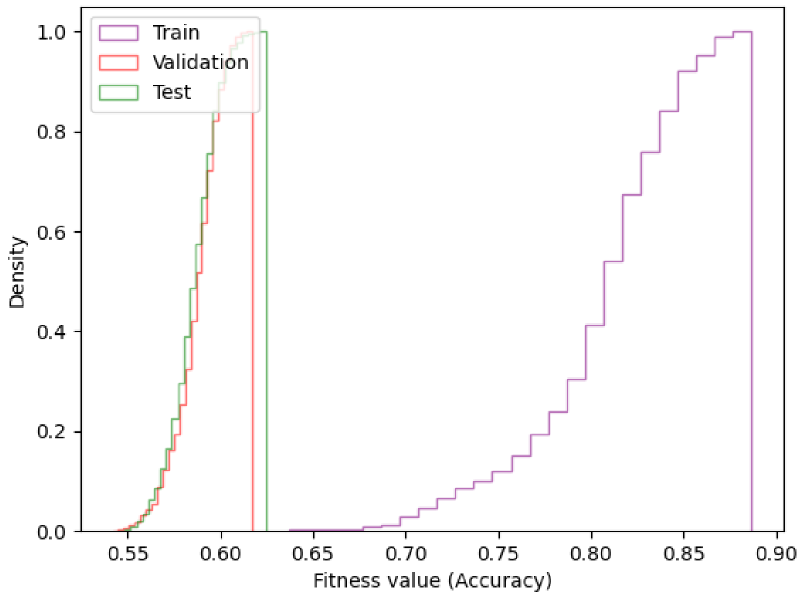


Figure 6.1: Cumulative distribution of fitness (overall accuracy) in training, validation, and test after 108 epochs of training on the NAS instance So2Sat LCZ-42.

We randomly sample $N = 1000$ architectures from the search space and evaluated them on the scene classification task So2Sat LCZ-42. Figure 6.1 displays the distribution of the models' fitness (overall accuracy) on the various splits of the So2Sat scene classification task. As observed in the publication associated with the classification instance[75], we find that differences in the distribution performance on the data splits align with the differences in geographical distribution behind the image observations. In other words, the performances on the validation and test split are similar since the observations in the splits are from the same cities. In contrast, the training splits originate from another set of cities with more diversity.

Besides, we also provide measurements for a model complexity-related metric, the latency of the models. Figure 6.2 displays the distribution of the latency of the search space, a.k.a. the time in milliseconds needed for the sampled solutions. We find that the behaviour of the models is diverse: it should be possible to discriminate models using this metric as an objective. We provide this value for researchers interested in multi-objective NAS for EO applications.

6.3 Outlook

Next, we discuss existing opportunities for extending this work. The current database prototype is a surrogate NAS benchmark, providing a mix of real and synthetic performance estimations for solutions in the search space of NASBench-101. The real fitness evaluations are made on the classification instance So2Sat LCZ-42 (Sentinel-2 sensor). Future work might explore the benchmarking of additional surrogate models[76] to better understand how this design choice might affect our NAS search settings in EO applications.

Besides, exploring additional EO classification instances (datasets) or additional

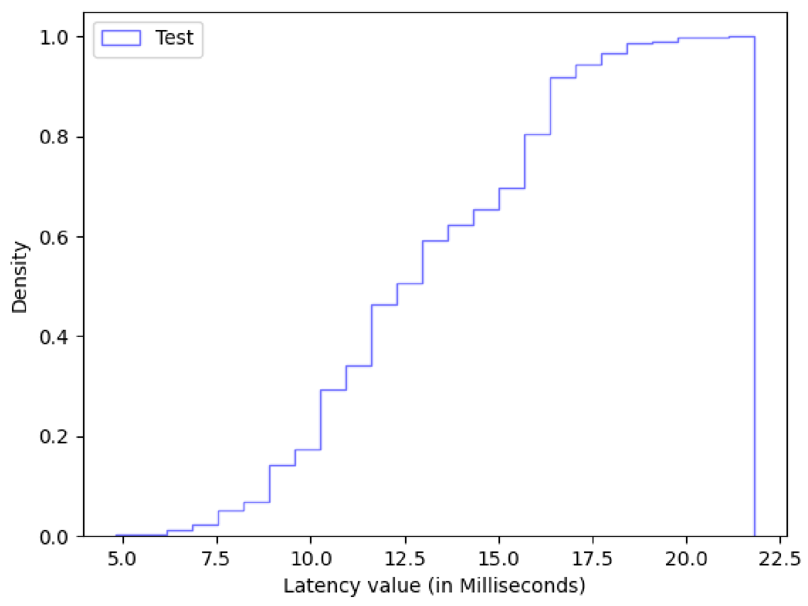


Figure 6.2: Cumulative distribution of the average latency (in Milliseconds) on the test split for the N=1000 models.

NAS Scenarios (search spaces) could also provide value. Indeed, to study the correlation and transferability of performance across instances (datasets) has been primarily studied at the scale of individual models. The availability of evaluations for additional instances in the NAS database could enable such study at a larger scale, the one of search spaces. Moreover, the availability of performance estimations on alternative search spaces, for given instances, could, on the other hand, compare the suitability of search spaces and their components to EO-specific applications. This could open the door to valuable research in the direction of search space engineering[77,78].

This page is intentionally left blank.

Chapter 7

Conclusion

The research conducted in this dissertation can be summarized as follows:

Landscape-aware AutoML: The aspect of *explainability* in AutoML systems is a desirable quality that is hard to achieve. Indeed, AutoML systems (e.g. NAS, HPO) aim at automating the design of decision-making algorithms for given tasks, using various components such as search spaces of desirable solutions, performance evaluation strategies, and optimizers. Given the complexity of the systems, identifying the contributions of components to a resulting solution is challenging. Moreover, *how do changes in the experimental setting of AutoML systems affect algorithm performances?* This research question is instrumental to the development of the works in this manuscript. To answer this, we have developed a framework that quantitatively assesses key features of the landscapes of an AutoML problem (e.g. ruggedness, multi-modality) and relate them to algorithmic performance. Besides, we have demonstrated the use of the tool in the following context: firstly, in characterizing the landscape of a NAS problem using an EO-specific classification instance; secondly, in the comparison of the characteristics of the landscapes of a NAS problem, in the context of various domains of application (EO and CV). This has led to the publication of *P1* (and P5). In the same vein, by analyzing AutoML problems and their performance evaluation strategies, we have discovered that the choice of the fitness evaluation metric could negatively affect the landscapes, making it harder to navigate for search strategies. This has led to the publication of *P4*. Regarding this chapter, we believe that the research conducted could be extended by investigating more complex AutoML scenarios for EO applications, such as NAS for the tasks of Object Detection, Semantic Segmentation, or even Change Detection. Indeed, these scenarios would require the study and design of dedicated search spaces or performance evaluation strategies and would constitute a great opportunity to develop the topic of landscape analysis for NAS in EO.

Efficient AutoML: Another important aspect of Data Science for EO is the notion of efficiency. Indeed, given the tremendous volume of data to process by data-driven methodologies, the ability to design solutions of low complexity or strategies to find such solutions given limited resources is crucial. In order to address these concerns, we have developed a method that helps AutoML search strategies (population-based NAS metaheuristics) converge faster and achieve better long-term performances (fitness of decision-making solutions found), with a data-driven initialization technique. We have demonstrated the effectiveness of the method using a CV application, as well as a successful transfer for initializing a search algorithm in an EO NAS instance. This work has led to the publication of *P2* (and P6). Moreover, we have proposed an NAS methodology that searches for low computational complexity decision-making solutions, constructed from a hybrid search space, with a modular backbone and variable elementary feed-forward units (normal cells). This enables to find solutions of competitive fitness, compared to manually designed and popular baseline classifiers of comparable complexity. This work led to the publication of *P3*. Future work might examine how to provide initialization strategies for population-based AutoML algorithms, beyond the case of metaheuristcs (e.g. Reinforcement Learning-based, Bayesian approaches), beyond the single objective of fitness (Multi-objective NAS), or beyond the scenario of classification (e.g. search spaces for Object Detection, etc.).

Accessible AutoML: The potential benefits of AutoML to the field of Remote Sensing are numerous. Indeed, from helping automate the tedious design of Machine Learning and Data Science pipelines, to discovering more optimal data-driven models (NAS) and their hyperparameters (HPO), the positive outcomes could affect a wide range of applications. However, AutoML is expensive to perform. For this reason, and in line with previous works in CV[39] and NLP[43], we have proposed a novel NAS benchmark to help enable free-of-cost and reproducible NAS research on an EO application. The prototype benchmark is in the form of a surrogate and tabular dataset of model evaluation, making use of an established search space for image classification. It uses a cell-based search space evaluated for the task of real-work EO scene classification, the So2Sat LCZ-42 dataset. It enables the development and benchmarking of single and multi-objective NAS search strategies. This work led to the publication of P7. Future work might explore extending the database to more EO instances (datasets), and scenarios (search spaces).

Bibliography

[1] Traoré, K. R.; Camero, A.; Zhu, X. X. Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems. *CoRR* **2021**, *abs/2111.01584*.

[2] Traoré, K. R.; Camero, A.; Zhu, X. X. Landscape of Neural Architecture Search Across Sensors: how much do they differ? *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **2022**, *V-3-2022*, 217–224.

[3] Traoré, K. R.; Camero, A.; Zhu, X. X. We Won't Get Fooled Again: When Performance Metric Malfunction Affects the Landscape of Hyperparameter Optimization Problems. Optimization and Learning. Cham, 2023; pp 148–160.

[4] Traoré, K. R.; Camero, A.; Zhu, X. X. A data-driven approach to neural architecture search initialization. *Annals of Mathematics and Artificial Intelligence* **2023**,

[5] Traoré, K. R.; Camero, A.; Zhu, X. X. Compact Neural Architecture Search for Local Climate Zones Classification. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* **2021**,

[6] Yuan, X.; Shi, J.; Gu, L. A review of deep learning methods for semantic segmentation of remote sensing imagery. *Expert Syst. Appl.* **2021**, *169*, 114417.

[7] Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015.

[8] Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2015**, *39*, 2481–2495.

[9] Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Cham, 2015; pp 234–241.

[10] Li, Z.; Wang, Y.; Zhang, N.; Zhang, Y.; Zhao, Z.; Xu, D.; Ben, G.; Gao, Y. Deep Learning-Based Object Detection Techniques for Remote Sensing Images: A Survey. *Remote Sensing* **2022**, *14*.

[11] Cheng, G.; Huang, Y.-M.; Li, X.; Lyu, S.; Xu, Z.; Zhao, Q.; Xiang, S. Change Detection Methods for Remote Sensing in the Last Decade: A Comprehensive Review. *ArXiv* **2023**, *abs/2305.05813*.

[12] Hutter, F., Kotthoff, L., Vanschoren, J., Eds. *Automated Machine Learning - Methods, Systems, Challenges*; Springer, 2019.

[13] Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.-L.; Deng, D.; Lindauer, M. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery* **2023**, *13*, e1484.

[14] Elsken, T.; Metzen, J. H.; Hutter, F. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* **2019**, *20*, 1–21.

[15] Brazdil, P.; van Rijn, J. N.; Soares, C.; Vanschoren, J. Metalearning. *Cognitive Technologies* **2022**,

[16] Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J. T.; Blum, M.; Hutter, F. Efficient and Robust Automated Machine Learning. NIPS. 2015.

[17] Thornton, C. J.; Hutter, F.; Hoos, H. H.; Leyton-Brown, K. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* **2012**,

[18] Zöller, M.-A.; Huber, M. F. Benchmark and Survey of Automated Machine Learning Frameworks. *Journal of Artificial Intelligence Research* **2019**, *70*, 409–472.

[19] Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-y.; Li, Z.; Chen, X.; Wang, X. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Comput. Surv.* **2021**, *54*.

[20] Chitty-Venkata, K. T.; Somani, A. K. Neural Architecture Search Survey: A Hardware Perspective. *ACM Computing Surveys* **2022**, *55*, 1 – 36.

[21] Malan, K. M. A Survey of Advances in Landscape Analysis for Optimisation. *Algorithms* **2021**, *14*, 40.

[22] Hernando, L.; Mendiburu, A.; Lozano, J. An Evaluation of Methods for Estimating the Number of Local Optima in Combinatorial Optimization Problems. *Evolutionary computation* **2012**, *21*.

[23] Matthew, R. C.; Mullin, M. Estimating the Number of Local Minima in Big, Nasty Search Spaces. In Proceedings of IJCAI-99 Workshop on Statistical Machine Learning for Large-Scale Optimization. 1999.

[24] Xiong, Z.; Zhang, F.; Wang, Y.; Shi, Y.; Zhu, X. X. EarthNets: Empowering AI in Earth Observation. *ArXiv* **2022**, *abs/2210.04936*.

[25] Xia, J.; Yokoya, N.; Adriano, B.; Broni-Bediako, C. OpenEarthMap: A Benchmark Dataset for Global High-Resolution Land Cover Mapping. *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* **2022**, 6243–6253.

[26] Toker, A.; Kondmann, L.; Weber, M.; Eisenberger, M.; Camero, A.; Hu, J.; Hoderlein, A. P.; Çaglar Senaras,; Davis, T.; Cremers, D.; Marchisio, G. B.; Zhu, X. X.; Leal-Taix'e, L. DynamicEarthNet: Daily Multi-Spectral Satellite Dataset for Semantic Change Segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* **2022**, 21126–21135.

[27] Kondmann, L.; Toker, A.; Rußwurm, M.; Camero, A.; Peressuti, D.; Milcinski, G.; Mathieu, P.-P.; Longépé, N.; Davis, T.; Marchisio, G. B.; Leal-Taixé, L.; Zhu, X. DENETHOR: The DynamicEarthNET dataset for Harmonized, inter-Operable, analysis-Ready, daily crop monitoring from space. NeurIPS Datasets and Benchmarks. 2021.

[28] Zhang, B.-T.; Mühlenbein, H. Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor. *Complex Syst.* **1993**, *7*.

[29] Dasgupta, D.; McGregor, D. R. Designing application-specific neural networks using the structured genetic algorithm. *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks* **1992**, 87–96.

[30] Stanley, K. O.; Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* **2002**, *10*, 99–127.

[31] Stanley, K. O.; Bryant, B. D.; Miikkulainen, R. Evolving Neural Network Agents in the NERO Video Game. 2005.

[32] Real, E.; Aggarwal, A.; Huang, Y.; Le, Q. V. Regularized Evolution for Image Classifier Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence* **2019**, *33*, 4780–4789.

[33] Zoph, B.; Le, Q. V. Neural Architecture Search with Reinforcement Learning. *ArXiv* **2016**, *abs/1611.01578*.

[34] White, C.; Neiswanger, W.; Savani, Y. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. AAAI Conference on Artificial Intelligence. 2019.

[35] Hanxiao, L.; Karen, S.; Yiming, Y. DARTS: Differentiable architecture search. *ICLR* **2019**,

[36] Dong, X.; Yang, Y. Searching for A Robust Neural Architecture in Four GPU Hours. *CVPR* **2019**,

[37] Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; Hutter, F. Understanding and robustifying differentiable architecture search. International Conference on Learning Representations (ICLR). 2020.

[38] Lindauer, M.; Hutter, F. Best Practices for Scientific Research on Neural Architecture Search. *Journal of Machine Learning Research* **2020**, *21*, 1–18, To appear.

[39] Chitty-Venkata, K. T.; Emani, M. K.; Vishwanath, V.; Somani, A. Neural Architecture Search Benchmarks: Insights and Survey. *IEEE Access* **2023**, *11*, 25217–25236.

[40] Ying, C.; Klein, A.; Christiansen, E.; Real, E.; Murphy, K.; Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. International Conference on Machine Learning. 2019; pp 7105–7114.

[41] Siems, J.; Zimmer, L.; Zela, A.; Lukasik, J.; Keuper, M.; Hutter, F. NAS-Bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777* **2020**,

[42] Mehta, Y.; White, C.; Zela, A.; Krishnakumar, A.; Zabergja, G.; Moradian, S.; Safari, M.; Yu, K.; Hutter, F. NAS-Bench-Suite: NAS Evaluation is (Now) Surprisingly Easy. International Conference on Learning Representations. 2022.

[43] Klyuchnikov, N.; Trofimov, I.; Artemova, E.; Salnikov, M.; Fedorov, M.; Filippov, A.; Burnaev, E. NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing. *IEEE Access* **2022**, *10*, 45736–45747.

[44] Dong, X.; Yang, Y. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. International Conference on Learning Representations (ICLR). 2020.

[45] Bansal, A.; Stoll, D.; Janowski, M.; Zela, A.; Hutter, F. JAHS-Bench-201: A Foundation For Research On Joint Architecture And Hyperparameter Search. Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track. 2022.

[46] Xie, S.; Kirillov, A.; Girshick, R.; He, K. Exploring Randomly Wired Neural Networks for Image Recognition. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). 2019; pp 1284–1293.

[47] Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; Dollár, P. Designing Network Design Spaces. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020; pp 10425–10433.

[48] Dollár, P.; Singh, M.; Girshick, R. B. Fast and Accurate Model Scaling. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* **2021**, 924–932.

[49] Yuan, J.; Liu, M.; Tian, F.; Liu, S. Visual Analysis of Neural Architecture Spaces for Summarizing Design Principles. *IEEE Transactions on Visualization and Computer Graphics* **2022**, *29*, 288–298.

[50] Ono, J. P.; Castelo, S.; Lopez, R.; Bertini, E.; Freire, J.; Silva, C. T. Pipeline-Profiler: A Visual Analytics Tool for the Exploration of AutoML Pipelines. *IEEE Transactions on Visualization and Computer Graphics* **2020**, *27*, 390–400.

[51] Yang, A.; Esperança, P. M.; Carlucci, F. M. NAS evaluation is frustratingly hard. International Conference on Learning Representations. 2020.

[52] Thomson, S. L.; Ochoa, G. On funnel depths and acceptance criteria in stochastic local search. *Proceedings of the Genetic and Evolutionary Computation Conference* **2022**,

[53] Pitzer, E.; Werth, B.; Karder, J. Dynamic Fitness Landscape Analysis. International Conference/Workshop on Computer Aided Systems Theory. 2022.

[54] Tong, H.; Minku, L. L.; Menzel, S.; Sendhoff, B.; Yao, X. What makes the dynamic capacitated Arc routing problem hard to solve: insights from fitness landscape analysis. *Proceedings of the Genetic and Evolutionary Computation Conference* **2022**,

[55] Ochoa, G.; Malan, K. M.; Blum, C. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Appl. Soft Comput.* **2021**, *109*, 107492.

[56] Ochoa, G.; Tomassini, M.; Vérel, S.; Darabos, C. A Study of NK Landscapes' Basins and Local Optima Networks. Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation. New York, NY, USA, 2008; p 555–562.

[57] Potgieter, I.; Cleghorn, C. W.; Bosman, A. S. A Local Optima Network Analysis of the Feedforward Neural Architecture Space. 2022 International Joint Conference on Neural Networks (IJCNN). 2022; pp 1–8.

[58] Mitchell, P.; Ochoa, G.; Lavinas, Y.; Chassagne, R. Local Optima Networks for Assisted Seismic History Matching Problems. EvoApplications@EvoStar. 2023.

[59] Shirakawa, S.; Nagao, T. Bag of local landscape features for fitness landscape analysis. *Soft Computing* **2016**, *20*, 3787–3802.

[60] Liefooghe, A.; Daolio, F.; Verel, S.; Derbel, B.; Aguirre, H.; Tanaka, K. Landscape-Aware Performance Prediction for Evolutionary Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation* **2020**, *24*, 1063–1077.

[61] Pimenta, C. G.; de Sá, A. G. C.; Ochoa, G.; Pappa, G. L. Fitness Landscape Analysis of Automated Machine Learning Search Spaces. Evolutionary Computation in Combinatorial Optimization. Cham, 2020; pp 114–130.

[62] Rodrigues, N. M.; Malan, K. M.; Ochoa, G.; Vanneschi, L.; Silva, S. Fitness landscape analysis of convolutional neural network architectures for image classification. *Inf. Sci.* **2022**, *609*, 711–726.

[63] Nunes, M.; Fraga, P. M.; Pappa, G. L. Fitness Landscape Analysis of Graph Neural Network Architecture Search Spaces. Proceedings of the Genetic and Evolutionary Computation Conference. New York, NY, USA, 2021; p 876–884.

[64] Hong, D.; Gao, L.; Yokoya, N.; Yao, J.; Chanussot, J.; Du, Q.; Zhang, B. More Diverse Means Better: Multimodal Deep Learning Meets Remote-Sensing Imagery Classification. *IEEE Transactions on Geoscience and Remote Sensing* **2021**, *59*, 4340–4354.

[65] Doda, S.; Wang, Y.; Kahl, M.; Hoffmann, E. J.; Ouan, K.; Taubenböck, H.; Zhu, X. X. So2Sat POP - A Curated Benchmark Data Set for Population Estimation from Space on a Continental Scale. *Scientific Data* **2022**, *9*, 715, Number: 1 Publisher: Nature Publishing Group.

[66] Pimenta, C. G.; de Sá, A. G. C.; Ochoa, G.; Pappa, G. L. Fitness Landscape Analysis of Automated Machine Learning Search Spaces. EvoCOP. 2020.

[67] Nunes, M.; Fraga, P. M.; Pappa, G. L. Fitness landscape analysis of graph neural network architecture search spaces. *Proceedings of the Genetic and Evolutionary Computation Conference* **2021**,

[68] Clergue, M.; Verel, S.; Formenti, E. An Iterated Local Search to find many solutions of the 6-states Firing Squad Synchronization Problem. *Applied Soft Computing* **2018**, *66*, 449–461.

[69] Sharma, A.; van Rijn, J. N.; Hutter, F.; Müller, A. Hyperparameter Importance for Image Classification by Residual Neural Networks. Discovery Science. Cham, 2019; pp 112–126.

[70] Pfisterer, F.; Schneider, L.; Moosbauer, J.; Binder, M.; Bischl, B. YAHPO Gym - An Efficient Multi-Objective Multi-Fidelity Benchmark for Hyperparameter Optimization. Proceedings of the First International Conference on Automated Machine Learning. 2022; pp 3/1–39.

[71] Eggensperger, K.; Müller, P.; Mallik, N.; Feurer, M.; Sass, R.; Klein, A.; Awad, N.; Lindauer, M.; Hutter, F. HPOBench: A Collection of Reproducible Multi-Fidelity Benchmark Problems for HPO. Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. 2021.

[72] Hirose, Y.; Yoshinari, N.; Shirakawa, S. NAS-HPO-Bench-II: A Benchmark Dataset on Joint Optimization of Convolutional Neural Network Architecture and Training Hyperparameters. Proceedings of The 13th Asian Conference on Machine Learning. 2021; pp 1349–1364.

[73] Hoefler, T.; Alistarh, D.; Ben-Nun, T.; Dryden, N.; Peste, A. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.* **2021**, *22*, 241:1–241:124.

[74] Demir, E.; Traoré, K. R.; Camero, A. Leveraging performance-based metadata for designing multi-objective NAS strategies for efficient models in Earth Observation. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* **2024**,

[75] Zhu, X. X. *et al.* So2Sat LCZ42: A Benchmark Data Set for the Classification of Global Local Climate Zones [Software and Data Sets]. *IEEE Geoscience and Remote Sensing Magazine* **2020**, *8*, 76–89.

[76] White, C.; Zela, A.; Ru, R.; Liu, Y.; Hutter, F. How Powerful are Performance Predictors in Neural Architecture Search? Advances in Neural Information Processing Systems. 2021; pp 28454–28469.

[77] Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; Dollár, P. Designing Network Design Spaces. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020; pp 10425–10433.

[78] Dollár, P.; Singh, M.; Girshick, R. Fast and Accurate Model Scaling. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021; pp 924–932.

This page is intentionally left blank.

Appendix A

Problems and Datasets

This section is dedicated to describing the datasets used for the various projects of the dissertation.

A.1 So2Sat LCZ-42

The So2Sat LCZ-42[75] is a database of multi-modal multi-spectral imagery serving as a benchmark for the task of LCZ classification. It consists of 400k pair of Sentinel-1 and Sentinel-2 images, and their LCZ labels. Each image covers 320 square meters on the ground and is recorded at a 10-meter resolution. Thus, they are visualized as 32 by 32-pixel images, with variable numbers of channels. In total, 17 labels are considered for the labeling, one for each LCZ.

Moreover, the dataset gathers images from 42 cities, aiming to help train global climate zone classifiers. Indeed 42 cities across the 5 continents are selected to collect the images constituting the training set. Then, 10 more cities are selected to create the validation and test sets.

A.2 NASBench-101

The NASBench-101[40] is a database of neural network models serving as a bench-marking platform for AutoML search algorithms. It contains 423k distinct model architectures labelled with their performances, after their training and evaluation on the task of image classification, CIFAR-10. The dataset is generated using a micro search space of image classifiers, where a backbone architecture resembling ResNet is fixed, but its building block is modified. This elementary building block, referred to as *cell*, is a DAG with constraints on the number of edges and nodes allowed. In practice, a *cell* represents a feed-forward module capable of learning advanced image features, an ability needed for building large classifiers. With this large amount of adjacent cells in the search space, NASBench-101 enables the bench-marking of single-objective or multi-objective search algorithms, optimizing the configuration of cells in a classification architecture, for reaching high classification performances.

A.3 YaHPO-Gym

YAHPO Gym[70] is a database of HP configurations dedicated to the bench-marking of HPO optimizers. It consists of HP configurations in 14 different scenarios (XG-Boost, SVM, ResNet, etc) evaluated on up to 700 instances of classification or regression. For each HP configuration, data regarding performances has been recorded using various metrics, after training and evaluating a given scenario (model) and instances (task). Besides, YAHPO Gym is a multi-fidelity surrogate HPO benchmark: it contains a mix of real and synthetic performance measurements, demonstrating in practice the reliability of surrogates in HPO problems.

A.4 DS-2019

DS-2019[69] is also a database of evaluations of HP configurations, designed for benchmarking HPO Algorithms. It consist of N=2000 evaluations of HP configurations of a ResNet classifier, obtained for 10 popular instances of image classification. The HP configuration space contains 12 variables affecting the stochastic gradient descent training algorithm, the training regularization protocol, as well as the method for early stopping of the ResNet classifier.

This page is intentionally left blank.

Appendix B

List of Publications

This section presents the publications related to the work introduced in this dissertation.

B.1 Publications in Journals

- Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang, *Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems*, Under review at Journal of Machine Learning Research (JMLR), 2023

- Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2023), *A Data-driven Approach to Neural Architecture Search Initialization*, Annals of Mathematics and Artificial Intelligence, pp. 1-28. Springer Nature, doi: 10.1007/s10472-022-09823-0. ISSN 1012-2443.

B.2 Publications in the Proceedings of Conferences

- Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2021), *Compact Neural Architecture Search for Local Climate Zones Classification*, In: 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 393-398. The 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 06. - 08. Oct. 2021, Online. doi: 10.14428/esann/2021.ES2021-55. ISBN ISBN 978287587082-7

- Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2023), We Won't Get Fooled Again: When Performance Metric Malfunction Affects the Landscape of Hyperparameter Optimization Problems, In: 6th International Conference on Optimization and Learning, OLA 2023, 1824, pp. 148-160. Springer, Cham, International Conference on Optimization and Learning, OLA 2023, 3-5 May 2023, Malaga, Spain, doi: 10.1007/978-3-031-34020-8_11. ISBN 978-303134019-2. ISSN 1865-0929

B.3 Other publications not included in the cumulative dissertation

- Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2022) *Landscape of Neural Architecture Search across sensors: how much do they differ?*, In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 3-2022, pp. 217-224, XXIV ISPRS Congress, 6.-11. June 2022, Nice, France, doi: 10.5194/isprs-annals-V-3-2022-217-2022. ISSN 2194-9042

- Traoré, Kalifou René and Camero, Andrés and Zhu, Xiao Xiang (2021), *Lessons from Clustering of a Search Space: a Data-driven initialization technique to Search*, Workshop on Data Science meets Optimization, International Joint Conferences on Artificial Intelligence (IJCAI), Online

- Demir, Emre and Traoré, Kalifou René and Camero, Andrés (2024), *Leveraging performance-based metadata for designing multi-objective NAS strategies for efficient models in Earth Observation*, In: The 32nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 09. - 11. Oct. 2024, Brugges, Belgium, doi: 10.14428/esann/2024.ES2024-94, ISBN 978-2-87587-090-2.

This page is intentionally left blank.

Journal of Machine Learning Research N/A (N/A) 1-32 Submitted N/A; Revised N/A; Published N/A

Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems

Kalifou René Traoré KALIFOU.TRAORE@DLR.DE
Data Science in Earth Observation
Technical University of Munich
Arcisstrasse 21, Munich, 80333, Germany
&
Remote Sensing Institute
German Aerospace Center (DLR)
Münchener Strasse 20, Weßling, 82234, Germany

Andrés Camero ANDRES.CAMEROUNZUETA@DLR.DE
Remote Sensing Institute
German Aerospace Center (DLR)
Münchener Strasse 20, Weßling, 82234, Germany
&
Helmholtz AI, Germany

Xiao Xiang Zhu XIAOXIANG.ZHU@TUM.DE
Data Science in Earth Observation
Technical University of Munich
Arcisstrasse 21, Munich, 80333, Germany

Editor: N/A

Abstract

Neural architecture search is a promising area of research dedicated to automating the design of neural network models. This field is rapidly growing, with a surge of methodologies ranging from *Bayesian optimization*, *neuroevoltion*, to *differentiable search*. However, despite all great advances, few studies have presented insights on the difficulty of the problem itself, thus the success (or failure) of these methodologies remains unexplained. In this sense, the field of *optimization* has developed methods that *highlight* key aspects to describe optimization problems. The *fitness landscape analysis* stands out when it comes to characterizing reliably and quantitatively search algorithms. In this paper, we propose to use *fitness landscape analysis* to study a *neural architecture search* problem. Particularly, we introduce the *fitness landscape footprint*, an aggregation of eight (8) general-purpose metrics to synthesize the landscape of an architecture search problem. We studied two problems, the image classification benchmark CIFAR-10, and the Remote-Sensing problem So2Sat LCZ42. The results present a quantitative appraisal of the problems, allowing to characterize the relative difficulty and other characteristics, such as the *ruggedness* or the *persistence*, that helps to tailor a search strategy to the problem. Also, the *footprint* is a tool that enables the comparison of multiple problems.

Keywords: Neural Architecture Search, Fitness Landscape Analysis

1. Introduction

Neural architecture search (NAS) has seen lots of advances in recent years (Ojha et al. (2017); Elsken et al. (2019); Ren et al. (2020)). State-of-the-art techniques have brought attention to *evolutionary algorithms* (Esteban et al. (2019)) as well as to *differentiable search* strategies (Liu et al. (2018); Dong and Yang (2019)) making NAS increasingly faster. Moreover, lots of efforts have been made to improve the reproducibility of research in the field, by proposing benchmarks (eg., Ying et al. (2019); Dong and Yang (2020); Siems et al. (2020)) and guidelines (Lindauer and Hutter (2020); Traoré et al. (2021)).

However, despite all improvements made so far, limited insights have been presented explaining the success of state-of-the-art strategies or the difficulty of performing the search itself on a given task (Yang et al. (2020)). Surprisingly given enough budget it has been proven that simple baselines such as *local search* are still top performers (White et al. (2020)). This holds for various search spaces and datasets in *computer vision* (CV) (Lindauer and Hutter (2020)).

On the other hand, the *optimization* field has explored and provided tools to address the issue of characterizing optimization problems. Among these, the *fitness landscape analysis* (FLA) (Watson (2010); Pitzer and Affenzeller (2012)), gives an intuitive idea of where (and how) the search is done and can be improved. It has been used to study problems and to benchmark algorithms in applications ranging from the field of physics, biology, to chemistry or pure mathematical optimization problems (Reidys and Stadler (2002)).

This study proposes to fill the gap by using FLA to quantitatively characterize the difficulty of performing NAS on a given search space and setting. Particularly, we propose to characterize the landscape of a NAS problem ($\mathcal{F}$) by relying on aspects such as the *density of fitness*, *fitness distance correlation (FDC)*, *ruggedness*, and *persistence*. Therefore, the main contributions of this study are:

- We introduce the *fitness landscape footprint*, a framework to characterize the landscape of a NAS problem ($\mathcal{F}$). In practice, this framework helps compare landscapes associated to various search spaces, fitness functions, neighborhood operators or even datasets.

- Using the *fitness landscape footprint*, we studied two image classification datasets, the classical CIFAR-10 (Krizhevsky (2012)) as well as the real-world remote sensing problem, So2Sat LCZ42 (Zhu et al. (2020)). Among the findings, we identify several clues indicating that NAS could be performed at shorter regimen (36 epochs), finding elite models early. Other findings show a common signature of fitness of the search space on both datasets and the visualization of landscape of both problems for various training settings.

The remainder of this paper is organized as follows. The next section briefly introduces the related work. Section 3 presents our proposal. Section 4 outlines the experimental setup. Section 5 presents the results. Section 7 summarizes the conclusions and proposes future work.

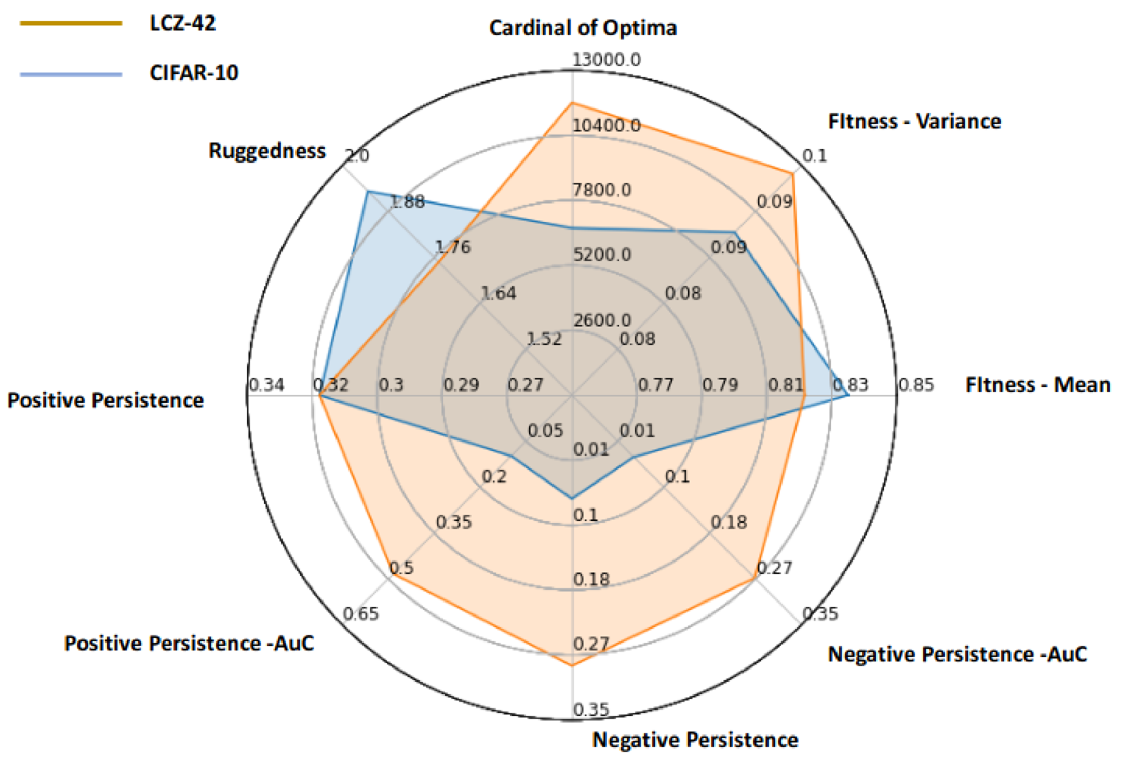


Figure 1: *Fitness landscape footprint* of samples trained for 36 epochs on CIFAR-10 and So2Sat LCZ42.

2. Related Work

This section introduces the related work in NAS as well as in fitness landscape analysis.

2.1 Neural Architecture Search (NAS)

In the late 1980s, researchers started exploring the use of *evolutionary computation* to design and train neural networks, a.k.a., *neuroevolution* (eg., Engel (1988); Montana and Davis (1989); Alba et al. (1993a,b); Yao (1993)). Neuroevolution gained popularity in the 2000s thanks to the *NeuroEvolution of Augmenting Topologies* (NEAT) method (Stanley and Miikkulainen (2002)). NEAT is a *genetic algorithm* (GA) that increasingly evolves complex neural network topologies (structures) and weights. More recently, Garciarena et al. (2021) take advantage of information obtained during deployment of neuroevolutionary NAS algorithm, to inform future runs of search. This information is to be stored in a Bayesian network-based metamodel and used in the case of search of *generative adversarial networks* (GAN) architectures in CV. The metamodel is shown to successfully guide a strategy based on *local search*, to find competitive models on the task of generating images from the MNIST image benchmark.

With the rise of *deep learning*, there has been a resurgence of these *old* methods in recent years to tackle the complex task of neural network design (Ojha et al. (2017)). For example, some authors proposed to used *harmony search* (Rosa et al. (2015)), GA (Zhining and Yunming (2015)), and *mixed integer parallel efficient global optimization* technique (van Stein et al. (2019)) to design *convolutional neural networks* (CNNs). Related to *recurrent neural networks* (RNNs), Bayesian optimization (Camero et al. (2020); Springenberg et al. (2016)), neuroevolution (Ororbia et al. (2019)), and hybrid approaches (Camero et al. (2019)), has been explored. Also, NEAT has been extended to fulfill the current needs of CNN (Miikkulainen et al. (2019)) and RNN (Rawal and Miikkulainen (2016)).

On the other hand, recent methods such as DARTS (Liu et al. (2018)) take advantage of *continuous optimization* to make search over a large graph of overlapping configurations (*Super-Net*) differentiable. These recent improvement result in large speedups in terms of search time (Elsken et al. (2019)) but also in some case in a lack of robustness and interpretability (Yang et al. (2020)).

In an attempt to make NAS more reproducible and accessible to the research community, additional efforts has been directed in providing open-source benchmarks of neural network configurations and their evaluations. These have been spanning several areas of applied *machine learning,* (ML) such as CV (eg., Ying et al. (2019); Dong and Yang (2020); Zela et al. (2020); Siems et al. (2020)) or even *natural language processing* (Klyuchnikov et al. (2020)). Given such available resources, the barrier for prototyping as well as better understanding NAS are progressively lowered.

2.2 Fitness Landscape Analysis

When it comes to the study of optimization problems, *fitness landscape analysis* (FLA) (Pitzer and Affenzeller (2012)) has been long used to characterize optimization problems, with applications ranging from *Physics*, *Biology*, to *Chemistry* or pure *Mathematical optimization* (Reidys and Stadler (2002)). At the conceptual level, FLA leverages knowledge of underlying structures of the tackled problems, while using general-purpose features to characterize them. One of the aims of FLA is to help predict performances and tuning algorithms, when solving an optimization problem. It does so by carefully considering appropriate descriptions of a fitness function, a search space and transition operators (Stadler (2002); Hernando et al. (2012); Matthew and Mullin (1999)) to model a problem. The use of descriptive features of the extracted landscapes include analysis of *density of states*, *enumeration of optima* (Hernando et al. (2012); Matthew and Mullin (1999)) and *neutral sets* or *basins*. Another example of a general-purpose FLA feature is the *fitness distance correlation* (Jones and Forrest (1995)), first applied to study the performances of a genetic algorithm on a white-box optimization problem. It aims at describing the hardness of a problem, using limited knowledge of an algorithm to be used. Its authors discuss a limitation in describing unknown functions to optimize for, i.e., in the case of black-box settings.

FLA has been successfully applied to predict performance (or performance enhancement). The work of Daolio et al. (2012) investigates the use of *local optima networks* (LON) to model fitness landscapes of combinatorial problems using a graph. It studies the relationships between LON features and the performance of meta-heuristics such as *iterated local search* algorithms. Their results show that some LON features can be used for the prediction of the running time to success. In Clergue et al. (2018), a general-purpose landscape-aware *hill-climbing iterated local search* strategy is introduced to solve the cellular automata problem of the *6-states firing squad synchronization problem* (FSSP). In particular, an acceptance criterion (*neutral hill climbing* or *netcrawler*) based on the neutrality of the landscape is proposed, to prevent the search algorithm from being stuck in search spaces with large plateaus.

The *value constraint satisfaction problem* (VCSP) framework is introduced in Kaznatcheev et al. (2020) to represent fitness landscapes. The work aims at finding classes of fitness landscapes enabling *local search* strategies to be tractable, i.e., solvable in polynomial time. This is done by identifying useful properties of VCs to be used, such as tree-structured graphs and boolean VCSP settings. Ochoa et al. (2008) is among the first topological and statistical analyses providing a characterization of combinatorial landscapes. Its authors use the concept of inherent networks, borrowed from statistical physics, to better represent landscapes graphically. They focus on the case of NK-landscapes and analyze its attributes such as basins of attractions, their size, and clustering coefficients.

Also, FLA has been used to characterize *multi-objective optimization* (MO) problems. For instance, in Liefooghe et al. (2020) it is proposed to predict the performance of *evolutionary multi-objective* (EMO) search algorithms on blac-box functions. The authors highlight that the use of manually designed features to analyze EMO performances is a common practice, and propose additional general-purpose features (*ruggedness* and *multi-modality*) to tackle some black-box MO problems. In Essiet and Sun (2020), the focus is put on *dynamical multi-objective optimization* problems (DMOPs) with a time-dependent MO fitness landscape. They propose a landscape-aware dynamical version of a classical algorithm (NSGA-III), using adaptive mutation and recombination operators in mating. The aim is to keep track of the moving MO fitness landscape, i.e., the Pareto front.

Recently, some authors have study ML problems using FLA. For example, Bosman et al. (2017) addresses the case of complexity reduction of neural networks using cost penalty regularization for weight elimination, by studying the change in the error landscape of the regularized neural networks. Particularly, the study proposes hyperparameter changes to the regularization, to make the error landscape more maneuverable to the back-propagation algorithm used in training. In Rodrigues et al. (2020), they study the generalization ability of CNNs on various ML problems. A strong emphasis is made to provide insight on tuning *neuroevolution* algorithms to fit the tasks.

When it comes to the particular topic of NAS, the exploitation of FLA is nascent. In Pimenta et al. (2020), authors propose to represent an *AutoML* pipeline in the framework on FLA, and analyze its heterogeneous search space using measures of FDC and neutrality ratio. Nunes et al. (2021) is among the first works to use FLA features to characterize the search space explored by NAS methods for automatically designing *graph neural networks* (GNNs). They use established metrics (*fitness distance correlation*, *dispersion*) to characterize the difficulty for NAS on several GNNs benchmarks.

3. Fitness Landscape Footprint

This section introduces the *fitness landscape footprint.* First, the fitness landscape $\mathcal{F}$ is defined. Then, we introduce general-purpose features for FLA, including the *fitness distance correlation*, *local optima* and the *ruggedness.* Next, we propose the concept of *persistence* to characterize over time the behavior of sampled configurations. Last but not least, we introduce the *fitness footprint* framework to describe and compare NAS optimization problems in a simple manner.

3.1 Fitness Landscape

A *fitness landscape* $\mathcal{F}$ is a framework to help study any optimization problem defined by a *search space* Ω, a *measurement of fitness* f for samples in Ω, and a *neighborhood operator* N to navigate Ω. It is defined as the triplet combination $\mathcal{F} = (\Omega, f, N)$ (Stadler (2002)).

In a landscape, the fitness function f assigns a fitness value to every configuration x in the search space Ω:

$$\begin{aligned} f : \Omega &\longrightarrow \mathrm{R} \\ x &\longmapsto f(x) \end{aligned} \tag{1}$$

In the context of NAS, we consider f as being the evaluation of performance for a neural network configuration $x \in \Omega$ after a training regime of length t_i.

In order to provide a structure to the fitness landscape, one needs to think of a way to arrange the configurations of the search space Ω, and how they can be reached from one another. This is the role of the neighborhood operator N which assigns to each solution x in the search space Ω a set of neighbors $N(x) \in P(\Omega)$. For our NAS use case, we consider the *Hamming distance* as the metric to define a neighborhood N as shown in Equation 2:

$$N(x) = \{y \in \Omega \mid d_{hamming}(x, y) = 1\} \tag{2}$$

where for each configuration $x \in \Omega$, its neighbors N comprise all configurations $y \in \Omega$ distant of a Hamming unit (1) to x.

3.2 Fitness Distance Correlation (FDC)

The *fitness distance correlation* (Jones and Forrest (1995); Pitzer and Affenzeller (2012)) is a classical FLA concept used to characterize the hardness of optimization problems. Originally used as a general-purpose score to study fitness landscapes, it can also be visually assessed as the fitness versus distance to a global optimum x^*, for all solutions $y \in \Omega$. Equation 3 introduces the concept:

$$FDC(x^*, \Omega, f) = \{(d_{hamming}(x^*, y), f(y)), \forall y \in \Omega\} \tag{3}$$

where x^* is the global optimum, Ω is the search space, f is a fitness function, and $d_{hamming}$ is the *hamming distance*.

3.3 Ruggedness

In the context of NAS, a *random walk* represents a path of consecutive random steps in the space of neural networks representations, where for each pair of consecutive solutions $d_{hamming}(x^i, x^{i+1}) = 1$. For instance, when configurations are represented by a binary vector, a step of random walk consist in the action of randomly flipping a bit in the vector, resulting in a new state.

In this context, an additional feature to characterize the difficulty of optimizing over a fitness landscape $\mathcal{F}$ is to measure its *ruggedness* (Stadler (2002)). A common metric of ruggedness is the autocorrelation ρ (serial correlation) on a series of fitness values f for configurations in a random walk $\mathrm{W} = \{x^0, ..., x^j, ..., x^n\}$ in Ω. Once ρ is derived, the final ruggedness τ is the inverse of the autocorrelation for all consecutive samples ($k = 1$ lags): $\tau = \frac{1}{\rho(1)}$. Equation 4 introduces the formula of the autocorrelation function ρ:

$$\rho(k) = \frac{\mathrm{E}[(f(x_i) - \bar{f})(f(x_{i+k}) - \bar{f})]}{Var(f(x_i))}, \forall i \in \mid \mathrm{W} \mid \tag{4}$$

In NAS, measuring the ruggedness of $\mathcal{F}$ can be interpreted as the variability in fitness f one can expect from a local search baseline algorithm in Ω.

3.4 Local Optima

Combinatorial optimization problems often aim at finding solutions either maximizing or minimizing a cost function f, in the case of single-objective problems (Pitzer and Affenzeller (2012)). Therefore, looking for such optimal configurations should not be diverted by the existence of locally optimal solutions.

Considering a fitness function f, and a neighborhood structure N, a configuration x^* in search space Ω is a local optimum (minimum) if its fitness is lower than any of its neighbors, i.e., $f(x^*) \leq f(y)$, $\forall y \in N(x^*)$. In this case, x^* is a *local minimum.* In the case of a fitness lower than any other solution in the search space, i.e., $f(x^*) \leq f(y)$, $\forall y \in \Omega$, then x^* is the *global minimum.* The same can be defined for *local maximum* and *global maximum.*

Moreover, a way to measure the difficulty of a fitness landscape $\mathcal{F}$ is to enumerate the existing local optima in Ω. On the individual basis, a local optimum can be retrieved using a local search procedure (Hernando et al. (2012)), e.g., a *best-improvement local search* (BILS) for a local maximum. The algorithm 1 describes the procedure of BILS.

Algorithm 1: Best-improvement local-search (BILS)

```
Choose an initial solution $x \in \Omega$;
repeat
    $x^* = x$;
    for $i = 1$ to $\mid N(x^*) \mid$ do
        Choose $y_i \in N(x^*)$;
        if $f(y_i) < f(x)$ then
            $x = y_i$;
        end
    end
until $x = x^*$;
```

Several methodologies have been defined to estimate the number of optima in Ω (Hernando et al. (2012)). One of the simplest, and least computationally expensive, is the *birthday problem* (Matthew and Mullin (1999)). As described in Algorithm 2, the procedure consist in an average estimation over T trials. For each trial i, we collect M local optima by applying BILS from M distinct and randomly selected starting points. Then, we measure k_i, the number of distinct local optima, until the first duplication out M samples at trial i. Then, we derive k_{mean} as the average rank of first duplicate for all T trials.

Algorithm 2: Analytics of the birthday problem

```
Let $T$ the total number of trials of enumeration;
for $i = 1$ to $T$ do
    Choose $M$ distinct random starting points in $\Omega$;
    Iteratively collect the $M$ Local Optima using BILS;
    Let $k_i$ the number of Optima at first duplication;
end
Let $k_{mean}$ the average number of Optima at duplication;
Derive $N$ the number of Optima using $k_{mean}$ and Eq. 5;
```

Equation 5 describes how to obtain the final estimation of number of optima N:

$$N \approx \frac{k_{mean}^2}{-2 * ln(1 - P_D)} \tag{5}$$

where $P_D = 0.5$ is the fixed probability of duplicated local optima, and k_{mean} the average number of (different) local optima found until the first duplication. In other words, if k_{mean} configurations are observed on average, with a chance $P_D = 0.5$ that two or more of them share the same rank, then the number of local optima approximates to N. It is analogous to the original *birthday problem*, where one tries to estimate the number k_{mean} of persons that can fit in a room until two or more share the same birthday (given a fixed chance of duplicates $P_D = 0.5$ and $N = 365$ days in a year). In our case, N is unknown, so we are tackling the *reverse birthday problem* (a.k.a., the *estimation of the martian year length*).

3.5 Persistence

NAS aims to find neural network configurations maximizing fitness f on unseen data (test sets) after a given budget of training time t_i. While performances are usually only considered at the end of the training time t_i, we propose to study configurations based on how their fitness evolve thought training time, given a series of fitness measurement $\{f(t_0), ..., f(t_i), ..., f(t_n)\}$ and respectively increasing training budget $\{t_0, ..., t_i, ..., t_n\}$.

Let us consider a ranking function $Ranking(N, t_i)$, that retrieves all models $y \in \Omega$, with respect to a rank N, based on $f(y)$ at training time t_i; and an ordered series of training duration $R_{all} = \{t_0, ..., t_i, ..., t_n\}$.

Then, we define *persistence* Π as the probability for model configurations ranked by function $Ranking(\cdot)$ to keep their initial rank (i.e., the one observed at t_0) through R_{all}. Equation 6 outlines Π:

$$\Pi(Ranking(\cdot), N) = \frac{P(\cap_{t_i \in \mathrm{R_{all}}} Ranking(N, t_i)}{P(Ranking(N, t_0))} \tag{6}$$

Particularly, we consider $Ranking(\cdot)$ to be either $TopRank(\cdot)$, i.e., retrieving all models ranked $TOP - N(percentiles)$, or $BottomRank(\cdot)$, for the $Bottom - N(percentiles)$ performers at training time t_i. Then, the positive persistence $\Pi_{Positive}(N)$ and $\Pi_{Negative}(N)$ informs on the probability for configurations in Ω to remain $Top - N$ or $Bottom - N$ performers over time, respectively.

$$\begin{aligned} \Pi_{Positive}(N) &= \Pi(TopRank(\cdot), N) \\ \Pi_{Negative}(N) &= \Pi(BottomRank(\cdot), N) \end{aligned} \tag{7}$$

We also propose to measure the *area under the curve* of the *persistence*. This metric shall inform on the evolution of the persistence as a function of the rank N (Equation 8).

$$AuC(\Pi(\cdot, N)) = \int_1^{N_{max}} \Pi(\cdot, k)\, dk \tag{8}$$

3.6 Fitness Landscape Footprint

In order to provide researchers with a tool that helps them to characterize, and analyze the potential and difficulty of a NAS problem, and to compare different problems (including different landscapes, fitness functions, neighborhood operators, and search spaces), we propose the **fitness landscape footprint**. The *footprint* is defined as a set of the following key quantities, derived from the analysis of the landscape:

- *Overall fitness*: A measure of the *expected overall fitness* f, and its *standard deviation* for all sampled configurations $y \in \Omega$. These inform on *how easy* it is to fit the ML task at hand with search space Ω.
- *Ruggedness*: A measure of the difficulty of performing *local search* in Ω via analysis of *random walks* (Section 3.3). In the case of several available walks, we select the ruggedness $\rho_{mean}(1)$ as the average of $\rho_i(1)$ for all evaluated routes i on a given

dataset d. The final measurement of the ruggedness τ_d is obtained as $\tau_d = 1/\rho_{mean}(1)$. A large ruggedness would imply large fluctuation (little correlation) from one step to the other of a local search. Little values indicate smoothness (high correlation) in fitness.

- *Cardinal of optima*: An estimation of the number of *local optima* in Ω (Algorithm 2).

- *Positive & negative persistence*: The probabilities Π for model configuration $y \in \Omega$ to remain among the best (or worse) over time (Section 3.5). In particular, the probability $\Pi(N = 25)$ of keeping a Q1 rank(Top or $Bottom$ -25%), and the *area under the curve* $AuC(N = 25)$ of the persistence for $N =< 25$. While the measurement of persistence at Q1 would inform on the chance of getting an elite model, its AuC (see Equation 8) would tell us about the evolution of such persistence at more restricted ranks ($N < 25$).

4. Experimental Setup

Our experiments aim to describe and compare the landscape of NAS problems using the *fitness footprint*. This section describes the datasets used to derive NAS landscapes, and provides additional details regarding the experiments.

4.1 Data Sets

CIFAR-10 is an image classification data se (Krizhevsky (2012)). It consists of 60000 images (32x32 pixels) and its correspondent label (with no overlap). The data is split in 10 classes (6000 images per class), namely: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Also, the data is split into training (50000 samples) and test (10000 samples).
The So2Sat LCZ42 is an *Earth observation* image classification dataset (Zhu et al. (2020)). It contains co-registered image patches from the Sentinel-1 and Sentinel-2 satellite sensors, all assigned to a single label out of 17 classes. Each pair is of 32x32 pixels, 10 multi-spectral bands for Sentinel-1 images and 8 bands for Sentinel-2. The existing classes describe each image patch in the global land cover framework of *local climate zones* (LCZs), with 10 classes assigned to urban or built areas (1 to 10) and 7 to natural sites (11 to 17). The classes are as follows: compact high-rise (1), compact mid-rise (2), compact low-rise (3), open high-rise (4), open mid-rise (5), open low-rise (6), lightweight low-rise (7), large low-rise (8), sparsely built (9), and heavy industry (10), dense trees (11), scattered tree (12), bush, scrub (13), low plants (14), bare rock or paved (15), bare soil or sand (16), and water (17). Note that in the original data set the classes 11 to 17 are related as A to G. Figure 2 displays four (4) pairs of Sentinel-1 and Sentinel-2 image patches respectively from classes 2, 6, 14 and 17.

The data is split into training (352366 images), validation (24188) and test (24119). It is important to note that two various pools of cities were used to build So2Sat LCZ42: samples collected from 32 cities around the globe were selected to form the training set, while samples from 10 other cites were used for the validation and test set, with a geographical split (east and west) to insure distinct samples.

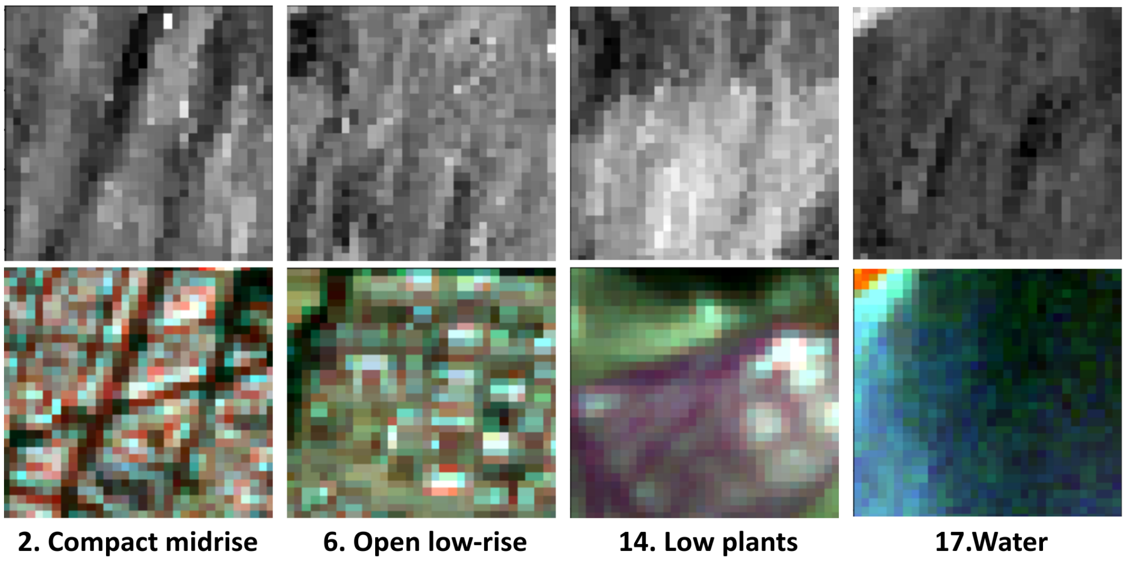


Figure 2: So2Sat LCZ42 samples. The top row contains SAR patches (Sentinel-1), followed by the associated Multi-spectral patches (Sentinel-2) in the bottom row.

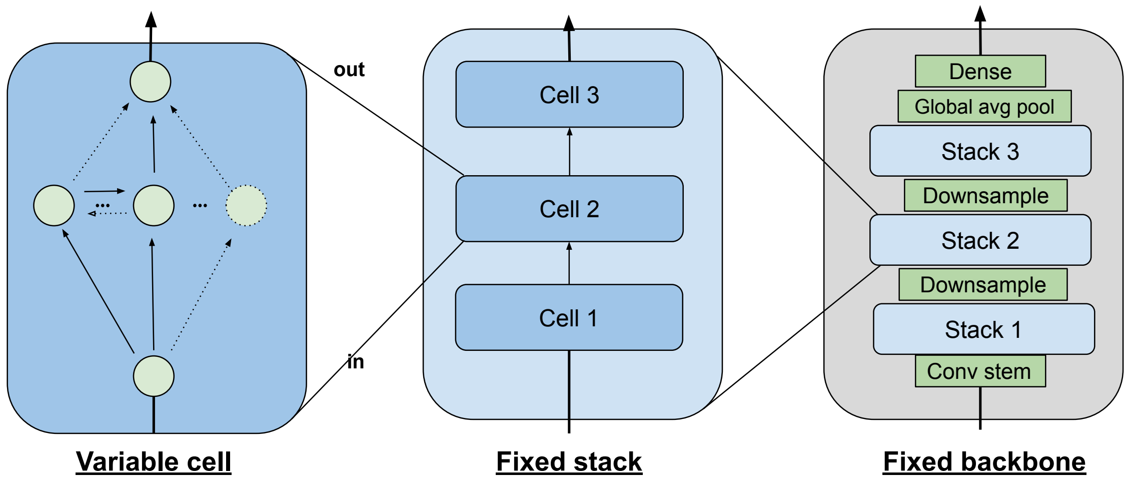


Figure 3: Visual description of the backbone of image classifier samples in NASBench-101.

The NasBench-101 is a benchmark for NAS methodologies in the context of image classification (Ying et al. (2019)). It consists of $N_{sol} = 453k$ distinct CNN architectures and their fitness evaluation at various training steps (4, 12, 36 and 108 epochs), on the dataset of CIFAR-10. Its underlying search space Ω is of CNN configurations with a fixed backbone consisting of a head, a sequence of three (3) repeated blocks, followed by a dense softmax decision layer. The head (conv stem) is a 3 x 3 convolution with 128 output channels. Each block or stack, is a sequence repeating three (3) times an elementary unit, referred to as *cell*. In this *micro search-space*, a cell is a DAG containing at most $V = 7$ nodes and $N_{edges} = 9$ edges. Moreover, each node has a label selected out of $L = 3$ possibilities: {3x3 convolution, 1x1 convolution, 3x3 maxpool}. In practice, a solution $x \in \Omega$ is represented by an adjacency matrix of variable size and its list of operators. Figure 3 illustrates the structure of the image classification backbone used in *The NasBench-101.*

4.2 Additional details

The following paragraphs provide experimental details. First, we explain the custom encoding used to represent samples involved in the FLA of both CIFAR-10 and So2Sat LCZ42. Then we detail the sampling of the search space, the performance evaluation speed-up, data distribution and others, only involved in So2Sat LCZ42 experiments.

Custom encoding: In order to analyze the landscape of both datasets, we use an alternative encoding to Ying et al. (2019), for solutions of the search space. In the original DAG, there are at most five (5) intermediate nodes (excluding IN and OUT), each labelled using L. In the new DAG, we consider non-labelled nodes by replacing each with three (3), for each possible state of operator in L. The new graph has a fixed number of nodes, $V = 1 + 5 * 3 + 1 = 17$ and the resulting adjacency matrix is non upper-triangular and of size $S = 17 * 17 = 289$. We decide to identify each candidate by the binary vector obtained after flattening the matrix.

Sampling of the search space: Training numerous models can be expensive (time and hardware resources), thus we identified the smallest number of sample to consider for representative results. Figure 4 displays the density of fitness (on validation, 36 epochs of training on CIFAR-10) for various sample sizes. The curves represent the fitness density for randomly selected models. Based on this information, we set $N = 100$ samples, because it presents a good trade-off between quality and number of evaluations. In practice, we use a *Latin hypercube sampling* (LHS) strategy to draw the samples (to have a well distributed sample). Because of the complexity of sampling on the larger encoding, we perform LHS on the joint representation of original adjacency matrix and list of operators.

Data distribution: Regarding experiments on So2Sat LCZ42, we split the original training set randomly into *final training* (80%) and *final test* (20%). The motivation is to ensure training and evaluation of models on the same data distribution, as done in CIFAR-10. For each data sample, we only consider Sentinel-2 as source of input.

Performance evaluation speed-up: In order to speed up the evaluation of models on So2Sat LCZ42, we look for the smallest subset of training data enabling representative fitness in test. Figure 5 shows fitness in test as a function of the share (%) of the training set, assessed with the *overall accuracy*, *kappa Cohen coefficient*, and *average accuracy.* The values are normalized with respect to using 100% of the training set. We identify that using 35% of the training set enables to reach 96.5% of the reference performance. Thus, we ran our experiments using this setting. For each model, we consider the fitness as the average fitness of three independent runs.

Miscellaneous: Due to memory limitations, we set the size of the batches to 128 (instead of 256) when training or evaluating on So2Sat LCZ42. The rest of the experimental hyperparameters are default as in Ying et al. (2019). The training and inference were done on an Nvidia V100 GPU.

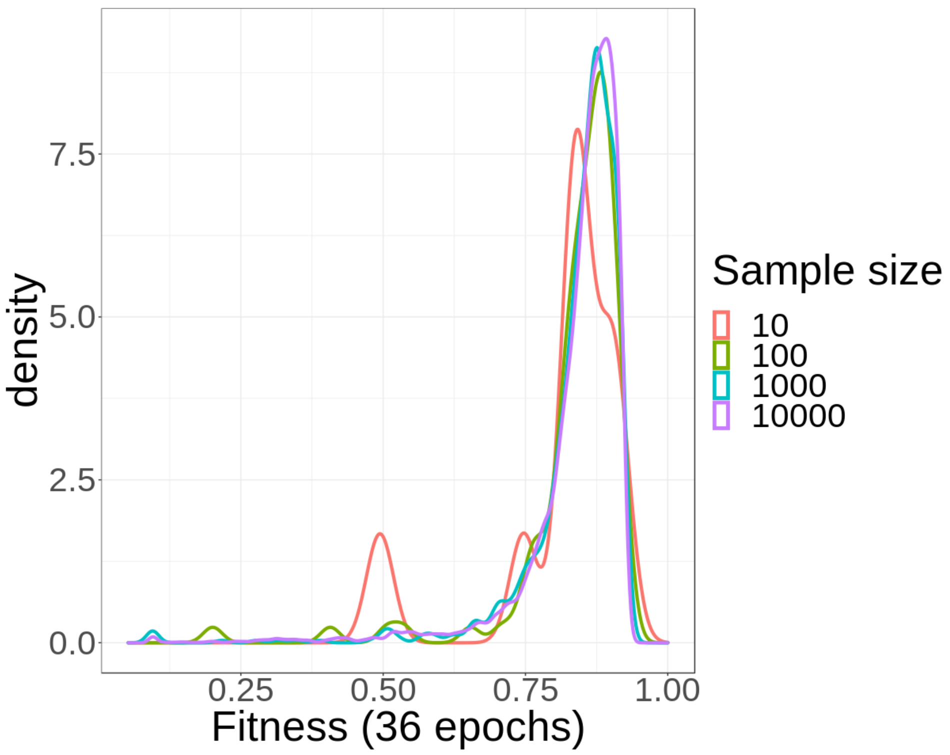


Figure 4: Density of fitness for various sizes of samples of CIFAR-10.

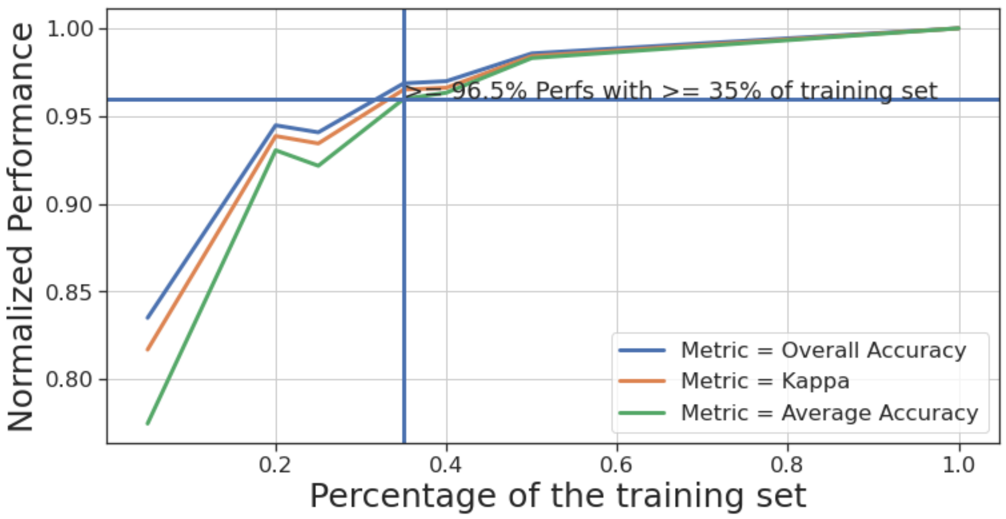


Figure 5: Fitness as a function of the share of training data on So2Sat LCZ42.

5. Results

This section introduces the experimental results. First, we study the fitness distribution for both problems. Then, we analyze the landscape characteristics, such as the *fitness distance correlation*, the *ruggedness* and *local optima*. Next, we present the *persistence* results. Finally, we *assess and compare* the respective *fitness footprints*.

5.1 Density of Fitness

In the context of NAS, the density of fitness measures the potential of a search space in fitting a given task. Figures 6 and 7 show the density of fitness in test for various training budgets for CIFAR-10 and So2Sat LCZ42, respectively.

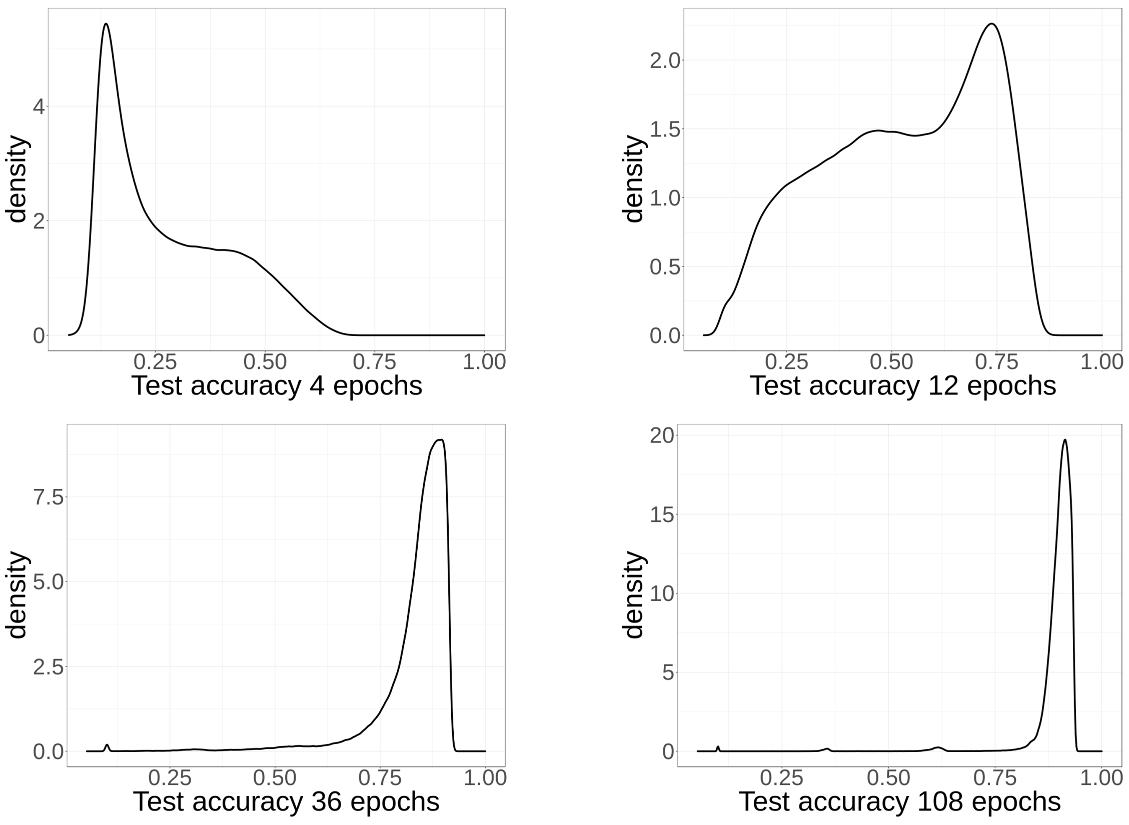


Figure 6: Density of fitness for various training budgets on CIFAR-10.

In both cases, the longer the training, the closer is the density to 1, the highest (and best) possible fitness value, with most fitness above 75% after 36 epochs. Thus, when trained long enough (≥36 epochs), most solutions of the search space show very good fitting capacity on both data sets. Therefore, from the search perspective, there is a high chance of retrieving a good solution from Ω on both domains, regardless of the ability of the NAS algorithm itself.

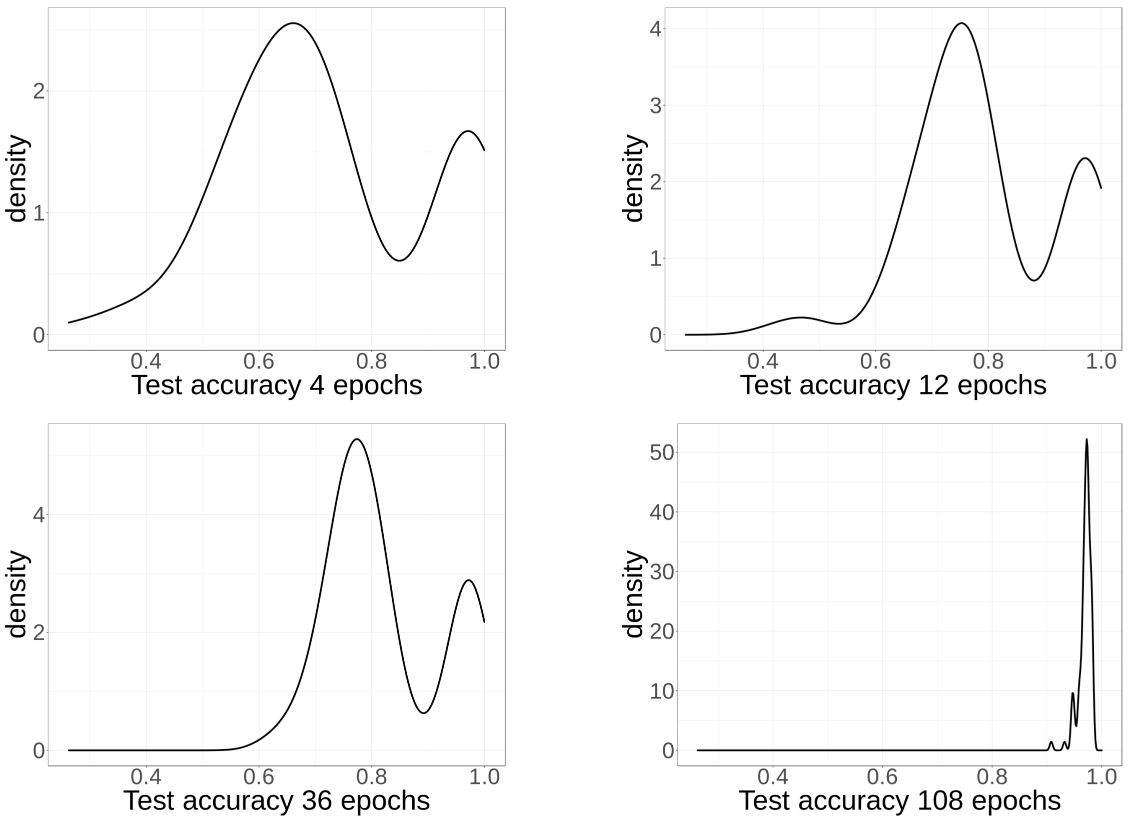


Figure 7: Density of fitness for various training budgets on So2Sat LCZ42.

In addition to measuring empirical densities of fitness, we investigate how they compare to theoretical distributions. Figure 8 provides results for such comparison on CIFAR-10. The theoretical distributions are the Beta (red), Weibul (green) and Lognormal (blue). The top-left and bottom-left hand figures show the associated histograms and plots of cumulative density functions (CDFs) of fitness. The top-right and bottom-right hand figures show the quantile-quantile (QQ) and percentile-percentile (PP) plots.

Table 1 presents the fitting error for each theoretical distributions. The fitness is observed in validation after 36 epochs of training. On CIFAR-10, PDF fitting is feasible and in favor of the Weibul distribution, which is close or on par with the empirical distribution of accuracy. Indeed, in terms of fitting error the Weibul distribution generates the highest likelyhood and lowest AIC and BIC error values.

Error Metric / Function	*Beta*	*Weibull*	*LogNormal*
Likelihood	5007293	**534817**	250868
AIC	-1001455	**-1069630**	-501732
BIC	-1001433	**-1069608**	-501710

Table 1: PDF fitting error on CIFAR-10

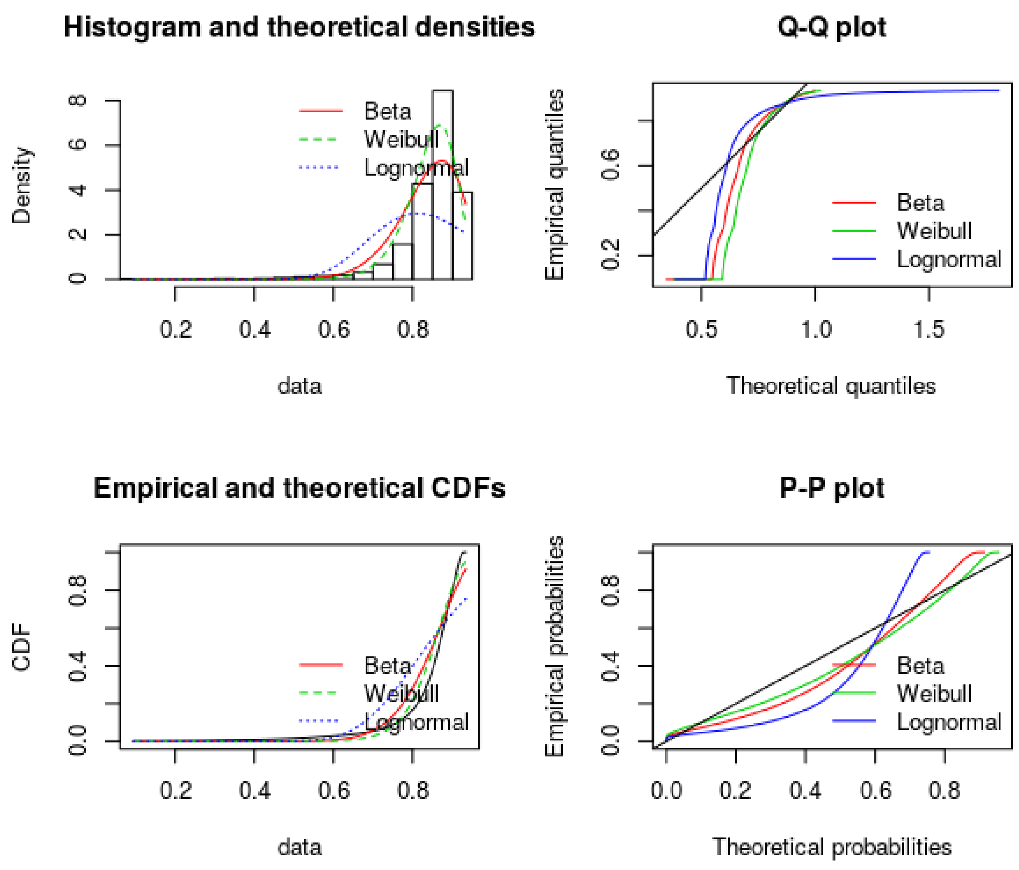


Figure 8: Comparison of empirical and theoretical densities of fitness on CIFAR-10, after 36 epochs of training.

Figure 9 and Table 2 present the results on So2Sat LCZ42. In this case, the LogNormal distribution fits better the challenging multi-modal empirical distribution. The other theoretical distributions fail to capture its second modality.

Error Metric / Function	*Beta*	*Weibull*	*LogNormal*
Likelihood	84.47	83.32	**92.24**
AIC	-164.91	-162.64	**-180.48**
BIC	-159.73	-157.43	**-175.27**

Table 2: PDF fitting error on So2Sat LCZ42

To summarize, the evaluated search space provides with high fitness values already after 36 epochs of training, on both datasets. Results also indicate the possibility to model the distribution of fitness of Ω for both datasets.

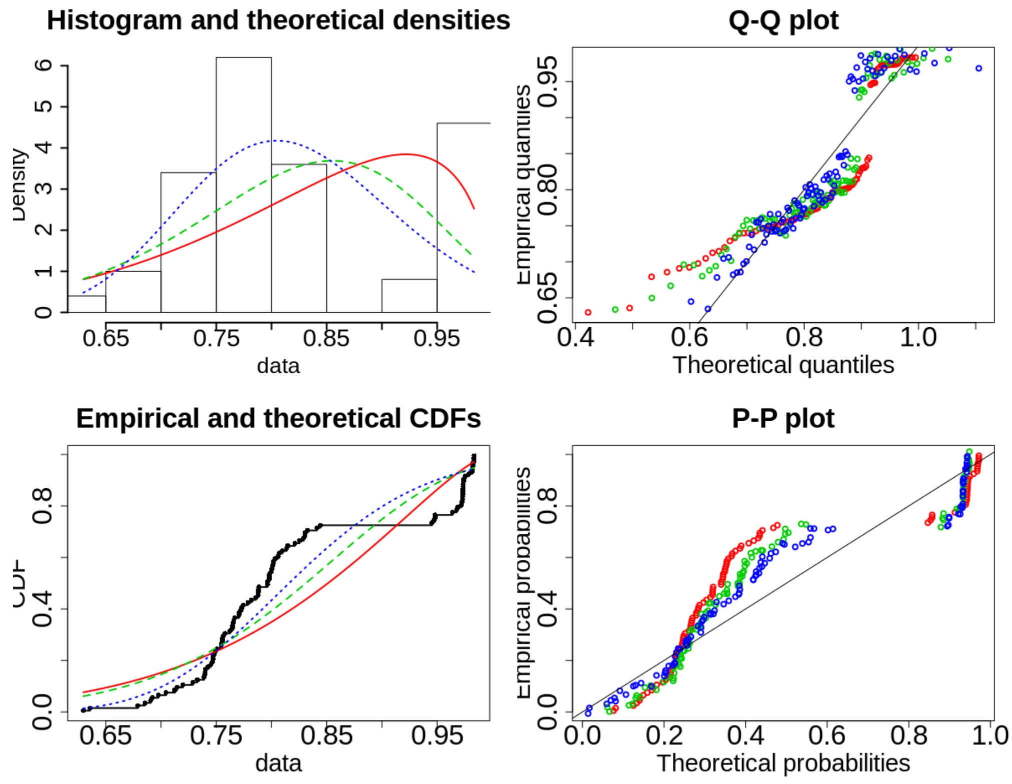


Figure 9: Comparison of empirical and theoretical densities of fitness on So2Sat LCZ42, after 36 epochs of training.

5.2 Fitness Distance Correlation

Figures 10 and 11 shows the FDC for CIFAR-10 and So2Sat LCZ42, respectively. The top-left plot shows the histogram of the Hamming distance to the global optimum. The top-right plot compares the FDC (linear fit) with the trained models (36 or 108 epochs). The bottom-left and right figures show the individual FDC (36 and 108 epochs of training). In all cases, the fitness is measured on the test set.

The histograms of distance to the optimum show, for both datasets, distributions appearing to be uni-modal and bell-shaped. They are centered around distance $d_{hamming} = 10$ and covering a wide range of distances. These results suggest that the models sampled on So2Sat LCZ42 are diverse and representative (of the search space). It also establishes a common ground for comparison of the FDC on both problems.

Regarding CIFAR-10, after 36 epochs of training, most solutions show a good fitness, i.e., above 70%, on the whole spectrum of distances. Also, a consistent increase in fitness (close to 1.66%) per unit of distance travelled towards the optimum is observed. Moreover, aside from the outliers, we notice the regrouping of poorer performers at $d_{hamming} = 14$, creating a valley-like shape in the landscape. After 108 epochs of training, the increase of fitness when approaching the optimum diminishes, indicating a flat landscape or *plateau*.

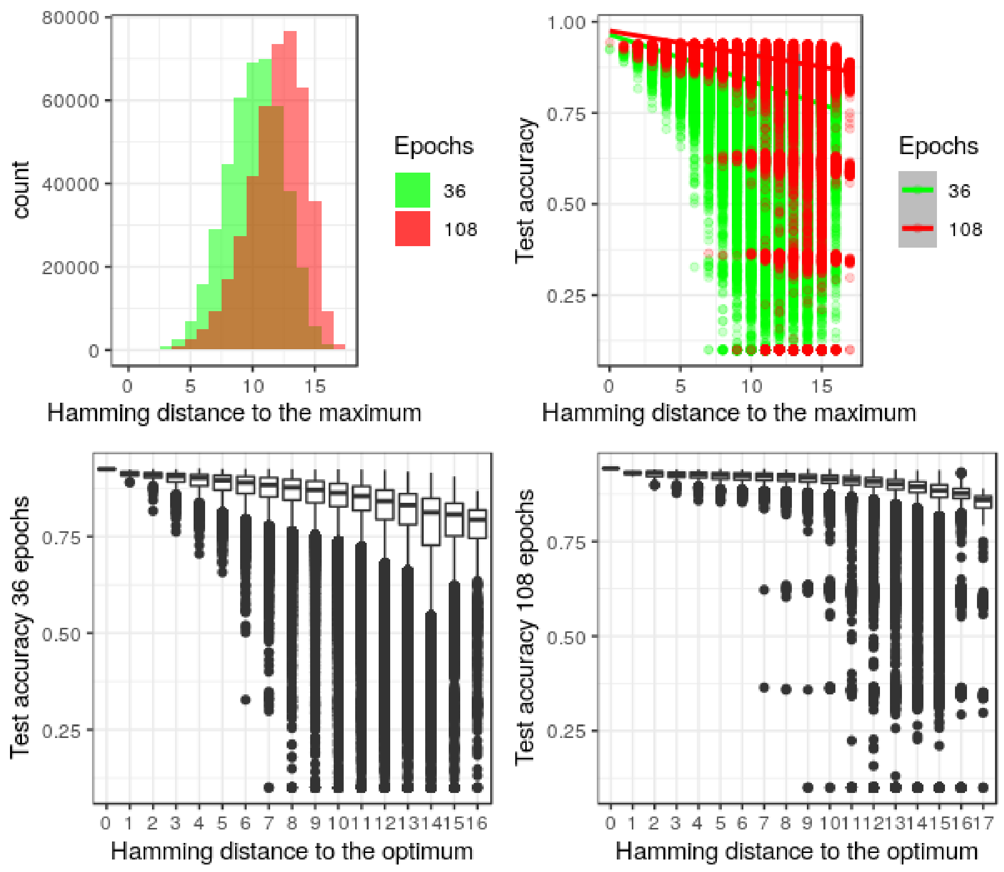


Figure 10: Fitness distance correlation on CIFAR10.

Regarding So2Sat, most of the solutions perform well (above 60% of fitness) after 36 epochs, regardless of their proximity to the optimum. The non-consistent increase in fitness when approaching the optimum suggest high ruggedness and the existence of several local optima (e.g., $d_{hamming} = 10, 11, 14$). In the long run the landscape becomes flat, with most solutions exhibiting high fitness (above 92%).

To summarize, the FDC of both problems show a landscape improving towards high fitness as the training budget increases, and with a plateau-like shape. Despite an apparent higher ruggedness, NAS trajectories could benefit with higher gain from training only for 36 epochs. Therefore, we decided to focus the remainder of this study in the more challenging scenario of a training limited to 36 epochs.

5.3 Ruggedness

To analyze the ruggedness, we study the behavior of random walks in Ω. Figure 12 shows evaluations of thirty (30) independent random walk routes on CIFAR-10. Each route is composed of one hundred (100) steps, starting from a randomly sampled configuration.

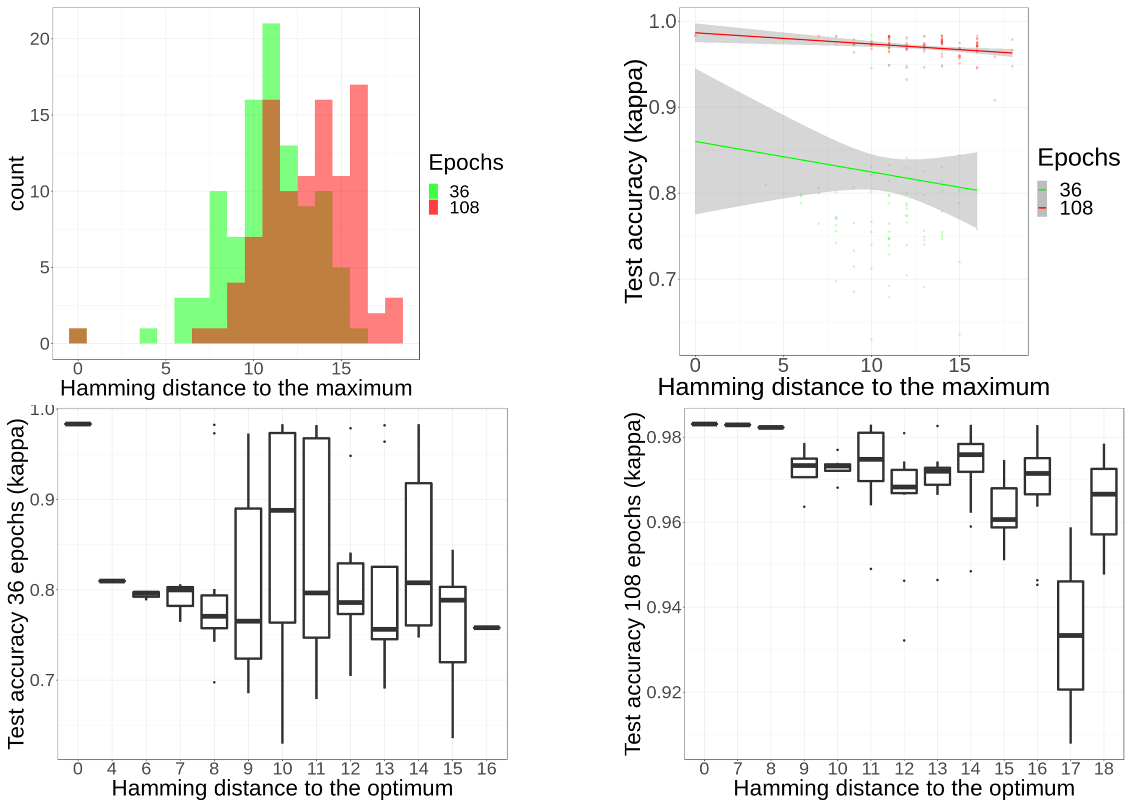


Figure 11: Fitness Distance Correlation on So2Sat LCZ42.

Figure 13 shows the fitness distribution for each route. Overall, there is a high fluctuation in fitness from one route to the other. Most exhibit a fitness above 75%, even though some have distributions in the lower end, suggesting walks being stuck near local minima. For example, route 22.

Particularly, two routes stand out given their distributions of fitness. Route 1 (R1) displays a median (med=0.89) in the higher end, with little variance (std=0.03), while route 16 (R16) has the lowest median of all (med=0.58), with a widespread distribution (std=0.14).

As a ground of comparison, we also evaluate those two extremes on So2Sat LCZ42. Results for both datasets are presented in Figure 14. All curves result from a processing with a moving average of five (5) steps.

For R1, we observe distributions that are centered around high values with little variance. Both paths show several identical bumps and curvatures. Subtracting a constant value $K = 6\%$ to the fitness on So2Sat LCZ42 results in a curve close to R1 on CIFAR-10. Similarly, for R16, the route on So2Sat LCZ42 shows a curvature similar to CIFAR-10. Removing a constant $K = 20\%$ from the fitness on So2Sat LCZ42 results in a curve closely fitting R16 on CIFAR-10. Therefore, the results suggest a similar ruggedness on both datasets, for the areas of the landscape being explored with R1 and R16. Additionally, it

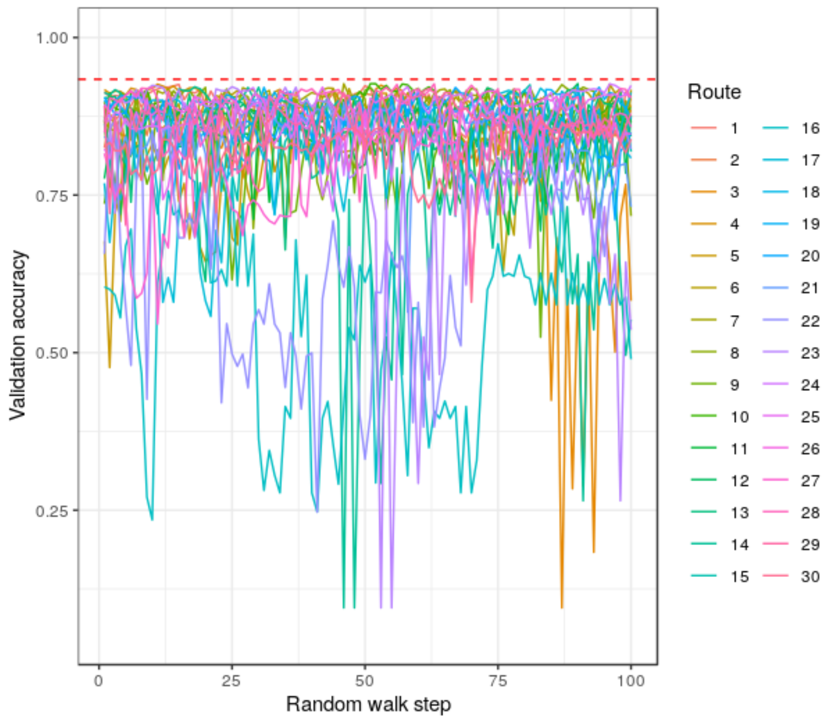


Figure 12: Random walk routes after 36 epochs on CIFAR-10.

is an indication of a common imprint of the search space Ω on trajectories of NAS, across datasets. So far, we identify a variability K in fitness, such as: $6\% \leq K \leq \%20$.

Summarizing, the variability (fitness) observed in the random walks suggests that there are areas in Ω with higher (or lower) fitness potential. The comparison of a few routes show similar trajectories and ruggedness on both datasets.

5.4 Local Optima

To complement the analysis of *ruggedness*, we look to quantify the number of *local optima* in Ω. Table 3 presents the results of the estimation on CIFAR-10. The fitness consist in the accuracy in test after 36 epochs of training. The estimation is obtained by executing Algorithm 2 for $T = 9$ trials. For each trial i, we perform $M = 200$ runs of *best improvement local search* in order to measure the index of the first occurrence of duplicates k_i. Note that because of the ascending nature of the BILS algorithm, it leads to local maxima.

On average, the length of the step is 2.86, with an improvement in the fitness of 5.03%. Over the whole experiment, the average index of repeat for a budget of M runs is $k_{mean} = 94$. Using Equation 5 and k_{mean}, we estimate $N_{CIFAR-10} = 6373$ local optima (maxima) on Ω (CIFAR-10). Besides, we report six (6) additional failing runs with no duplicates found within M runs.

Next, we look to approximate this value on So2Sat LCZ42 with only a few evaluations. Assuming a continuity in fitness for neighboring solutions in Ω, i.e., locality, we derive the

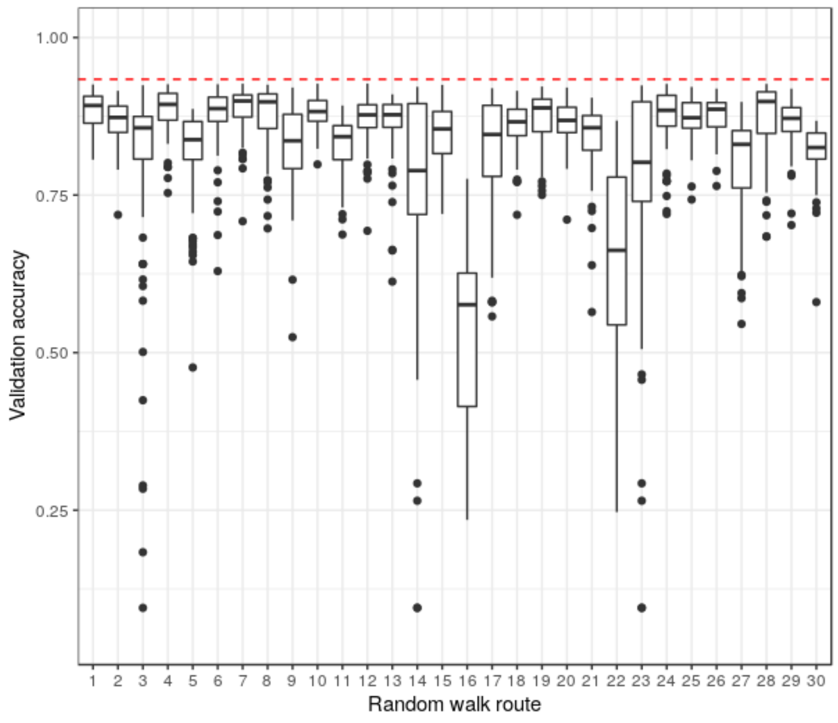


Figure 13: Distribution of the fitness of the random walks on CIFAR-10 (36 epochs training).

Trial	*Avg. Step*	*Avg. Improvement(%)*	*First Repeat k*	*Cardinal*
1	2.90	4.53	94	6373
2	2.89	5.48	57	2343
3	2.78	4.76	26	487
4	3.02	5.55	58	2426
5	2.86	5.06	38	1041
6	3.01	4.5	195	27429
7	2.83	5.77	52	1950
8	2.80	4.78	129	12003
9	2.70	4.79	197	27994
Summary	**2.86**	**5.03**	**94**	**6373**

Table 3: Enumeration of *local optima* (maxima) in CIFAR-10 via the *birthday problem* statistics.

number of local maxima N_{proxi} by counting solutions with higher fitness than their $N_{nei} = 10$ nearest neighbors. Using such approximation for the $N_{samples} = 100$ identical samples on both datasets, we derive a ratio of enumeration using:

$$r_{optima} = \frac{N_{proxi-LCZ42}}{N_{proxi-CIFAR-10}} \tag{9}$$

Then, we derive final the enumeration on So2Sat LCZ42 using:

$$N_{LCZ42} = r_{optima} * N_{CIFAR-10} \tag{10}$$

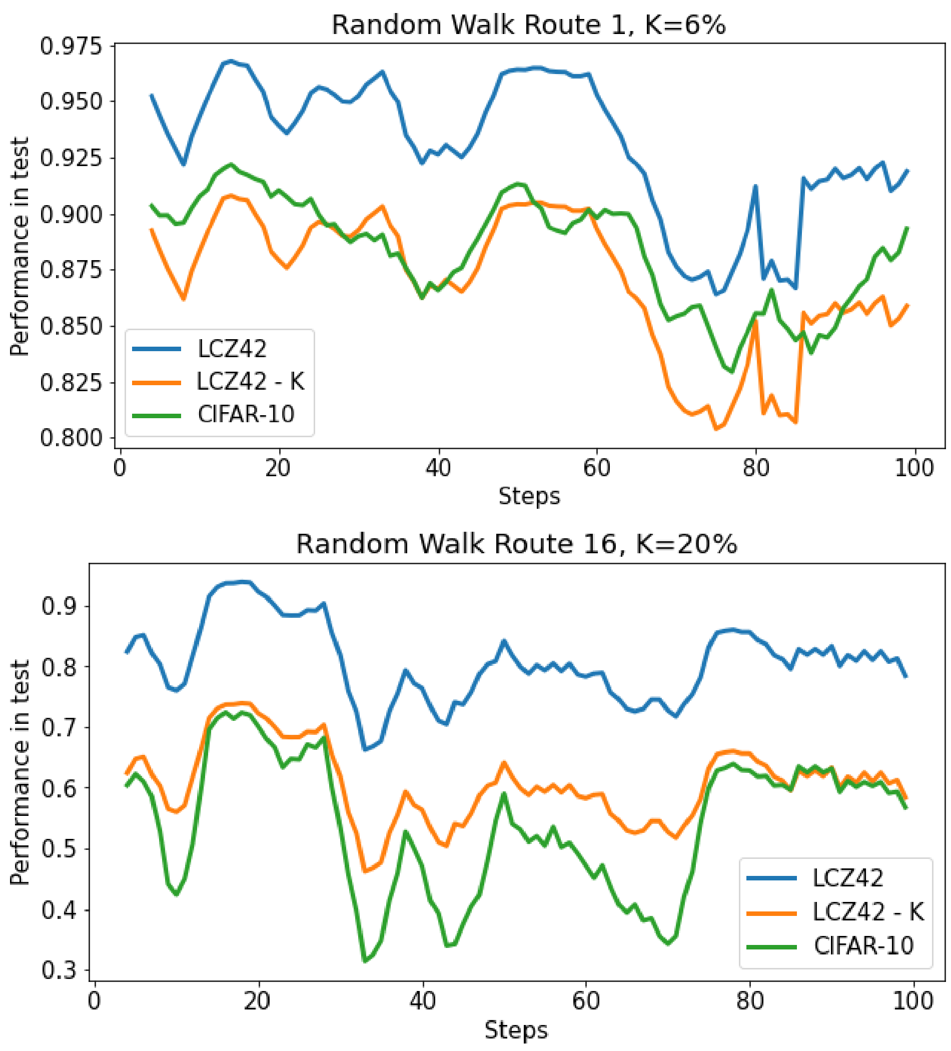


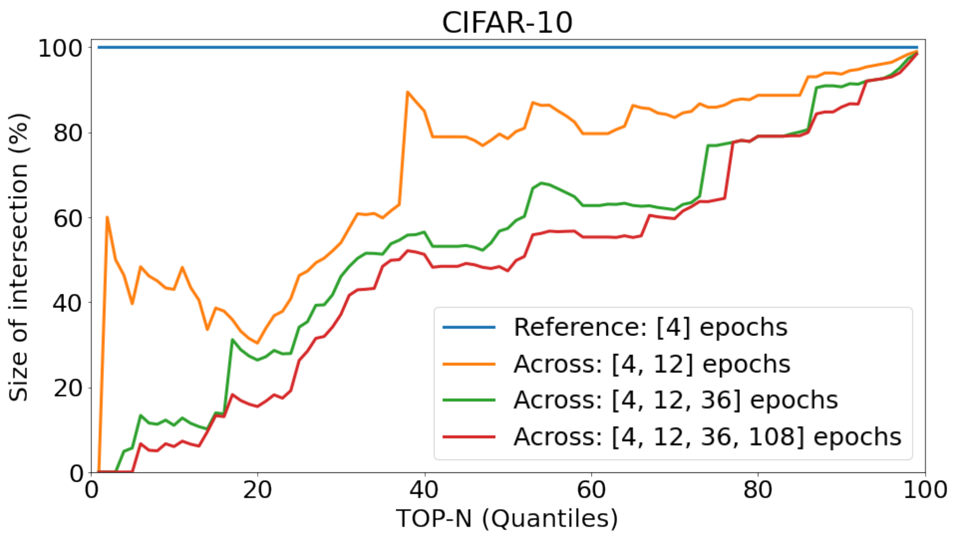
Figure 14: Random walk routes 1 and 16 on CIFAR-10 and So2Sat LCZ42. Fitness in test for models trained 36 epochs.

where, $r_{optima} = 75\%$ is the obtained ratio, and $N_{CIFAR-10} = 6373$ the ground truth approximation on CIFAR-10. This leads us to $N_{LCZ42} = 11153$, the approximation of the enumeration on LCZ-42.

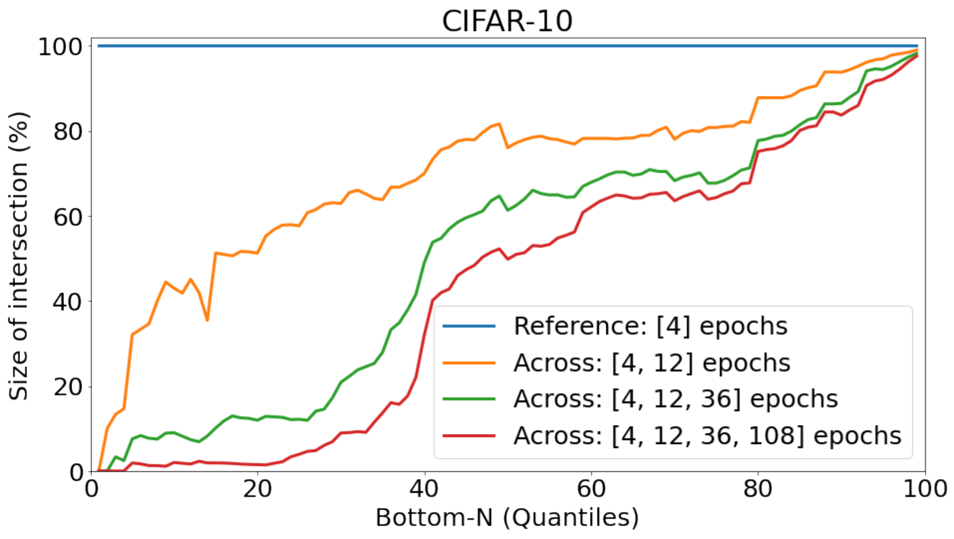
Overall, results indicate approximately 75% more local optima on the landscape of LCZ-42 than on CIFAR-10.

5.5 Persistence

Also, we analyzed the samples from the perspective of *persistence* in their rank over time. Figure 15 depicts results of *persistence* on CIFAR-10, considering $N_{samples} = 1000$ samples and their fitness in test. The top and bottom plots show the positive and negative persistence, respectively. For each plot, the blue curve represent the reference population, i.e., the set of models at a given $Rank - N$ (Nth percentile) considering the fitness after 4 epochs of training. The yellow curve stands for the share of these models maintaining the same $Rank - N$ after 12 epochs. The green and red curves are for the remaining models also maintaining the same $Rank - N$ after 36 and 108 epochs of training.



(a)



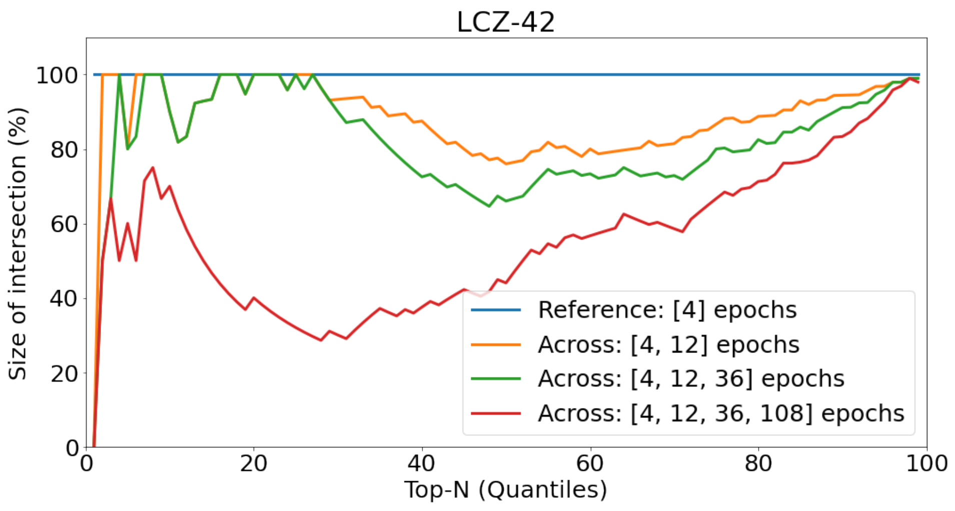
(b)

Figure 15: Positive and Negative Persistence across training regimes in CIFAR-10.

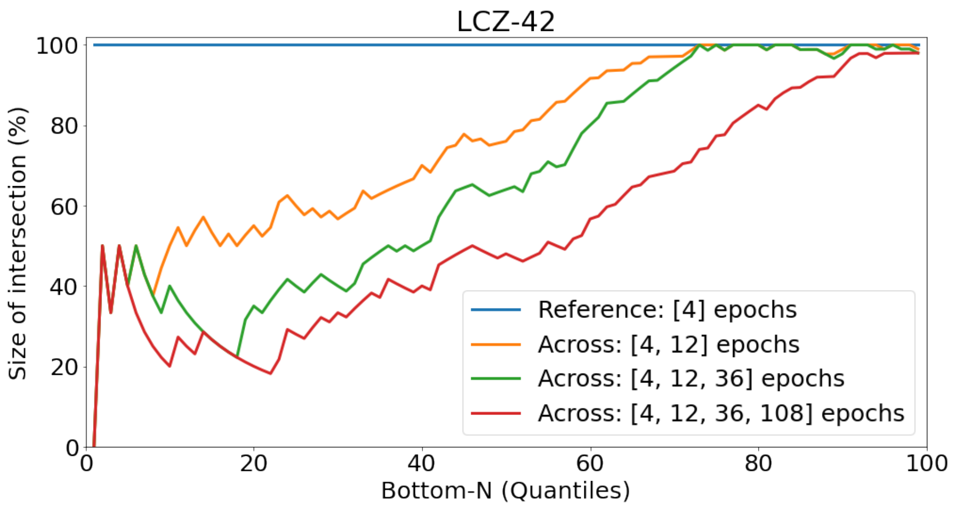
Regarding CIFAR-10, the chance of remaining $Top - N\%$ until 12 epochs is rather high for all N (all above 30%), despite an important drop with N below 40. We notice that the chance of remaining $Top - N\%$ performer until 36 epochs and until 108 epochs, are almost overlapping, with a consistent decrease as N increases. Similarly, the chance of remaining

$Bottom - N\%$ performer until 36 and until 108 epochs are tied, even tough for N below 20, chances tend to zero. If a model is $Top - N\%$ (or $Bottom - N\%$) performer until 36 epochs, it will most likely remain $Top - N\%$ (or $Bottom - N\%$) performer until 108 epochs.

Figure 16 introduces the *persistence* for So2Sat LCZ42. In this case, the positive persistence remains quite high (above 25% for all N). For the elite performers (Q1), we notice a high peak in persistence as N drops below 30. On the other hand, the negative persistence displays a steady drop until N equals 20. Results on this problem indicate in general higher positive and negative persistence, in particular for the elite N-performers, solidifying their rank over time.



(a)



(b)

Figure 16: Positive and Negative Persistence across training regimes in So2Sat LCZ42.

Overall, the positive persistence in both problems evolves similarly, and it is noticeable even for ranks below the first quartile (Q1). On the other hand, the negative *persistence* on LCZ-42 remains consistently higher.

5.6 Fitness Landscape Footprints

Finally, we present the *fitness footprint* for both datasets. Figure 1 depicts the *footprint* of CIFAR-10 in blue, and So2Sat LCZ42 in orange. Except for *persistence*, the fitness (test) is considered for 36 epochs, as the landscapes appear more challenging at this stage. The main takeaways are:

- *Overall fitness*: Both data sets show a similar overall fitness (around 83%), but So2Sat LCZ42 present a slightly higher deviation (10%), suggesting more variability in the fitness of its search space.
- *Ruggedness*: The 10% higher *ruggedness* on CIFAR-10 (1.93 instead of 1.75) is coherent with the qualitative assessment made of the random walks (Figure 14), suggesting more fluctuations and difficulties to overcome by *local search*-based algorithms on CIFAR-10 than on So2Sat LCZ42. While this was assessed on 36 epochs, we expect it to hold true for longer training regimes.
- *Cardinal of optima*: With approximately 75% more local optima on So2Sat LCZ42 (11153 versus 6373), the chances of being trapped in a non-optimal region of the space are higher on So2Sat LCZ42 than on CIFAR-10. Then, solution *diversity* should be considered when designing a (*local search*-based) algorithm to do NAS on So2Sat LCZ42 (i.e., exploration-exploitation).
- *Positive persistence*: Elite models persist over time, i.e., the chances of finding a model that will be top-25% performer in test from 4 until 108 epochs of training is 32%. Moreover, on So2Sat LCZ42, elite models (rank $< 25\%$) will remain on the top with higher probability (0.48). Therefore, spotting elite performers early could help to improve NAS performance.
- *Negative persistence*: Furthermore, we observe on So2Sat LCZ42 a higher chance of models remaining bottom-25% performers across training time (28% versus 6.4%) and a larger *AuC* of *persistence* for ranks below 25%. In other words, it is important to avoid poor performers early is more critical on So2Sat LCZ42, while it is not a concern on CIFAR-10.

Overall, we observe similarities in the characteristics of the *footprints* of both problems. Some key aspects such as the *persistence* could be help improve *local search* baselines on both problems.

6. Discussion

The following paragraphs provide a discussion of the results obtained. First, we discuss results of each individual aspects of the *fitness landscape analysis*. Then we discuss the *footprint* as a way to summarize the analysis.

Density of fitness: Our first experiments consisted in measuring the PDFs of fitness on both problems, and attempting to fit them with various theoretical distributions. In doing so, we notice similarities in the evolution of densities in both problems. Indeed, the PDFs tend to be uni-modal, very narrow distributions and close to 1, the longer the training. In other words, most solution of the search space have high-capacity fitting both datasets, given enough training. We consider this as a first indication that most NAS search strategies might perform well with 108 epochs of training, regardless of their search ability (on the given problems). Thus, when benchmarking NAS performances on both datasets, we recommend the use of less training in order to better differentiate the algorithm ability.

Fitness Distance Correlation: Investigating the FDC enabled us to uncover the potential gain in fitness per travelled hamming distance for a NAS optimizer. The longer the training, the flatter the fitness landscapes, with high fitness across all the distances to the global optimum. These results indicate that a NAS algorithm can expect higher gain per iteration with intermediate training budget, i.e., 36 epochs. Indeed, the flat shape of the landscape after 108 epochs allows for a more limited margin in NAS *trajectories*.

Moreover, the results hold for *local search* based algorithms, as well as evolutionary computation-based approaches, on the given search space and sample encoding.

Ruggedness: Besides, the analysis of the random walks on both problems (see Figure 14) for two different routes shows that the fitness on both datasets are just at a constant K away, with a similar curvature. This suggests a strong influence from a common element of the landscapes: the search space Ω.

Additionally, the slightly larger *ruggedness* coefficient on CIFAR-10 suggests slightly more fluctuations in local search trajectories for the dataset. However, because of the expensive nature of the experiments, we estimated the coefficient on LCZ-42 out of fewer walks (2 versus 30). Future work could investigate a more in depth measurement and comparison, given a larger budget of evaluations. This could help better understand how the locality in fitness differs from one landscape (dataset) to the other.

Local optima: We proceeded to describe the landscapes by assessing the existence of *local optima*. Our estimation indicated a larger number of local optima on So2Sat LCZ42 than on CIFAR-10. This suggests adapting local search-based NAS Algorithms to this problem, in order to avoid getting *stuck* in suboptimal areas of the landscape.

Besides, we hereby comment some aspects of the enumeration on CIFAR-10, using the Algorithm 2. We reported a few (6) failing attempts to recover collisions out of $M = 200$ runs. This suggests that for some seeds, the limited budget M of repetition was not enough, and involved a slight underestimation of the average index at collision K_{means}. A more accurate estimation could be obtained with a larger budget M of repetition. In this work, we mainly aimed at providing an early approximation of this value with a restricted budget M, following Matthew and Mullin (1999).

Persistence: Then, we investigated the samples from the perspective of a proposed metric, the *persistence*. Results indicate a similar positive *persistence* of 33% on So2Sat LCZ42 and CIFAR-10. This implies that chances are significant of finding good models early in their training, in both datasets. From the perspective of search, this also suggest that the chances that an area of the search space remains *fruitful* ($Top - Nth$ performers) are high.

On the other hand, we observed a higher negative *persistence* and greater *areas under the curve* (positive and negative) on So2Sat LCZ42. Overall, the high persistence (positive or negative) on So2Sat LCZ42 indicate good chances of finding early in training models, that will perform god (or bad) after longer training. Suggesting a potential gain by spotting good and bad performers early (i.e., considering a short training regime). From the perspective of the search, this also suggest that chances are quite high that a region of the search space remains *fruitful* (with $Top - Nth$ performers) or *bad* (with $Bottom - Nth$ performers) over training time.

Overall, results are in favor of performing search with a short training budget (at most 36 epochs), as models might have a good *persistence* in their fitness.

Fitness footprint: Finally, we introduced the *footprint* as a way to summarize the previous aspects of a FLA.

Regarding So2Sat LCZ42, the *fitness footprint* tells us that we can expect good but slightly more variable fitness on the search space of NASBench-101. It also informs us that when deploying NAS on So2Sat LCZ42 there might be potential margin to gain and losses to avoid using simple heuristics. Indeed, a heuristic to spot elite models early in training could help filter the search space in order to focus on more fruitful areas of the search space. On the other hand, a heuristic to avoid poor performers early could help reduce the overall complexity by saving training time in selection. Another critical heuristic to have is one enabling diversity in *local search* helping to avoid bad regions of the search space. As there is potentially more adjustment in terms of heuristics in order to avoid losses in fitness (larger cardinal of optima, negative persistence), So2Sat LCZ42 might be a slightly more challenging problem for NAS given the current search space.

Regarding CIFAR-10, the *fitness footprint* also describes overall good and stable performances on the search space. A larger *ruggedness* lets us anticipate more difficulties for deploying *local search*. On the other hand, CIFAR-10 also exhibits potential gains to have by incorporating heuristics for spotting early elite models when deploying NAS algorithms.

Overall, we observe similar characteristics in both *footprints*. Therefore, we may extrapolate from one problem to the other.

7. Conclusion

In this paper, we apply *fitness landscape analysis* tools to evaluate the hardness of NAS instances, i.e., the fitness landscape of the architecture search optimization problem in a fixed search space, and with a defined fitness function and neighborhood operator. Given this context, we propose the *fitness landscape footprint*, a novel general-purpose framework to characterize and compare NAS problems. The insights provided by the footprint may be used to assess the expected performance and to relate the difficulties that a search strategy will face at an instance level, among others.

We evaluate our proposal on instances from two image classification datasets: CIFAR-10 and the real-world Remote Sensing dataset of So2Sat LCZ42 for *local climate zone* image classification. For both, we consider the NAS problem of optimizing convolutional neural networks based on the well-known search space of NASBench-101 (Ying et al. (2019)).

Among the findings, we identify several clues indicating that NAS could be performed at shorter regimen (36 epochs), finding elite models early in their training. Other findings show a common signature of fitness of the search space on both datasets and the visualization of landscape of both problems for various training settings.

Last but not least, we believe that ability to compare NAS instances using the *footprint* could help (I) identifying the search space generating the most favorable NAS instance out of several possibilities, (II) identifying the fitness evaluation setting (sensor fusion, dataset) generating the most favorable NAS instance out of several possibilities, (III) identifying the neighborhood operator (sample encoding, distance or mutation function) generating the most favorable NAS instance out of several possibilities. Or (IV) identifying a favorable instance using a combination of (I), (II) and or (III). For future work might investigate some points listed above, as well as the use of insights provided by a *footprint* to help better calibrate a search strategy in a given NAS instance.

Acknowledgments and Disclosure of Funding

Authors acknowledge support by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. [ERC-2016-StG-714087], Acronym: *So2Sat*), by the Helmholtz Association through the Framework of Helmholtz AI [grant number: ZT-I-PF-5-01] - Local Unit "Munich Unit @Aeronautics, Space and Transport (MASTr)" and Helmholtz Excellent Professorship "Data Science in Earth Observation - Big Data Fusion for Urban Research"(grant number: W2-W3-100), by the German Federal Ministry of Education and Research (BMBF) in the framework of the international future AI lab "AI4EO – Artificial Intelligence for Earth Observation: Reasoning, Uncertainties, Ethics and Beyond" (Grant number: 01DD20001) and grant DeToL, and a DAAD Doctoral Research fellowship.

References

E Alba, JF Aldana, and JM Troya. Genetic algorithms as heuristics for optimizing ann design. In *Artificial Neural Nets and Genetic Algorithms*, pages 683–690. Springer, 1993a.

Enrique Alba, JF Aldana, and José M Troya. Full automatic ann design: A genetic approach. In *International Workshop on Artificial Neural Networks*, pages 399–404. Springer, 1993b.

A. S. Bosman, A. Engelbrecht, and Mardé Helbig. Fitness landscape analysis of weight-elimination neural networks. *Neural Processing Letters*, 48:353–373, 2017.

Andrés Camero, Jamal Toutouh, and Enrique Alba. Random Error Sampling-based Recurrent Neural Network Architecture Optimization. *arXiv preprint arXiv:1909.02425*, 2019.

Andrés Camero, Hao Wang, Enrique Alba, and Thomas Bäck. Bayesian Neural Architecture Search using A Training-Free Performance Metric. *arXiv preprint arXiv:2001.10726*, 2020.

Manuel Clergue, Sébastien Verel, and Enrico Formenti. An iterated local search to find many solutions of the 6-states firing squad synchronization problem. *Applied Soft Computing*, 66:449–461, 2018. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2018.01.026. URL `https://www.sciencedirect.com/science/article/pii/S1568494618300322`.

Fabio Daolio, Sebastien Verel, Gabriela Ochoa, and Marco Tomassini. Local optima networks and the performance of iterated local search. *GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation*, 10 2012. doi: 10.1145/2330163.2330217.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. *CVPR*, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, April 2019.

Jonathan Engel. Teaching Feed-Forward Neural Networks by Simulated Annealing. *Complex Systems*, 2:641–648, 1988.

Ima Okon Essiet and Yanxia Sun. Tracking variable fitness landscape in dynamic multi-objective optimization using adaptive mutation and crossover operators. *IEEE Access*, 8:188927–188937, 2020. doi: 10.1109/ACCESS.2020.3031498.

Real Esteban, Aggarwal Alok, Huang Yanping, and V. Le Quoc. Aging evolution for image classifier architecture search. *AAAI*, 2019.

Unai Garciarena, Nuno Lourenço, Penousal Machado, Roberto Santana, and Alexander Mendiburu. On the exploitation of neuroevolutionary information: Analyzing the past for a more efficient future, 2021.

Leticia Hernando, Alexander Mendiburu, and Jose Lozano. An evaluation of methods for estimating the number of local optima in combinatorial optimization problems. *Evolutionary computation*, 21, 12 2012. doi: 10.1162/EVCO_a_00100.

Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, page 184–192, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603700.

Artem Kaznatcheev, David A. Cohen, and Peter Jeavons. Representing fitness landscapes by valued constraints to understand the complexity of local search. *J. Artif. Intell. Res.*, 69:1077–1102, 2020. doi: 10.1613/jair.1.12156. URL `https://doi.org/10.1613/jair.1.12156`.

Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. Nas-bench-nlp: Neural architecture search benchmark for natural language processing. *CoRR*, abs/2006.07116, 2020. URL `https://arxiv.org/abs/2006.07116`.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

Arnaud Liefooghe, Fabio Daolio, Sébastien Verel, Bilel Derbel, Hernan Aguirre, and Kiyoshi Tanaka. Landscape-aware performance prediction for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 24(6):1063–1077, 2020. doi: 10.1109/TEVC.2019.2940828. URL `https://hal.archives-ouvertes.fr/hal-02294201`.

Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21:1–18, December 2020. URL `https://arxiv.org/abs/1909.02453`. To appear.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Rich Caruana Matthew and Matthew Mullin. Estimating the number of local minima in big, nasty search spaces. In *In Proceedings of IJCAI-99 Workshop on Statistical Machine Learning for Large-Scale Optimization*, 1999.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier, 2019.

David J Montana and Lawrence Davis. Training Feedforward Neural Networks Using Genetic Algorithms. *Proceedings of the 11th International Joint Conference on Artificial intelligence - Volume 1*, 89:762–767, 1989.

Matheus Nunes, Paulo M. Fraga, and Gisele L. Pappa. Fitness landscape analysis of graph neural network architecture search spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, page 876–884, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383509. doi: 10.1145/3449639.3459318. URL `https://doi.org/10.1145/3449639.3459318`.

Gabriela Ochoa, Marco Tomassini, Sebastien Verel, and Christian Darabos. A study of nk landscapes' basins and local optima networks. *GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008*, 11 2008. doi: 10.1145/1389095.1389204.

Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60(January):97–116, 2017.

Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. Investigating recurrent neural network memory structures using neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 446–455. ACM, 2019.

Cristiano G. Pimenta, Alex G. C. de Sá, Gabriela Ochoa, and Gisele L. Pappa. Fitness landscape analysis of automated machine learning search spaces. In Luís Paquete and Christine Zarges, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 114–130, Cham, 2020. Springer International Publishing. ISBN 978-3-030-43680-3.

Erik Pitzer and Michael Affenzeller. A comprehensive survey on fitness landscape analysis. In *Recent advances in intelligent engineering systems*, pages 161–191. Springer, 2012.

Aditya Rawal and Risto Miikkulainen. Evolving deep lstm-based memory networks using an information maximization objective. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 501–508. ACM, 2016.

Christian M. Reidys and Peter F. Stadler. Combinatorial landscapes. *SIAM Review*, 44(1): 3–54, 2002. ISSN 00361445. URL `http://www.jstor.org/stable/4148412`.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020.

Nuno M. Rodrigues, Sara Silva, and Leonardo Vanneschi. A study of generalization and fitness landscapes for neuroevolution. *IEEE Access*, 8:108216–108234, 2020. doi: 10.1109/ACCESS.2020.3001505.

Gustavo Rosa, João Papa, Aparecido Marana, Walter Scheirer, and David Cox. Fine-tuning convolutional neural networks using harmony search. In Alvaro Pardo and Josef Kittler, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 683–690, Cham, 2015. Springer International Publishing.

Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *NIPS*, 2016.

Peter F. Stadler. Fitness landscapes. In Michael Lässig and Angelo Valleriani, editors, *Biological Evolution and Statistical Physics*, volume 585 of *Lect. Notes Phys.*, pages 187–207, Berlin, 2002. Springer-Verlag. doi: 10.1007/3-540-45692-9_10.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Kalifou René Traoré, Andrés Camero, and Xiao Xiang Zhu. Lessons from the clustering analysis of a search space: A centroid-based approach to initializing nas. *arxiv*, 2021.

Bas van Stein, Hao Wang, and Thomas Bäck. Automatic configuration of deep neural networks with parallel efficient global optimization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.

Jean-Paul Watson. An introduction to fitness landscape analysis and cost models for local search. In *Handbook of metaheuristics*, pages 599–623. Springer, 2010.

Colin White, Sam Nolen, and Yash Savani. Local search is state of the art for neural architecture search benchmarks. *International Conference on Learning Representations, Workshop on AutoML*, 2020.

Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HygrdpVKvr`.

Xin Yao. A review of evolutionary artificial neural networks. *International journal of intelligent systems*, 8(4):539–567, 1993.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

Arber Zela, Julien Siems, and Hutter Frank. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *International Conference on Learning Representations*, 2020.

You Zhining and Pu Yunming. The genetic convolutional neural network model based on random sample. *International Journal of u-and e-Service, Science and Technology*, 8(11): 317–326, 2015.

Xiao Xiang Zhu, Jingliang Hu, Chunping Qiu, Yilei Shi, Jian Kang, Lichao Mou, Hossein Bagheri, Matthias Haberle, Yuansheng Hua, Rong Huang, et al. So2sat lcz42: A benchmark data set for the classification of global local climate zones [software and data sets]. *IEEE Geosci. Remote Sens.*, 8(3):76–89, 2020.

This page is intentionally left blank.

Annals of Mathematics and Artificial Intelligence
https://doi.org/10.1007/s10472-022-09823-0

A data-driven approach to neural architecture search initialization

Kalifou René Traoré[1,2] · Andrés Camero[2,3] · Xiao Xiang Zhu[1,2]

Accepted: 19 October 2022

Abstract

Algorithmic design in neural architecture search (NAS) has received a lot of attention, aiming to improve performance and reduce computational cost. Despite the great advances made, few authors have proposed to tailor initialization techniques for NAS. However, the literature shows that a good initial set of solutions facilitates finding the optima. Therefore, in this study, we propose a data-driven technique to initialize a population-based NAS algorithm. First, we perform a calibrated clustering analysis of the search space, and second, we extract the centroids and use them to initialize a NAS algorithm. We benchmark our proposed approach against random and Latin hypercube sampling initialization using three population-based algorithms, namely a genetic algorithm, an evolutionary algorithm, and aging evolution, on CIFAR-10. More specifically, we use NAS-Bench-101 to leverage the availability of NAS benchmarks. The results show that compared to random and Latin hypercube sampling, the proposed initialization technique enables achieving significant long-term improvements for two of the search baselines, and sometimes in various search scenarios (various training budget). Besides, we also investigate how an initial population gathered on the tabular benchmark can be used for improving search on another dataset, the So2Sat LCZ-42. Our results show similar improvements on the target dataset, despite a limited training budget. Moreover, we analyse the distributions of solutions obtained and find that that the population provided by the data-driven initialization technique enables retrieving local optima (maxima) of high fitness and similar configurations.

✉ Xiao Xiang Zhu
xiaoxiang.zhu@tum.de

Kalifou René Traoré
kalifou.traore@dlr.de

Andrés Camero
andres.camerounzueta@dlr.de

1 Data Science in Earth Observation, Technical University of Munich, Arcisstrasse 21, 80333, Munich, Bavaria, Germany

2 Remote Sensing Institute, German Aerospace Center (DLR), Münchener Strasse 20, 82234, Weßling, Bavaria, Germany

3 Helmholtz AI, Munich, Germany

Keywords AutoML · Neural architecture search · Evolutionary computation · Search · Initialization

Mathematics Subject Classification (2010) 68Txx · 68T20 · 68T45

1 Introduction

Deep learning has successfully been applied to a wide variety of problems, showing (in many cases) *super human* performance [1, 2]. However, this success has been followed by an increasing complexity of the deep learning models, that in most cases are manually designed [3]. State-of-the-art *deep neural networks* (DNNs) may have several million parameters, thus automating the design of DNNs is the *logical* next step.

Neural architecture search (NAS) is the process of automating architecture engineering [3, 4]. Great advances have been made in this matter, but in practice NAS has not been adopted yet [3, 4]. From an optimization point of view, NAS is a challenging task. It requires dealing with huge search spaces (the deeper the model, the bigger the search space), mixed type solutions (i.e., a combination of integer, discrete, and real values to represent the model), and solutions that are *expensive* to evaluate (i.e., training a DNN on a large data set may take several days).

Researchers are working to alleviate NAS challenges. Several (optimization) approaches have been tailored to design neural networks [5–7], to speed up model evaluation [8, 9], and a lot of effort have been put to design NAS search spaces [5, 6]. Recently, some authors have released performance evaluation databases [10, 11], aiming to democratize NAS, i.e., *everyone* can test a NAS algorithm regardless of having powerful computational resources, and to improve the reproducibility [3].

Despite the great advances made so far, there remain open questions. For example, many NAS approaches can be categorized as population-based algorithms [3, 4]. However, little attention has been drawn to the initialization of the population. Considering all the available resources, including NAS databases, we propose to address the following question: *Can we improve the performance of a population-based NAS algorithm by initializing its population with a data-driven approach?* To this problem, we propose a novel approach to initializing population-based NAS algorithms. First, a tailored clustering analysis of a target search space is performed. Second, after obtaining satisfying clustering results, the centroids are extracted and used to initialize a population based NAS algorithm.

To validate our proposal, we selected three population-based NAS algorithms: an evolutionary algorithm (EA) [12], a genetic algorithm (GA) [13], and aging evolution (AE) [6], and benchmark it on the NAS-bench-101 [10] dataset, against the most popular initialization methods (random initialization and Latin hypercube sampling). We find that centroids extracted using *Bayesian Gaussian Mixture of models* (BGM) for clustering are a promising approach to initialize the population. Particularly, our approach used with GA shows significant long-term improvements (after 2000 iterations, in test) over random initialization and Latin hypercube sampling. When used with EA, a faster convergence (in validation) and a significant long-term improvement over random initialization and Latin hypercube sampling is observed. Besides, we also investigate how an initial population gathered with our proposal on a tabular benchmark can be used for augmenting search on a real world problem, the So2Sat LCZ-42 scene classification dataset. Our results show that this set of solutions help accelerate the convergence EA on the target dataset, despite a short training budget.

Last but not least, we investigate the distributions of the solutions found by the benchmarked algorithms. More precisely, we compare their configurations based on the initialization used for each algorithm. Our results suggest that the proposed initialization method (centroids) enable retrieving local optima (maxima) of high fitness and similar configurations.

To summarize, the contributions of this paper are the following:

- We propose a novel data-driven initialization method for population-based NAS algorithms: After performing a clustering analysis of the search space, we use the obtained centroids to initialize a NAS algorithm. The aim of the proposal is to improve the convergence and long-term performances of a target algorithm, without modifying it.
- We benchmark the proposal on three NAS algorithms and against popular initialization baselines, and report improvements over other initialization techniques.
- We investigate how the proposed method using a tabular benchmark as input can be used for augmenting search on another dataset, the So2Sat LCZ-42. We report similar improvements on the target dataset, despite a limited training budget.

The remainder of this article is organized as follows: The following section introduces NAS and briefly summarizes the state-of-the-art of initialization techniques. Section 3 describes the proposed methodology. Section 4 introduces the experimental setup. Section 5 presents the results. Finally, Section 6 outlines the conclusions and proposes future work.

2 Related work

In this section, we summarize some of the most relevant works related to our proposal. First, we introduce NAS and highlight the state-of-the art, with a special emphasis on metaheuristic approaches. Second, we outline the population initialization problem. Third, we describe the three baseline algorithms that we aim to initialize.

2.1 Neural architecture search

NAS is the process of automating architecture engineering [3]. Currently, it is considered to be a subfield of AutoML [14]. However, its roots can be tracked to the late 1980s, where the use of *evolutionary computation* was explored to design and train neural networks [15–19]. These ideas gather together under the *neuroevolution* concept, and in the 2000s gained popularity thanks to the *NeuroEvolution of Augmenting Topologies* (NEAT) method [20], a *genetic algorithm* (GA) that increasingly evolves complex neural network topologies and weights. Later, due to apparition of deep learning, the neuroevolution research started to attract attention again [3, 4].

From the optimization (algorithm) point of view, many approaches can be found in the neuroevolution literature, ranging from evolutionary algorithm (EA) [21], GA [22], *harmony search* [23] and *mixed integer parallel efficient global optimization* technique [24], to Bayesian optimization [7]. And also, from the point of view of the neural network architecture, e.g., recurrent neural network [25], convolutional neural network [26] and generative adversarial networks [27].

On the other hand, in the past few years, a *new branch* of NAS approaches emerged based on *continuous optimization* (e.g., DARTS [5]). Particularly, these approaches search over a large graph of overlapping configurations (i.e., the *Super-Net*) using a gradient-based approach. These recent improvements result in large speedups in terms of search time [3] but also in some case in a lack of robustness and interpretability [28].

Despite the NAS approach used, the literature stresses the importance of optimizing the architecture of a deep neural network (given a particular problem). The main challenges of NAS are three-fold: First, the number of the parameters increases in proportion to the number of layers, thus the search space is huge. Second, the search space is (usually) a mix of categorical (e.g., the type of operation, the activation functions, ...), real (e.g., the weights), and integer (e.g., the number of hidden layers, the number of neurons per layer, ...) or discrete (e.g., the adjacency matrix) values, resulting in a complicated problem, i.e., each parameter type require a different optimization approach. Third, the evaluation of an architecture is extremely resource and time-consuming. Therefore, NAS problems fall into the family of expensive optimization problems.

To cope with the latter problem, i.e., the evaluation cost, and aiming to improve reproducibility, lots of effort have been made to provide open-source benchmarks for NAS. Several areas of applied *machine learning* have been included in these benchmarks, including *computer vision* (CV) [11, 29–31] and *natural language processing* (NLP) [32], among others [3]. Also, some authors have explored techniques to speed up the performance evaluation, including learning curve-based estimation [8], one-shot (weight sharing) [33, 34], training-free methods (a.k.a., 0-shot) [9, 35, 36], among others. The mixed search space problem has been faced from multiple perspectives, ranging from tailored encoding [37, 38], and specific operations [25], to mixed (hybrid) approaches [39].

Finally, to tackle the problems that arise due to the size of the search space (first challenge), several authors have invested time tailoring the design of the search space [5, 6], providing tools to assess its *quality* [40, 41], and proposing techniques to adapt the search space [7, 42], among others. Despite all the advances made in this regard, the initialization of NAS algorithms (especially the population-based ones) has not received much attention. However, it is important to remark that starting from a set of *good* solutions is key to solve large-scale optimization problems using a population of finite size [43].

2.2 Population initialization techniques

All population-based metaheuristic algorithms share a common step: The population initialization. The goal of this initialization is to provide a first set of solutions, that (normally) will be improved in an iterative way until the termination criteria is met. A *good* (or *bad*) initial population facilitates (or prevents) finding the optima [43–45], and this is especially true for large-scale optimization problems that use a small population size [43], which is the most common case. Therefore, the greater the search space (given a limited population size), the smaller the chance to cover promising regions of the search space [46].

In the past few decades, some authors have started to propose initialization techniques aiming to boost the performance of population-based metaheuristic algorithms (mainly EA) [47, 48]. Great advances have been made, for example, [49] shows that initialization can increase the probability of finding global optima, [50] shows that stability can be improved, and in [51] it is show that the solution quality is related to the initialization, among many others [47, 48].

However, in black-box optimization problems, such as NAS, it is not possible to determine beforehand what is a good and bad solution. Therefore, not all initialization techniques are suitable for NAS. Moreover, few practical rules of thumb are provided in the literature to choose an appropriate initialization technique. Thus, from a practitioner perspective, it is unclear how to choose the right initialization technique [47].

Considering all these limitations, most practical NAS implementations rely on (quasi)random [49] initialization or Latin hypercube sampling (LHS) [7, 52–54]. Therefore, in this study, we propose to tackle the population initialization problem for NAS.

2.3 Baseline algorithms

Next, we introduce three important population-based algorithm, that we consider as baselines for the benchmark of initialization techniques. Particularly, a Genetic Algorithm (GA), an Evolutionary Algorithm (EA), and the *Aging Evolution* (AE) [6], a popular EA-based algorithm specifically designed for NAS on CV problems.

2.3.1 Genetic algorithm

A GA is a population-based meta-heuristic algorithm inspired by natural evolution [13]. At a glance, a *population* of *individuals* (a.k.a. solution) is evolved using selection, crossover, mutation, and replacement operations. Particularly, we use the implementation available in the latest version (1.3.1) of DEAP library [55]. Algorithm 1 presents a high-level view of the implemented GA.

```
input: The size of the population $pop\_size$, the crossover probability $cx\_p$, mutation
probabilities $mut\_p$ and $mut\_i$, and the maximum number of evaluations
$max\_evaluations$. Alongside with $train$, $validation$, and $test$ data sets.
SOLUTION $\leftarrow \emptyset$
POPULATION $\leftarrow$ Initialize($pop\_size$)
Evaluate(POPULATION, $train$, $validation$)
EVALUATIONS $\leftarrow pop\_size$
SOLUTION $\leftarrow$ Best(SOLUTION, POPULATION)
while EVALUATIONS $\leq max\_evaluations$ do
    OFFSPRING $\leftarrow \emptyset$
    for $j \leftarrow 1$ to $pop\_size$ do
        PARENT_1 $\leftarrow$ BinaryTournament(POPULATION)
        PARENT_2 $\leftarrow$ BinaryTournament(POPULATION)
        CHILD $\leftarrow$ SinglePointCrossover(PARENT_1, PARENT_2, $cx\_p$)
        OFFSPRING $\leftarrow$ OFFSPRING + CHILD
    end
    OFFSPRING $\leftarrow$ Mutate(OFFSPRING, $mut\_p$, $mut\_i$)
    Evaluate(OFFSPRING, $train$, $validation$)
    POPULATION $\leftarrow$ OFFSPRING
    SOLUTION $\leftarrow$ Best(SOLUTION, POPULATION)
    EVALUATIONS $\leftarrow$ EVALUATIONS + $pop\_size$
end
PERFORMANCE $\leftarrow$ Evaluate(SOLUTION, $train$, $test$)
return SOLUTION, PERFORMANCE
```

Algorithm 1 Genetic algorithm.

Particularly, an individual encodes a neural network architecture (in the given search space) by a mix of binary entries, that represent the adjacency matrix of the architecture, and categorical values, that correspond to the operations on the edges of the adjacency matrix. Please refer to Section 4.1 for more details.

An initial *population* of size *pop_size* is initialized by the function `Initialize(·)`. Particularly, we define three variations to initialize the population: Random initialization, LHS, and our proposed method (Section 3). An *individual* is evaluated by the function `Evaluate(`$\mathcal{D}_1$`,` $\mathcal{D}_2$`)`. The decoded architecture is trained using SGD on $\mathcal{D}_1$ data set, and evaluated (accuracy) on $\mathcal{D}_2$ data set (a.k.a., the fitness). Then, the best *solution* (i.e., the one with the highest accuracy) of the *population* is selected by the `Best(·)` function.

Then, the evolution takes place. First, an *offspring* of size *pop_size* is created. More specifically, each *offspring* individual is created by a single point crossover operation `SinglePointCrossover(`$\mathcal{P}_1$`,` $\mathcal{P}_2$`, cx_p)` with probability *cx_p*, where $\mathcal{P}_i$ is selected using a binary tournament operation `BinaryTournament(·)`. Note that with probability 1 - *cx_p* one of the parents $\mathcal{P}_i$ is returned unmodified. Later, the *offspring* is mutated with probability *mut_p* by the function `Mutate(·)`. If mutated, each position is mutated using bit-flip (for the binary entries) or round-robin (for the categorical values) with probability *mut_i*. The *offspring* is evaluated using `Evaluate(·)`, and the current *population* is replaced by the *offspring*. Finally, the best *solution* is updated, i.e., if the fitness of the best individual in the *population* is higher than the current best *solution*, then the best individual become the best *solution*. Once the number of *max_evaluations* is reached, the best solution is evaluated using the *test* data set.

2.3.2 $(\mu + \lambda)$ Evolutionary algorithm

The $(\mu + \lambda)$EA [12], a generic population-based metaheuristic algorithm, evolves a population of μ individuals by creating λ offspring. Then, both the original population and the offspring are combined, and the best μ individuals replace the population. Algorithm 2 presents a high-level view of the $(\mu + \lambda)$EA basic implementation provided by the latest version (1.3.1) of DEAP library [55].

The *population* (refer to Section 2.3.1) of size μ is initialized using the `Initialize(·)` function. Then, the *population* is evaluated using the `Evaluate(·)` function (refer to Section 2.3.1). Then, the evolutionary process takes place. First, an *offspring* of size λ is generated by randomly sampling (with uniform probability) individuals from the *population*. Following, the *offspring* is mutated using the `Mutate(·)` function (refer to Section 2.3.1), and the offspring is evaluated. In the last evolutionary step, the *population* and the *offspring* are combined, ranked according to their fitness, and the top μ individuals are selected by the `RankSelection(·)` to replace the current *population*.

Once the number of evaluations is greater than *max_evaluations*, the best individual of the *population* is selected by `Best(·)`, i.e., the *solution*. Finally, the *solution* is evaluated on the *test* data set.

2.3.3 Aging evolution

A few years ago, the *Aging Evolution* (AE) [6], an *EA*-based approach to NAS, became popular because it achieved state-of-the-art performance on classical CV benchmarks.

```
input: The size of the population $\mu$, the size of the offspring $\lambda$, mutation probabilities
$mut\_p$ and $mut\_i$, and the maximum number of evaluations $max\_evaluations$.
Alongside with $train$, $validation$, and $test$ data sets.
POPULATION $\leftarrow$ Initialize($\mu$)
Evaluate(POPULATION, $train$, $validation$)
EVALUATIONS $\leftarrow \mu$
while EVALUATIONS $\leq max\_evaluations$ do
    OFFSPRING $\leftarrow$ RandomSample(POPULATION, $\lambda$)
    OFFSPRING $\leftarrow$ Mutate(OFFSPRING, $mut\_p$, $mut\_i$)
    Evaluate(OFFSPRING, $train$, $validation$)
    POPULATION $\leftarrow$ RankSelection(POPULATION + OFFSPRING, $\mu$)
    EVALUATIONS $\leftarrow$ EVALUATIONS + $\lambda$
end
SOLUTION $\leftarrow$ Best(POPULATION)
PERFORMANCE $\leftarrow$ Evaluate(SOLUTION, $train$, $test$)
return SOLUTION, PERFORMANCE
```

Algorithm 2 $(\mu + \lambda)$ Evolutionary algorithm.

Algorithm 3 outline AE. Notice that the nomenclature does not match exactly the one proposed in [6], instead the algorithm presents a version that is *closer* to Algorithms 1 and 2. Also, we used the implementation available on NASBench-101 repository.

```
input: The size of the population $pop\_size$, the size of the tournament $k$, and the
maximum number of evaluations $max\_evaluations$. Alongside with $train$, $validation$,
and $test$ data sets.
SOLUTION $\leftarrow \emptyset$
POPULATION $\leftarrow$ Initialize($pop\_size$)
Evaluate(POPULATION, $train$, $validation$)
SOLUTION $\leftarrow$ Best(SOLUTION, POPULATION)
EVALUATIONS $\leftarrow pop\_size$
while EVALUATIONS $\leq max\_evaluations$ do
    OFFSPRING $\leftarrow$ Tournament(POPULATION, $k$)
    OFFSPRING $\leftarrow$ Mutate(OFFSPRING)
    Evaluate(OFFSPRING)
    SOLUTION $\leftarrow$ Best(SOLUTION, OFFSPRING)
    Enqueue(POPULATION, OFFSPRING)
    Dequeue(POPULATION, OFFSPRING)
    EVALUATIONS $\leftarrow$ EVALUATIONS + 1
end
PERFORMANCE $\leftarrow$ Evaluate(SOLUTION, $train$, $test$)
return SOLUTION, PERFORMANCE
```

Algorithm 3 Aging evolution.

AE is a steady state EA, where the *oldest* individual of the population is replaced by the offspring. Particularly, the population of size *pop_size* is initialized using the function

`Initialize(·)`, evaluated using the function `Evaluate(·)` (we are *reusing* the function defined above in this section), and the best *solution* is selected from the population by the function `Best(·)`.

Then, the evolution begins. First, an individual (i.e., the *offspring*) is selected using a k tournament selection. Second, the *offspring* is mutated by a two-step process `Mutate(·)`: (i) a *hidden state mutation*, the connections between operations in a graph-represented solution (cell) are modified, and (ii) an *operation mutation*, the operation within the *cell* is modified. Then, the *offspring* is evaluated, and if its performance is higher than the previous best seen *solution*, the solution is replaced by the *offspring* using the function `Best(·)`, In the last step of the evolution, the *oldest* individual of the population (i.e., the earliest evaluated one in the population) is replaced by the *offspring* using the `Enqueue(·)` (add the new one) and `Dequeue(·)` (remove the oldest one) functions. The authors of AE claim that exists a parallel between the introduced age-based removal to a *regularization* of the evolution.

The evolution continues until the number of evaluated candidate solutions exceed the predefined budget *max_evaluations*. Finally, the best *solution* of the population is returned.

3 A data-driven approach to initializing a NAS search strategy

This section introduces our search initialization technique. First, we describe the overall pipeline of the methodology. Second, we detail the *feature engineering* of the proposed approach.

3.1 Pipeline

This study sets out to answer the following question: *Can we improve the convergence of a population-based NAS algorithm by initializing it with a data-driven approach?*

To address this issue, we propose a data-driven initialization technique, depicted in Fig. 1 and formally described in Algorithm 4. Particularly, we propose to perform a cluster analysis on a randomly selected set of solutions. Then, we propose to use the centroids to initialize the population of a population-based NAS algorithm.

Let us consider a machine learning task and a search space Ω (i.e., a set of neural architectures). Then, let α be a clustering algorithm, and μ the size of the population to retrieve. First, (*i*) we sample a set of N architectures (`RandomSampling(·)`), were N comes from

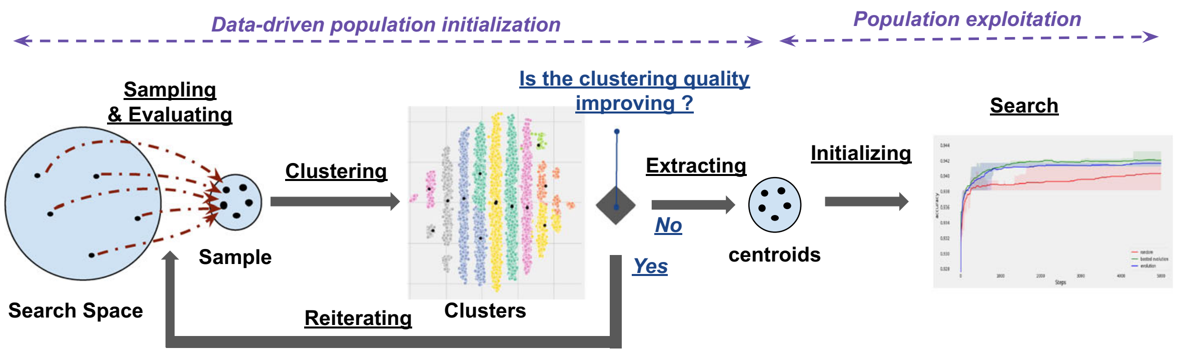


Fig. 1 The pipeline of the proposed data-driven approach to search initialization

a predefined list of sample sizes (*list_sample_size*). Then, (*ii*) we train and evaluate the performance of all sampled architectures (`EvaluateModel(·)`). Later, (*iii*) we encode (`EncodeModel(·)`) each solution using the desired feature representation ε (for more details, refer to Section 3.2). Note that as the feature vector consists of an architecture representation and its performance in test (see Table 1), the resulting clusters should relate to specific behaviors (performances) on the learned task.

Then, (*iv*) we reduce the dimension of the input features (`ReduceDimensions(·)`), as processing high dimensional and sparse data could be extremely challenging. Following, (*v*) we assign a cluster to all samples (`ClusterFeatures(·)`) using the desired clustering technique α and a predefined number of clusters μ (a.k.a., the population size). In the following step, (*vi*) we assess qualitatively and quantitatively the results using the metrics `M` (`Assess(·)`). If the quality of the new clustering is not improving the quality achieved so far, the process is stopped. Otherwise, the `clusters` are stored, a new sample size is retrieved from *list_sample_size*, and the whole process is repeated.

Once the termination criteria is met, i.e., all the sample sizes (from *list_sample_size*) have been evaluated *or* the clustering quality is not improving (refer to `Assess(·)`), the `centroids` of the `clusters` are extracted (`ExtractCentroids(·)`) and decoded (`DecodeModel(·)`).

Finally, the set of decoded centroids, i.e. the set of architectures that correspond to the centroids of the clusters (`init_architectures`), is returned. Then, the idea is to use this set of architectures (`init_architectures`) to initialize a population-based NAS algorithm.

To summarize, our data-driven initialization approach is:

- *Composite*: It is a multistep initialization procedure relying on sampling a search space, clustering it, and initializing an algorithm with the centroids extracted.
- *Generic*: It is not application-specific, in fact the clustering could be done on any type of search space given an encoding including a solution representation and its fitness evaluation.
- *Stochastic*: The stochasticity of the procedure depends on the randomness of the tool selected for clustering.

Also, it is important to note that the clusters may be analyzed to obtain insights regarding the *archetypes* (i.e., the representative architectures), including the most frequent operations and the connection between the operations (i.e., the edges in the graph).

Besides, our method does not require a target search baseline to be adapted. It can be seen as an external and complementary module helping augment a pre-existing algorithm. Indeed, an algorithm to be initialize would receive an initial population of desired sized, and ready to be used for deployment.

Table 1 Description of the feature representation encoding the solutions of the search space

Feature representation	Components
Short	Adjacency matrix + operations + fitness
Long	Expanded adjacency matrix + operations + fitness

Each component is in the form of a list, and the '+' symbol refers to the concatenation operator. The fitness component consists in the list of fitness measured for the various training budget available for a given solution

```
input: The desired population size $\mu$, a list of sample sizes $list\_sample\_sizes$ (decreasing
order), the feature encoding type $\varepsilon$, a clustering algorithm $\alpha$, the list of metrics to evaluate
the clustering quality $M$, and a search space $\Omega$.
CLUSTER_QUALITY ← $\emptyset$
for SAMPLE_SIZE ∈ LIST_SAMPLE_SIZES do
    SAMPLE ← RandomSampling($\Omega$, SAMPLE_SIZE)
    SAMPLE ← EvaluateModel(SAMPLE)
    ENCODED ← EncodeModel(SAMPLE, $\varepsilon$)
    REDUCED ← ReduceDimensions(ENCODED)
    LABELS ← ClusterFeatures(REDUCED, $\alpha$, $\mu$)
    NEW_CLUSTER_QUALITY ← Assess(REDUCED, LABELS, $M$)
    if CLUSTER_QUALITY > NEW_CLUSTER_QUALITY then
        break
    end
    CLUSTER_QUALITY ← NEW_CLUSTER_QUALITY
    CLUSTERS ← (ENCODED, LABELS)
end
CENTROIDS ← ExtractCentroids(CLUSTERS)
INIT_ARCHITECTURES ← DecodeModel(CENTROIDS, $\varepsilon$)
return INIT_ARCHITECTURES
```

Algorithm 4 Data-driven initialization technique.

3.2 Feature representation

To best take advantage of information about the search space when clustering, we first introduce a minimal feature engineering.

As we look to uncover models and structures relevant to NAS algorithms via clustering, we seek a feature representation encoding an architecture as well as its performances. As in [10], we consider neural architectures identified by an elementary component repeated in blocks, a feed-forward cell. This cell is a directed acyclic graph (DAG), with a maximum number of operations (nodes), a maximum number of transformations (edges) and a fixed set of possible operations (e.g., max pool, convolution 3x3) labeling each node. A cell is in practice represented as a list of selected operations and an adjacency matrix of variable size.

In order to use such data in the proposed pipeline, we construct two versions of clustering feature representation, briefly described in Table 1. The first one (**Original**, or *Short Encoding*) consists in concatenating for each model, its adjacency matrix, the list of operations, and the list of performances in test for all available training duration $\{t_0, t_1, t_2, t_3\}$. Note that this is a variable length feature representation due to the nature of the adjacency matrix.

Alternatively, the second representation (**Binary**, or *Long Encoding*) corresponds to using the expanded adjacency matrix, i.e., the matrix that considers all possible operations (according to the constraints of the search space). This is a fixed length encoding. Figure 2 shows an example of a generic adjacency matrix being tranformed into an expanded one, right before the step of flattening. Additionaly, Fig. 13 (and Table 3) provide a visualization of such matrix for solutions retrieved in various algorithm initialization settings.

Moreover, for both encodings, the vector form of the adjacency matrix is obtained by a flattening in row-major fashion, where consecutive elements in a row are put next to each other, while iterating over the different rows.

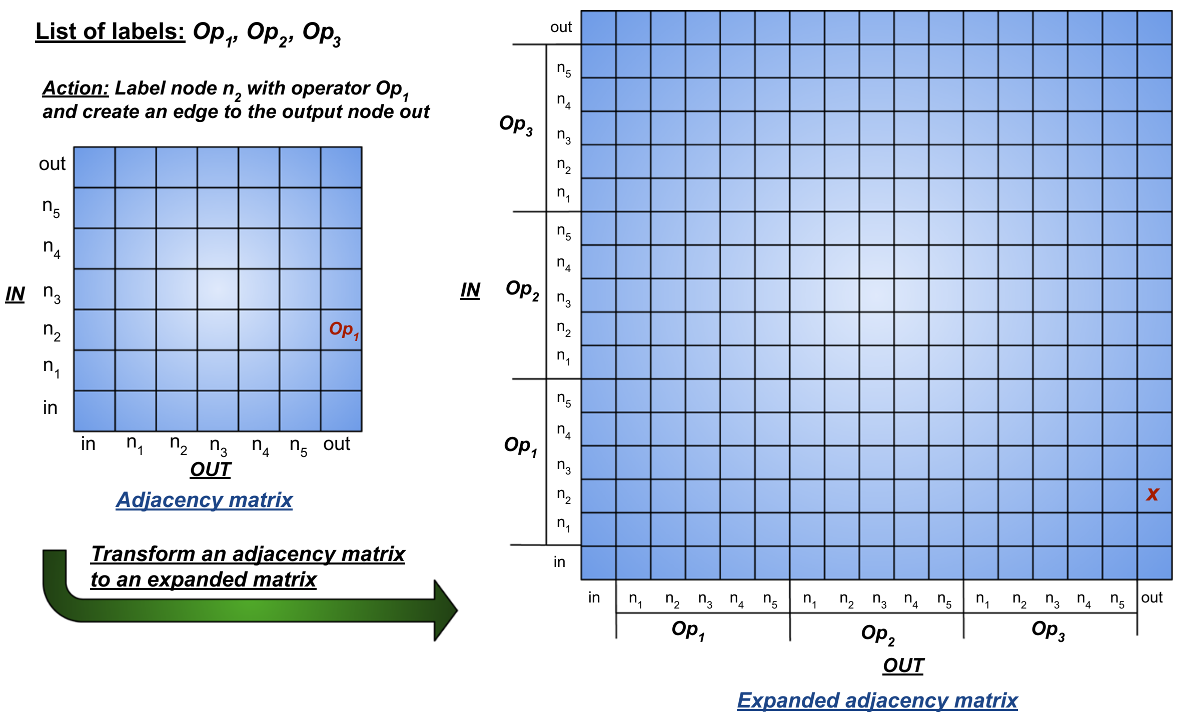

List of labels: Op_1, Op_2, Op_3
Action: Label node n_2 with operator Op_1 and create an edge to the output node out
out
n_5
n_4
IN
n_3
n_2
n_1
in
Op_1
OUT
Adjacency matrix
Transform an adjacency matrix to an expanded matrix
Op_3
Op_2
Op_1
x
OUT
Expanded adjacency matrix

Fig. 2 A visual description of the transformation of a standard adjacency matrix into an expanded adjacency matrix for a neural cell in NASBench-101

4 Experimental setup

The experiments performed aim to validate that the initialization of a population-based NAS Algorithm can benefit from models identified via Clustering Analysis of a search space. In this Section, first, we introduce the problem used to validate our proposal. Second, we present the parameters used for performing the experiments on clustering. Third, we detail the performance metrics used to assess the quality of the clustering.

4.1 NASBench-101

NASBench-101 is a database of neural network architectures and their performance evaluated on the data set of CIFAR-10. It contains $N = 450K$ unique architectures [10]. Indeed, to tackle the given machine learning task of CIFAR-10, all contained models use of a classical image classification structure similar to ResNet. Indeed, the backbone of a model contains a head, a body, and a tail. Its body is made by alternating three (3) times a block with a down-sampling module. Each block is obtained by repeating three (3) times a module called 'cell'. A cell is a computational unit that can be represented by a DAG. It consists of an input node, an output node and intermediate nodes representing operations (convolution 3x3, convolution 1x1, maxpool 3x3), and connections indicating features being transformed. Therefore, each architecture differs by its cell. In practice, the DAG of a model is encoded by an adjacency matrix and a list of operations labelling the associated nodes. The constraints on such DAG are the following: There can be at most $N = 7$ nodes and $E = 9$ edges in a cell.

Moreover, all models were trained for 108 epochs using the same experimental setting (i.e., learning rate, etc.), but performance evaluations in training, validation, and test were also provided after 4, 12 and 36 epochs.

4.2 Hyperparameters for clustering

The clustering experiments were done with a set of randomly sampled models. The size of the sample is identified in Section 5.2. The considered clustering algorithms are k-means, DBSCAN, BIRCH, spectral clustering, and a BGM. All were obtained from the latest version (0.24.1) of the *scikit-learn* library [56]. Table 2 shows the hyperparameters selected for each method, including the maximum number of iterations (*Max iter*), the number of samples used at initialization (*N init*), and other algorithm-specific parameters (*Other*). Note that they are either default (NA) or slightly modified to provide satisfying clustering performances.

4.3 Clustering performance evaluation

Moreover, we use various ways of assessing the quality of the results for each step of the approach. Regarding the preparation of the initialization, we propose to measure the clustering performance using the following three (3) standard metrics: Silhouette coefficient [57], Calinski-Harabasz [58] and Davies-Bouldin index [59]. These metrics inform on how well separated and dense are the resulting clusters. They all apply in the context of clustering with missing labels, which is relevant, as we seek to investigate relevant clusters and features for NAS algorithms without prior assumptions, and well-defined clusters should provide us with centroids (i.e., models) that are good representatives of all the architectures in the search space. The Silhouette coefficient is a metric comprised between -1 and +1, with higher values associated to more dense and separated clusters The Calinski-Harabasz index also rates a better defined clusters with higher values. Similarly, the Davies-Bouldin index, measures a *similarity* between clusters, providing smaller values for better clustering. Additionally, we propose to corroborate the quantitative assessment with a qualitative analysis (visual inspection) for validation before the next step.

Regarding the step of exploitation, we assess the quality of the proposed initialization method using the performance of the search algorithm initialized (best obtained accuracy in test). More importantly, we compare the performance against initializing the same algorithm with random and LHS initialization. Also, as a sanity check, we compare the results against a *random search*.

5 Results

In this section, we present results on clustering for accelerating NAS algorithms. First, we show results on selecting the proper tool for dimension reduction and the hyperparameters

Table 2 Hyperparameters of the clustering algorithms

Method	Max iter	N init	Other
KMEANS	500	50	kmeans++ init
DBSCAN	500	200	eps=0.30
BIRCH	500	NA	threshold=0.12
SPECTRAL	500	NA	NA
BGM	500	NA	Dirichlet weight distribution, full co-variance

for the clustering. Then, we show results on identifying the number of clusters providing satisfying clustering performances. We also present results on qualitatively assessing the clusters quality for various algorithms. Then, we provide results on improving NAS performances using a centroid-based initialization for three (3) NAS evolutionary algorithms. Last but not least, we show results of a quantitative assessment for solutions found from the bench-marking, in the form of matrices of cell occurrence.

5.1 Dimension reduction

To begin our experimental study, we seek to calibrate the dimension reduction of the input features. To do so, we arbitrarily fix the number of samples to $N = 10000$, in order to perform relevant experiments.

Figure 3 shows clustering performance as a function of the number of components of input features. The blue and red curves display performance using respectively the Short (Original) and the Long encoding (Binary). The dimension reduction is performed using PCA, and the clustering with k-means using a fixed number of clusters $N = 10$.

Using the Short encoding, the three metrics are in favor of using a small number of components for input features via PCA. Indeed, the smaller the number of components the higher the Silhouette and Calinski-Harabasz scores, and the lower the Davies-Bouldin index, with optimal values for using two (2) components. The same is observed when using the Long encoding.

Using the Long encoding yields slightly better performance than the Short encoding, with a sensible improvement on larger number of components with PCA.

As both encoding are rather sparse (length of up to 58, or up to 298), we study the effect of this sparsity in the dimension reduction tool. Figure 4 shows clustering performances as a function of the number of components of input features, for various reduction tool. The blue and red curves display performances using respectively PCA and Truncated SVD as dimension reduction tools. Plot (a) and (b) display results using respectively the Short and the Long encoding. The clustering is performed with k-means.

Trying an alternative dimensional reduction tool (Truncated SVD) more suitable for highly sparse data does not worsen results on the Short encoding (see Fig. 4a). Moreover, it allows for a slight improvement over PCA when using the Long encoding (see Fig. 4b).

To summarize, the findings show that reducing the dimensions of the input features to 2D provides the best performances on both encoding. Using the Long encoding improves the results. Also, using Truncated SVD shows slight improvements as it is more suitable for sparse data. Given these findings, the following experiments are performed using Truncated SVD for a 2D reduction of input.

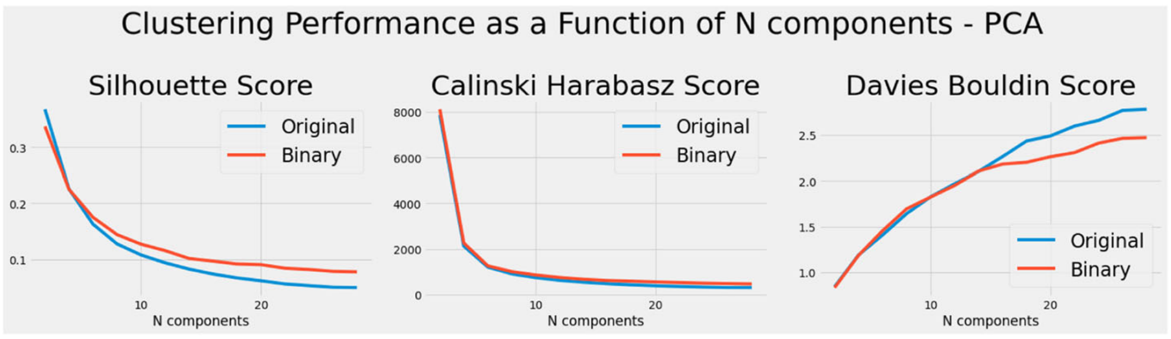
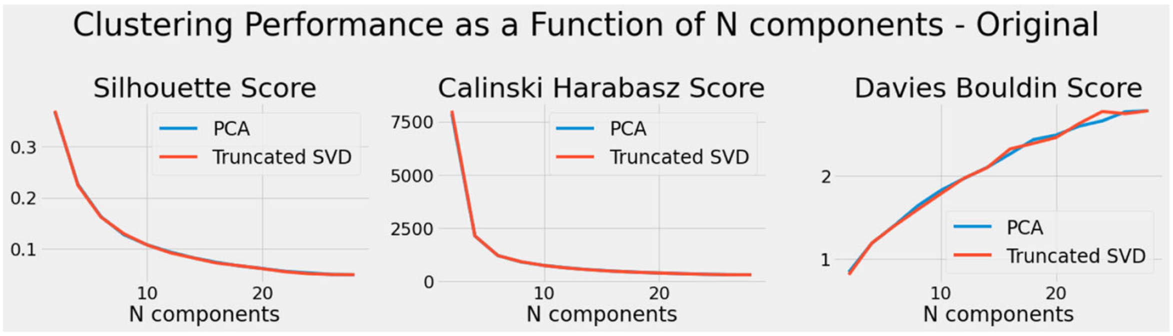
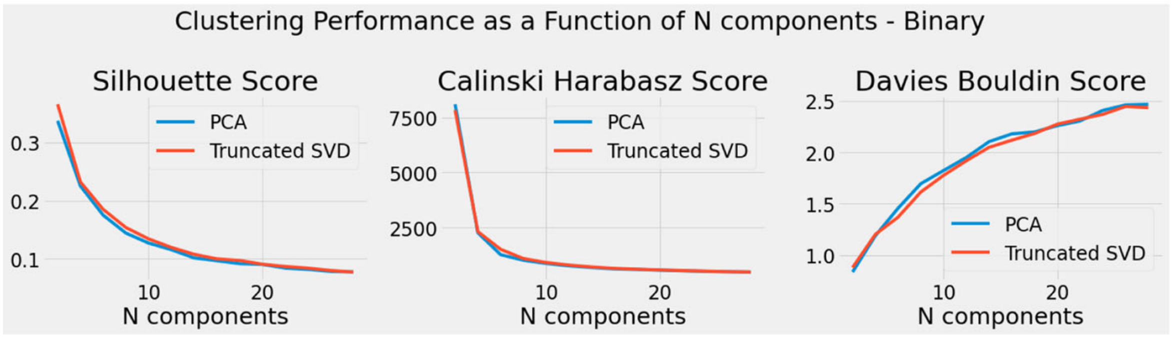


Fig. 3 Input feature reduction for clustering, with an arbitrary number of clusters $N = 10$



(a) Using the Short encoding



(b) Using the Long encoding

Fig. 4 Identifying the proper reduction tool for sparse data, using an arbitrary number of clusters of $N = 10$

5.2 Number of samples

Having identified a suitable dimension reduction tool (Truncated SVD) and value for the number of components to reduce to (N=2), we now seek to find a satisfying number of samples for the clustering.

Figure 5 shows clustering performances as a function of the number of clusters, for various sample sizes. All were obtained when clustering with the Short encoding for feature representation. The blue, orange, green, red, purple and brown curves are for respectively using 100, 500, 1000, 2500, 5000 and 10000 random samples. This range of values enables us to consider small to intermediately large complexity for our proposal.

Overall, all performance metrics points towards the use of large number of clusters. Indeed, the higher the number of clusters, the higher the Silhouette and Calinski-Harabasz

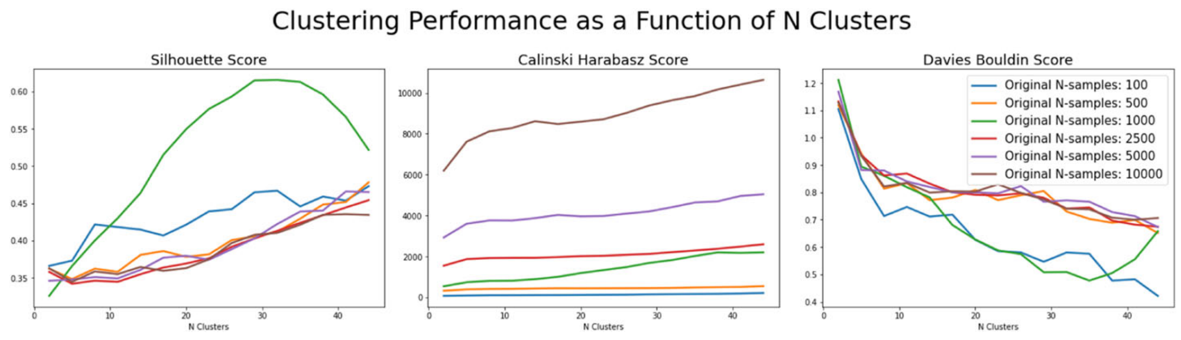


Fig. 5 Identifying the proper sample size, when clustering using Truncated SVD for dimension reduction ($N = 2$ components) and k-means

scores, and the lower the Davies-Bouldin index. These trends are observed for all sample sizes. Moreover, for both the Silhouette score and the Davies-Bouldin index, the plotted functions are of similar values (overlap of clustering score as function of the number of clusters) for most sample sizes. Only the Calinsky-Harabasz index discriminates towards the use of increasingly larger sample sizes.

Given these findings, we identify $N = 10000$ (the largest tested value) as the sample size to use for optimal clustering in future experiments.

5.3 Number of clusters

Next, we look closer into the number of clusters to use, when clustering with various feature representations.

Figure 6 shows clustering performance as a function of the number of clusters. The blue and red curves display the performance results using respectively the Short and the Long encoding. All input features were reduced to two (2) components using Truncated SVD, and the clustering is performed with k-means.

Using both encodings, all performance metrics point towards the use of large number of clusters. The higher the number of clusters, the higher the Silhouette and Calinski-Harabasz scores, and the lower the Davies-Bouldin index. Additionally, the clustering performances seem to reach a plateau for intermediate values of number of clusters between twenty (20) and thirty (30). This is observed when considering the Calinski-Harabasz score and the Davies-Bouldin index, and for both encodings. Then, the performance slightly improves, when the number of clusters exceeds this range. In practice, we found in Section 5.4 that for a queried number of clusters greater or equal to twenty (20), the number of retrieved clusters using the *scikit-learn* toolbox is of $N_{short} = 19$ and $N_{long} = 13$ when using, respectively, the Short and Long Encodings.

Therefore, results suggest using an intermediate- to-large number of clusters for improving the k-means clustering performances, with a preference for the Short encoding. Therefore, we set the number of clusters to $N_{short} = 19$ and $N_{long} = 13$ when using, respectively, the Short and Long Encodings for the following experiments.

5.4 Qualitative cluster analysis

As an additional way to validate the clustering results, we seek to visualize the clusters, and compare them to the natural layout of the reduced data.

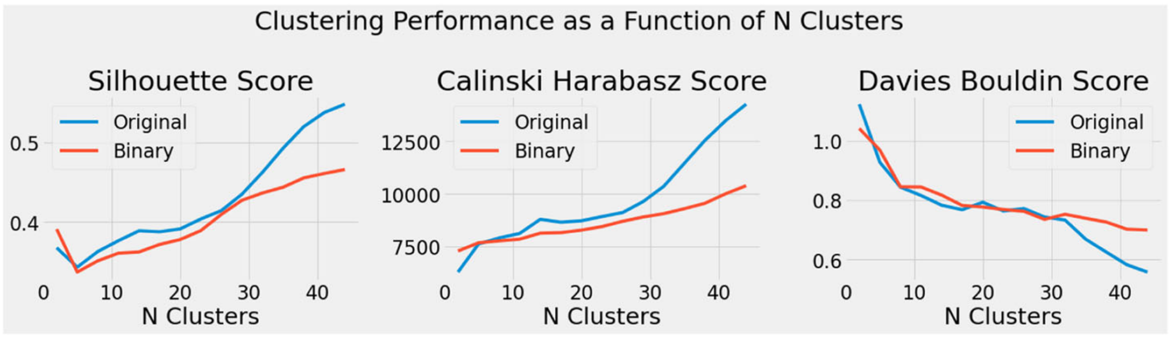


Fig. 6 Identifying the proper number of clusters, using Truncated SVD for dimension reduction ($N = 2$ components) and k-means

Figure 7 depicts visual clustering results for five algorithms: k-means, spectral clustering, DBSCAN, Birch, and BGM. All input features were reduced to two (2) components using Truncated SVD. Plots (a) and (b) display results using the Short and the Long encoding, respectively.

When using the Short encoding (Fig. 7a), the clusters seem to have natural horizontal to diagonal (45 degree) layout. This layout is not well captured by the evaluated algorithms. The BGM seems to provide the most satisfying results, despite little calibration.

When using the Long encoding (Fig. 7b), clusters naturally layout in well separated vertical columns. This is also best captured by BGM.

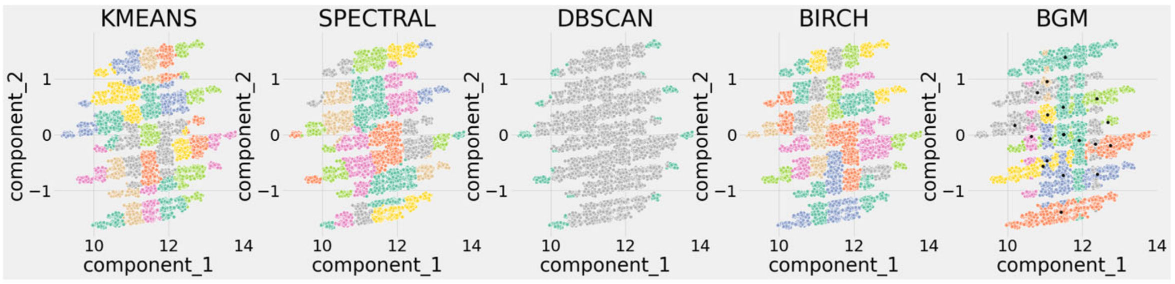
Overall, results suggest using BGM for robust clustering on both feature representations.

5.5 Initialization benchmark

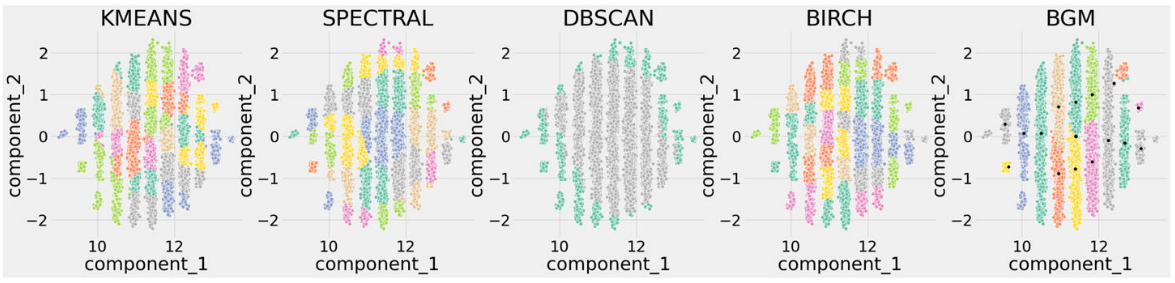
5.5.1 CIFAR-10

In order to assess the quality of the centroids extracted, we use them for initializing the baselines algorithms GA, $(\mu + \lambda)$EA, and *Aging Evolution*.

Figure 8 shows performance in validation for the three search baselines. The color red stands for the random sampling initialization (*rand*), blue for LHS, and green for centroids (i.e., our approach). From left to right, the GA, EA, and Aging Evolution results are plotted. The top row corresponds to 36 epochs of training, and the bottom one to 108 epochs. In all cases, each algorithm is executed 100 independent times. Each plot provides with the mean fitness of the current population (bold), complemented with the range of fitness (min/max). The centroids are initialized considering the Short encoding and BGM previous results. The population size is set to 19 (i.e., the number of centroids) in all cases (refer to Section 5.3).



(a) Using the Short encoding



(b) Using the Long encoding

Fig. 7 Qualitative analysis of clustering for both feature representations. Here we use Truncated SVD for dimension reduction, and $N = 2$ components

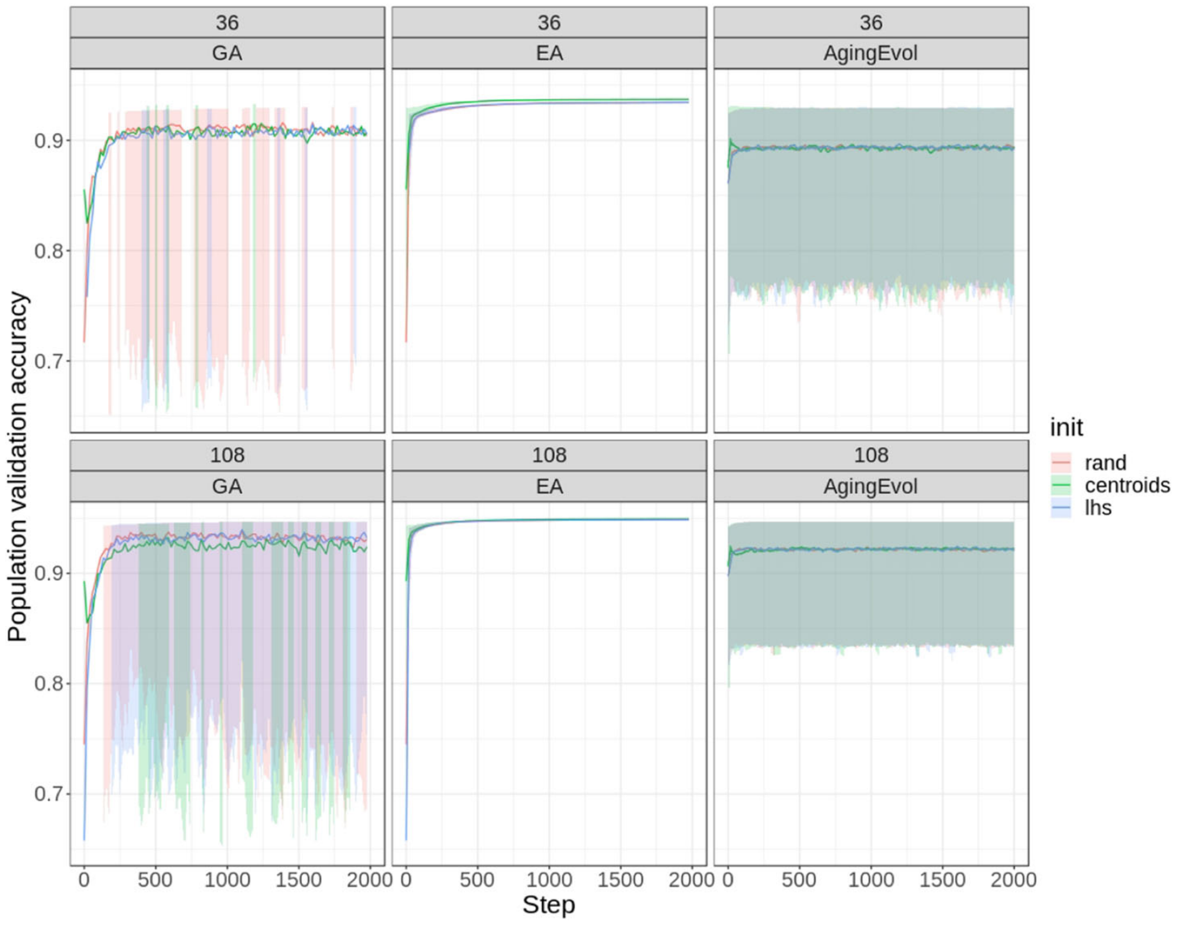


Fig. 8 Performances in validation of various NAS algorithms, when clustering with the Short Encoding

Regarding GA (Algorithm 1), *pop_size*=19, *max_evaluations*=1995 (i.e., population size 19, evolved 104 generations), *mut_p*=0.2, *cx_p*=0.5, and *ind_pb*=0.05. Regarding EA (Algorithm 2), $\mu = \lambda$=19, *mut_p*=0.8, and *ind_pb*=0.1. Regarding Aging Evolution (Algorithm 3), it is run with a tournament size $k = 10$.

For all three baselines, we observe that the centroid-based initialization provides with the highest initial mean population fitness. On the other hand, both LHS and random sampling-based initialization provide with a very low initial mean fitness (up to 20 percentage points of difference). The EA takes the best advantage of this improved initial population: It converges faster and has long-term improvements over an initialization with random sampling or LHS. Both GA and Aging Evolution fail to benefit from such improvements as their mean population fitness plummets after a few iterations, and reaches similar values to those of the alternative initialization techniques (rand or LHS). This is observed when searching either after 36 or 108 epochs of training.

Figure 9 summarizes the benchmark provided in Fig. 8. In particular, it provides box plots of performance in test for the best found solutions (100 runs) after 2000 search evaluations. It also complements the three baseline algorithms, i.e., GA, EA, and Aging Evolution, with random search (RS). The plot on the left corresponds to 36 epochs of training (i.e., the evaluation of the solutions), and the right one to 108 epochs.

Overall, performances in test after deployment (2000 evaluations) are similar to those in validation. Indeed, the ranking is preserved: The EA reaches the highest mean fitness, for all initialization settings. EA and GA have very narrow fitness distributions, while Aging

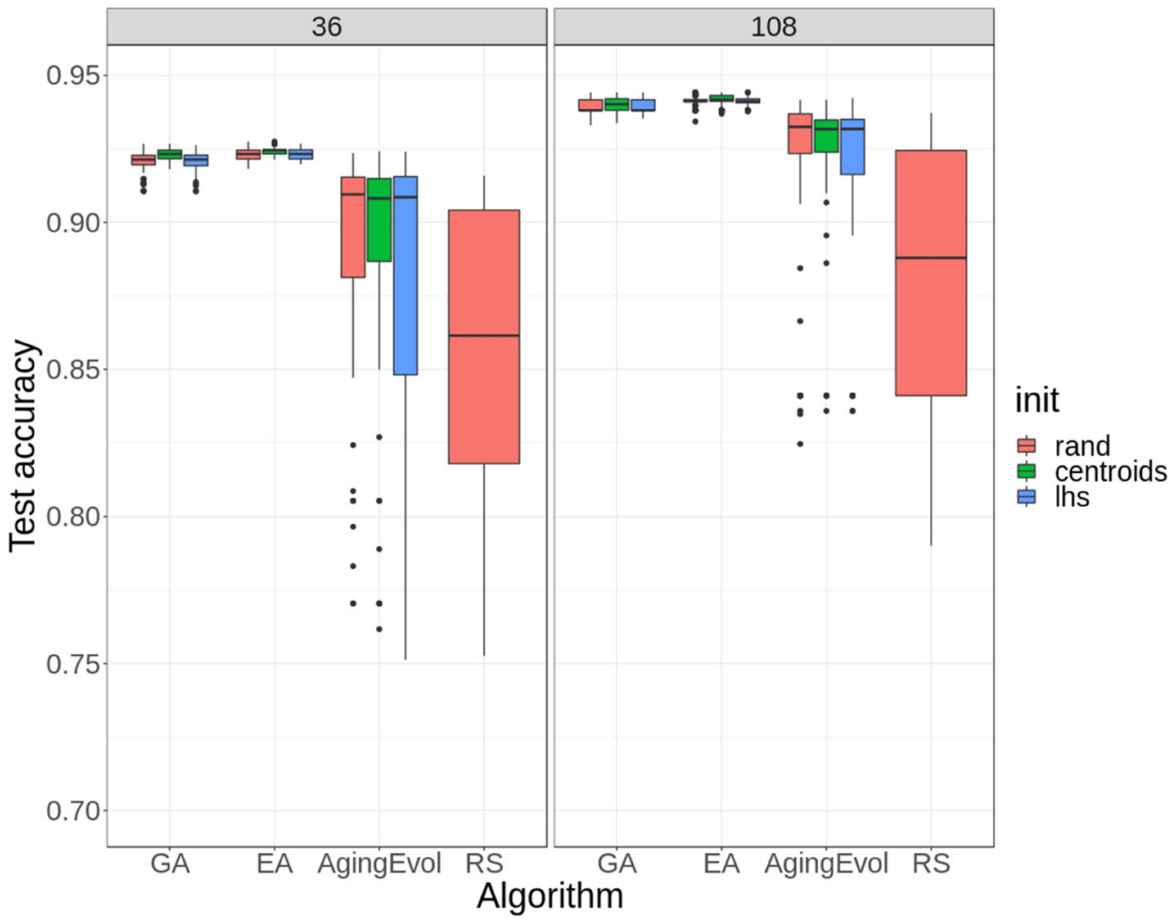


Fig. 9 Benchmark of NAS algorithm performances after 2000 iterations. The search is performed either when training solutions for 36 or 108 epochs. The data-driven initialization techniques involve the Short encoding

Evolution has a more spread one. All the baselines improve over Random Search. For GA and EA, centroids help reach higher mean test fitness over other initialization techniques.

To complement these results, we performed a Wilcoxon rank-sum test. For GA and when selecting after 36 epochs of training, the p-value for the centroid-based initialization versus random sampling is $4.093 \cdot 10^{-7}$. Versus LHS, it is $2.324 \cdot 10^{-7}$. When selecting after 108 epochs, it is 0.69 versus random sampling, and 0.039 versus LHS. For EA and when selecting after 36 epochs of training, the p-value for the centroid-based initialization versus random sampling is $5.611 \cdot 10^{-8}$. Versus LHS, it is $3.767 \cdot 10^{-6}$. When selecting after 108 epochs, it is 0.006 versus random sampling, and 0.006 versus LHS. Thus, the centroid-based initialization significantly improves over LHS and random sampling, for both EA and GA when selecting after 36 epochs of training. For EA, it improves significantly over LHS and random sampling in all training budgets.

Figure 10 depicts the results for Long encoding benchmark. In all cases, we set the pop_size=13 (=$\mu = \lambda$), following the recommendations from Section 5.3. The number of evaluations is set to 1989 (153 generations).

Similarly, centroids obtained considering the Long encoding enable all baseline algorithms to have an improved initial mean population fitness. Also, EA is the best at taking advantage of this initialization (centroids), with an improved convergence, up until 500 to 1000 evaluations.

Figure 11 summarizes the benchmark provided in Fig. 10 with performances in test, in the same fashion as Fig. 9

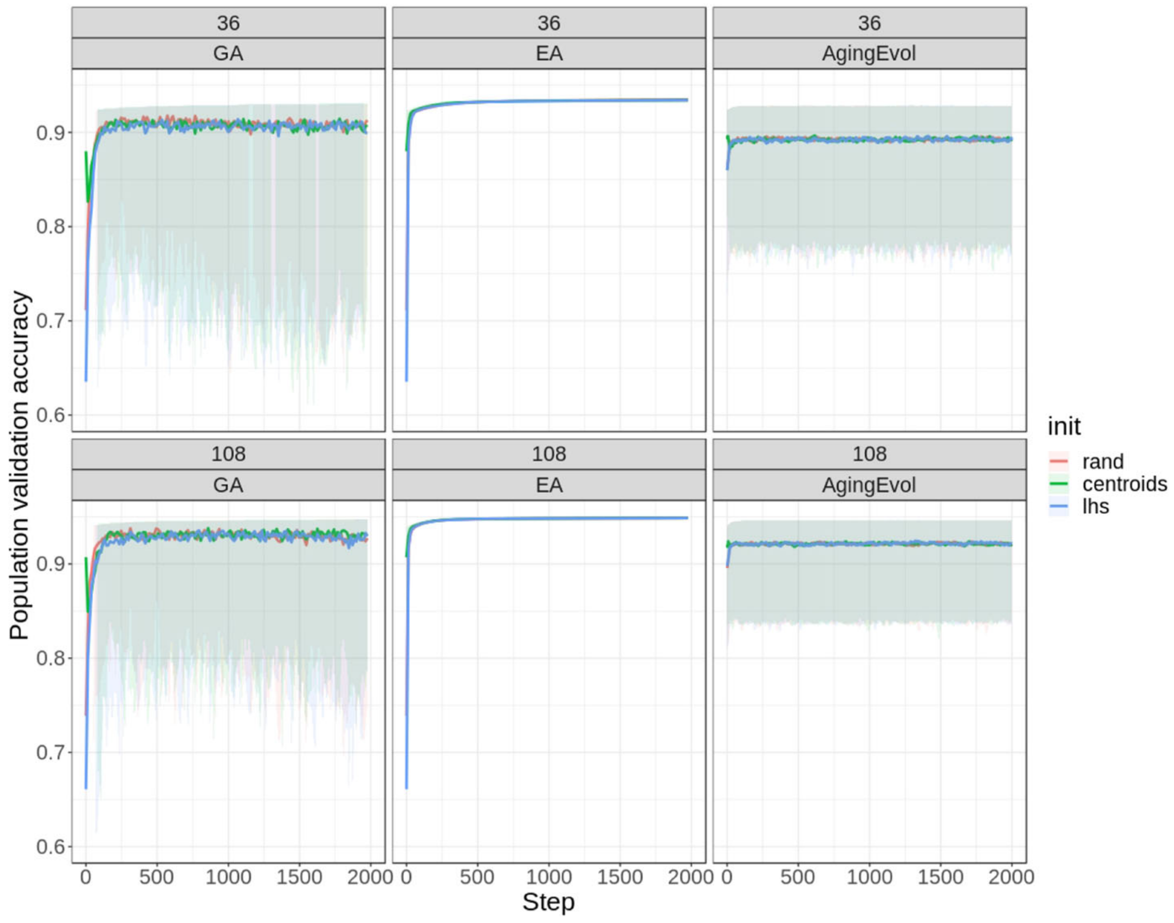


Fig. 10 Performances in validation of various NAS algorithms, when clustering with the Long encoding

Results of performance in test after deployment are similar to those obtained considering the Short encoding for finding the initial population. However, the centroids do not provide with improvements to the final performance of the baseline algorithm. Also, we notice that the centroid-based initialization worsen the distribution of fitness of solutions found by Aging Evolution (i.e., larger variance).

To summarize, the centroids extracted from a fitness-based clustering of the search space seem to be a promising strategy to initialize a population-based search algorithms. We observe improved convergence and long-term performances of EA with a centroid-based initialization, over LHS and rand, when considering the Short encoding. In the case of searching with only 36 epochs of budget, it also helps final test performances for GA (Short encoding).

The limited improvements when clustering with the Long encoding might be explained by the fact that the baselines (EA, GA, and *Aging Evolution*) are deployed on models using the Short encoding. Note that experiments using the Long encoding were discarded because of the increased complexity for the search procedure. Future work might explore this option, as it could help better exploit the extracted population.

5.5.2 So2Sat LCZ-42

Next, we ask ourselves: *can we re-use an initial population provided by the proposed technique, but for other purposes (e.g., datasets)?*

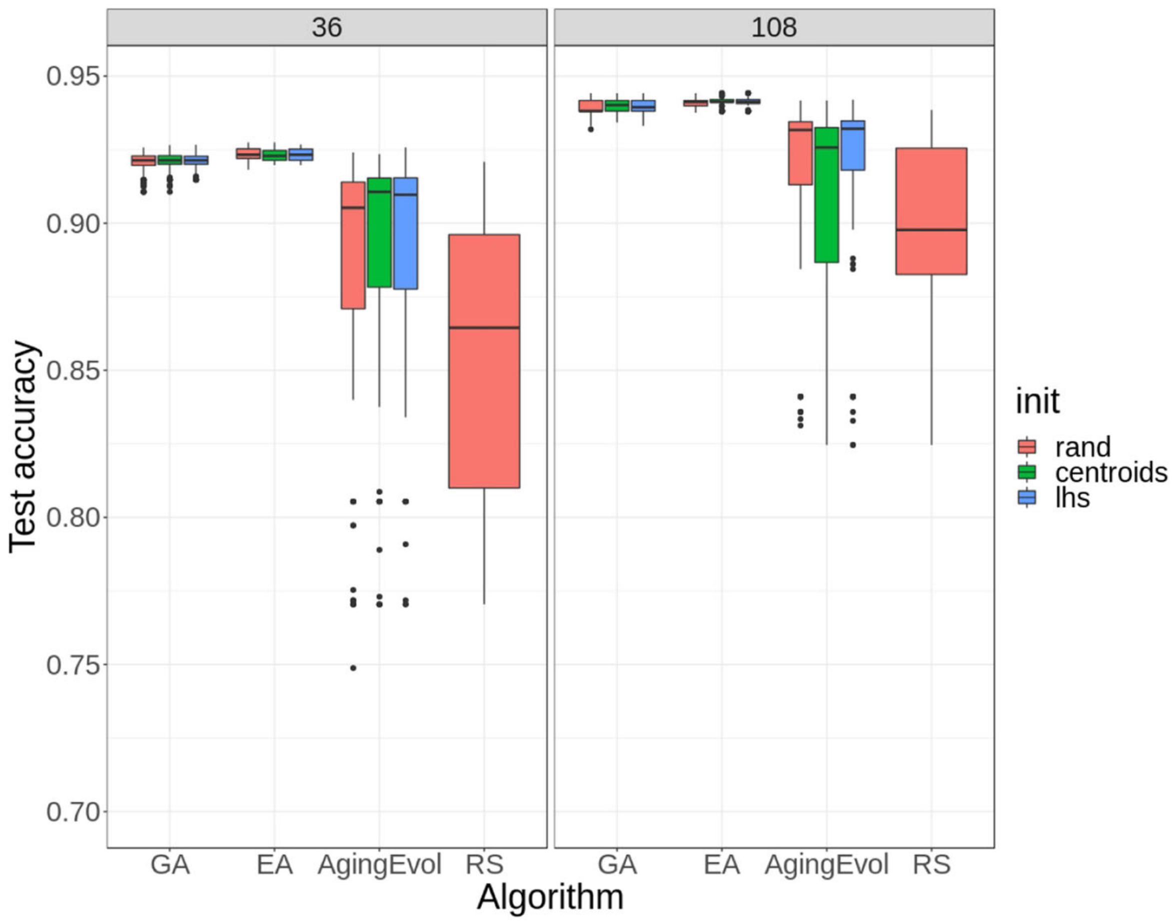


Fig. 11 Benchmark of NAS algorithm performances after 2000 iterations. The search is performed either when training solutions for 36 or 108 epochs. The data-driven initialization techniques involve the Long encoding

To answer this question, we perform experiments using the real-world image classification dataset So2Sat LCZ-42. More precisely, we seek to know if an initial population obtained using as input a sample from NASBench-101, can benefit a search algorithm deployed on the So2Sat LCZ-42 [60]. Given the results obtained in the previous sections and because of practical limitations, we only consider EA for search and random sampling as alternative sampling baseline. Indeed, EA is the algorithm for which the most success is observed using our approach, and random sampling is more stable than LHS. We also limit the number of evaluations to $N_{evaluations} = 100$ and training budget of 4 epochs. Figure 12 shows such results using the Short Encoding, with *pop_size*=19 (=$\mu = \lambda$). Both settings are executed five independent times.

Overall, we observe similar results of convergence on LCZ42 than initially seen on CIFAR-10. Indeed, over all the trials, the mean test accuracy of the initial population provided by the centroids is much larger (about 14 percentage points) than the one obtained by random sampling. Besides, this gap reduces after 1 generation (19 evaluations), but the centroid-based approach remains improving.

To summarize, early results of initializing search on a complementary dataset, indicate that initial population gathered on NASBench-101 also enables to accelerate the convergence of the algorithm on another dataset, even in the case of low training budgets. Since the improvements in fitness increases with more training budget, this suggests that the gap in fitness might even become wider in profit of the proposed method when training longer.

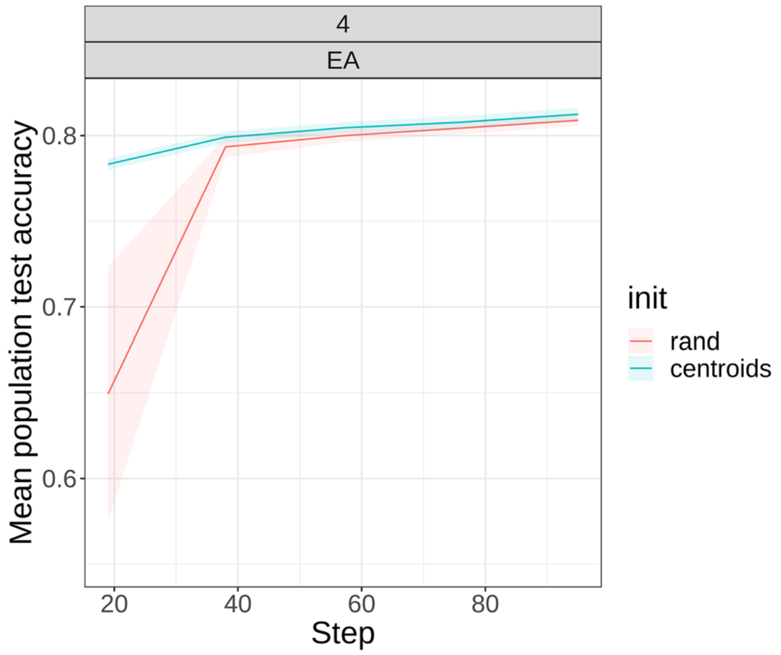


Fig. 12 Performances in test of various initialization techniques used with EA, when clustering with the Short Encoding, on the So2Sat LCZ42. Each curve shows the mean (bold line) and deviation of the performance in test, for its respective setting (five runs)

5.6 Visualization of the solutions found

Last but not least, we look to gain insights into the solutions found by the algorithms deployed in Section 5.5.1.

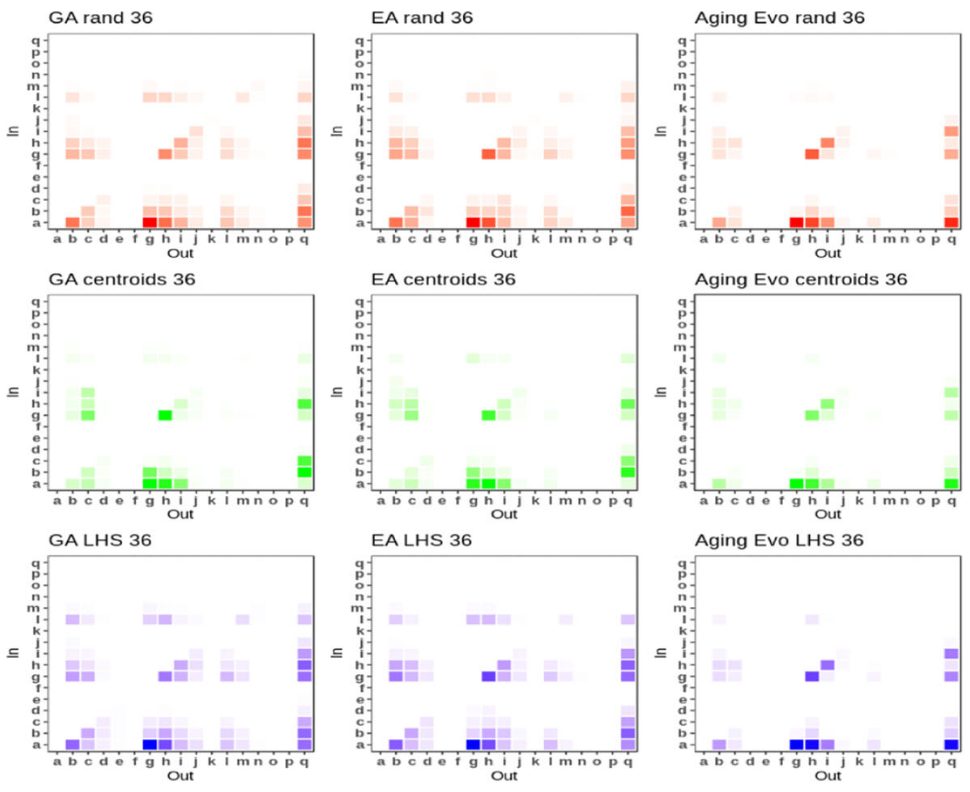
Figure 13 provides a visualization of solutions found (100 independent runs) by the search baselines, for all initialization settings, considering the Short encoding. More precisely, it shows the frequency of connections on the expanded adjacency matrix (100 solutions), for each baseline. The darker, the higher the frequency. Figure 13a and b show results when searching respectively after 36 or 108 epochs of training. From left to right appear results for GA, EA, and Aging Evolution. From top to bottom appear results using as initialization random sampling (rand), centroids, and LHS.

Figure 14 provides the same visualization of solutions found, but considering the Long encoding.

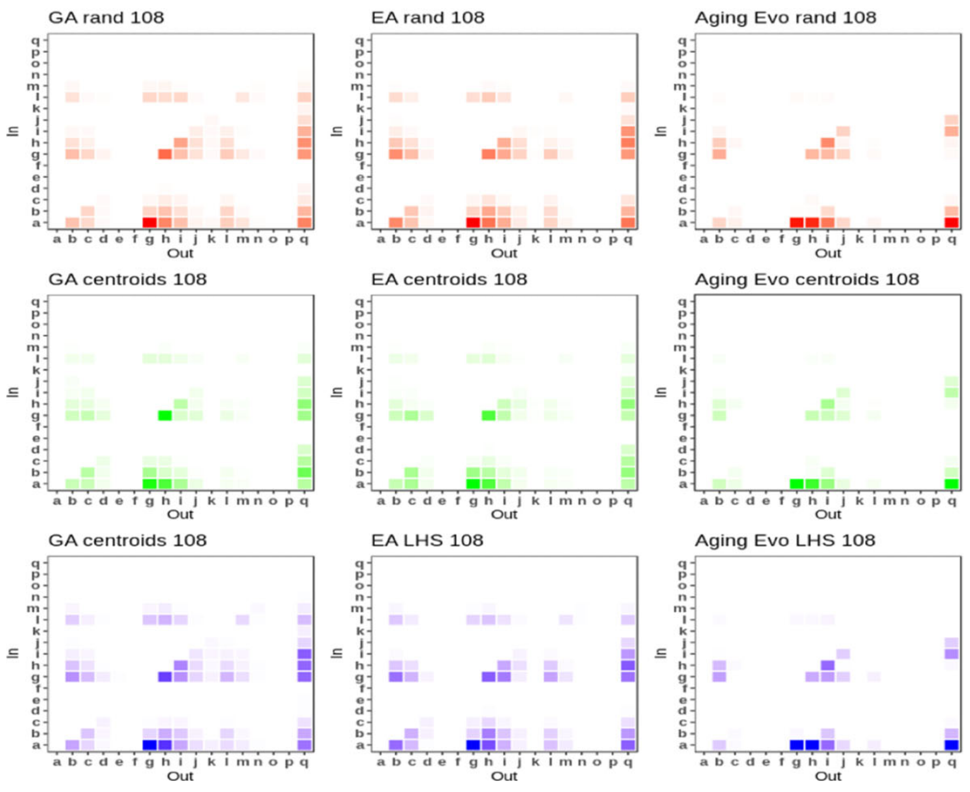
Table 3 provides a legend for the labelling of both Figs. 13 and 14. Note that the matrices shown describe distributions of elementary *feed-forward modules* that are DAGs with *nodes*, *edges* and *node labels*. In each figure, all adjacency matrices are labelled using seventeen characters ranging from *a* to *q*, where each reffers to either the input node (resp. *a*), the output node (resp. *q*), or one of the five possible intermediated nodes (at most) and its node label. Then, the remaining letters are reffering to the intermediated nodes being tagged with either the first (resp. *b* to *f*), the second (resp. *g* to *k*) or third (resp. *l* to *p*) node label. Section 4.1 and Fig. 2 offer explainations on the construction of such matrices.

Overall, the connections gathered from the solutions found after 36 epochs of training differ from those found after 108 epochs. In the first case, the *activations* on the adjacency matrices have clusters that are more restricted, as opposed to the more widespread and larger clusters obtained when searching after 108 epochs of training.

Besides, we also observe a difference in the output based on the algorithm used to find the solutions. EA and GA provide solutions whose connections are overall similar, in the form

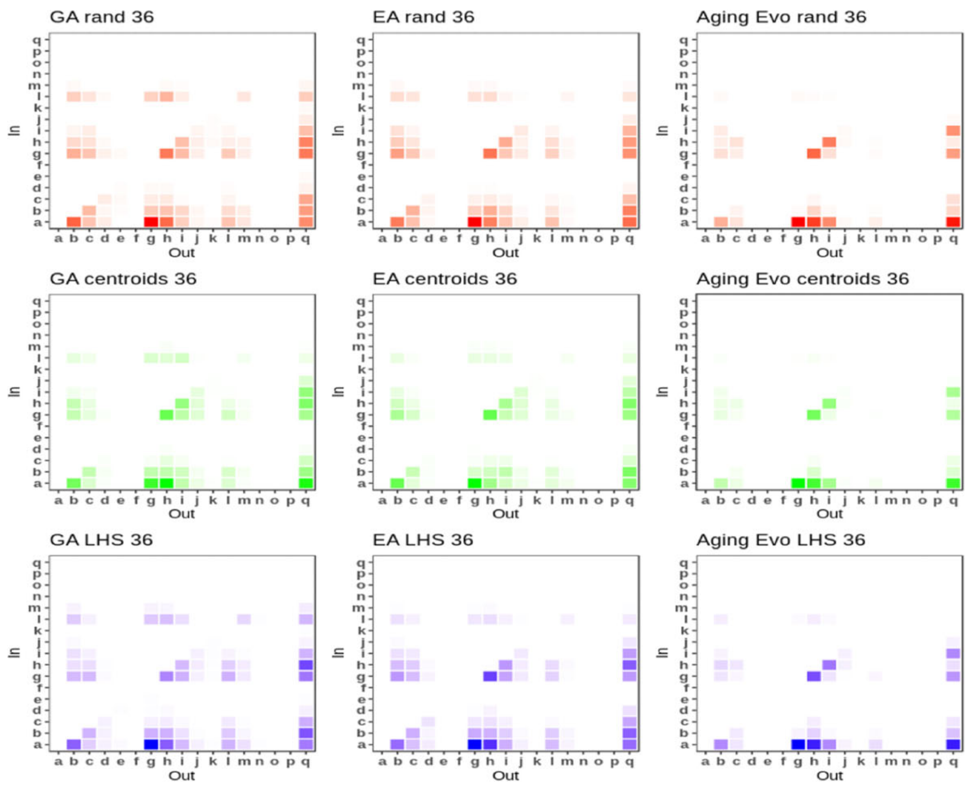


(a) With a budget of 36 epochs

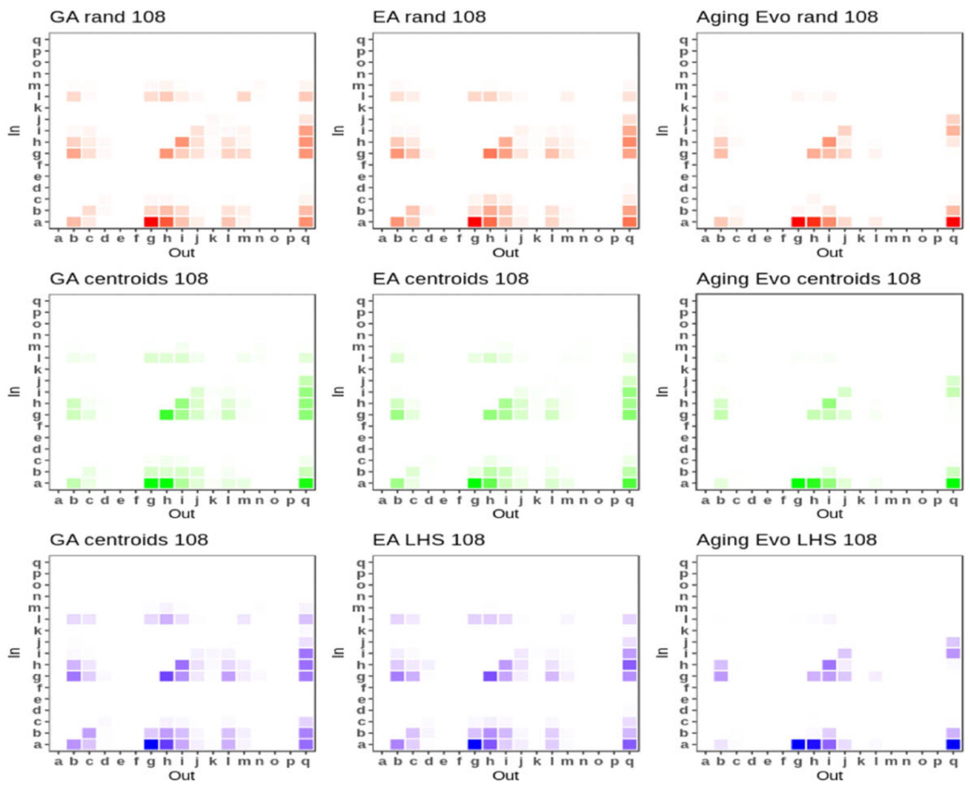


(b) With a budget of 108 epochs

Fig. 13 Visualization of solutions found (N=100) considering the Short Encoding



(a) With a budget of 36 epochs



(b) With a budget of 108 epochs

Fig. 14 Visualization of solutions found (N=100) considering the Long Encoding

Table 3 Legend for Figs. 13 and 14: Each character is associated to a combination of *node label* and *node index* according to the search-space of the NASBench-101 database

Figure label	Node Label	Node index
a	input	0
b	Conv 1x1 + BatchNorm + Relu	1
c	Conv 1x1 + BatchNorm + Relu	2
d	Conv 1x1 + BatchNorm + Relu	3
e	Conv 1x1 + BatchNorm + Relu	4
f	Conv 1x1 + BatchNorm + Relu	5
g	Conv 3x3 + BatchNorm + Relu	1
h	Conv 3x3 + BatchNorm + Relu	2
i	Conv 3x3 + BatchNorm + Relu	3
j	Conv 3x3 + BatchNorm + Relu	4
k	Conv 3x3 + BatchNorm + Relu	5
l	MaxPool 3x3 + BatchNorm + Relu	1
m	MaxPool 3x3 + BatchNorm + Relu	2
n	MaxPool 3x3 + BatchNorm + Relu	3
o	MaxPool 3x3 + BatchNorm + Relu	4
p	MaxPool 3x3 + BatchNorm + Relu	5
q	output	6

of widespread clusters. On the other had, the Aging Evolution has patterns of connections in its cells that are regrouped and in slightly smaller cluster.

Furthermore, we analyzed the solutions based on the initialization technique used when deploying search. Across all settings, it appears to be more diverse solutions (on average more activated cell in adjacency matrices) obtained via LHS and random sampling, than for a centroid-based initialization.

To summarize, the longer the training allowed when selecting models, the more diverse are the solutions retrieved. Also, EA and GA tend to find more diverse solutions than Aging Evolution. When it comes to initialization, the centroids-based approach results in solutions that are more similar to each other, with matrices of adjacency that are less activated.

We find that the patterns highlighted in this section correlate with the findings of the authors in [41]. In the study, the authors show that on the search space of NASBench-101, the longer the training the more narrow the fitness distribution with most solutions having close to the top fitness after 108 epochs of training. They also showed that the fitness landscape becomes flat, with many local optima. Therefore, when searching with a training budget of 108 epochs and a fixed number of iterations, a search algorithm is likely to retrieve more diverse solutions than after 36 epochs, since most of them satisfy the criterion of high fitness.

When it comes to the differences based on the algorithm to be used, this could be explained both the very rugged landscape (many local maxima) and the nature of the algorithms. Indeed, as Aging Evolution provides with non-diverse sets of solutions, which could be explained by it being stuck in local maxima and not diversifying enough, i.e., discarding *old* solutions.

Regarding the centroids, Section 5.5 already shows that they consist of an initial population of particularly high average fitness, with little variance. This could be explained by the centroids being potential local maxima of high fitness and very diverse nature, since coming from distinct clusters.

6 Conclusion

In this study, we seek to gain insights about a search space of image classification models in order to improve the performance of NAS algorithms. More precisely, we want to know if the convergence of a search strategy could be improved using a data-driven initialization technique exploiting the search space.

For this purpose, we propose a novel approach to improve the performances of a NAS search strategy. First, we perform a clustering analysis of the search space, involving a sequence of sub-tasks. It summarizes as follows: we sample models from a search space, reduce their dimension, perform a clustering. After a careful tuning of the clustering pipeline (number of dimensions, clusters, etc.), we select the algorithm providing the best qualitative and quantitative results. Second, we extract and use the centroids as an initial population to a search strategy.

We validate our proposal by initializing three (3) evolutionary algorithms, namely a genetic algorithm (GA), an evolutionary algorithm (EA), and Aging Evolution (AE), and benchmark our data-driven initialization method against conventional initialization baselines, i.e., random initialization and Latin Hypercube Sampling (LHS). To test the algorithms, we query the dataset of NAS-Bench-101, providing with a search space of image classifiers and their fitness evaluation on CIFAR-10. Our results show that centroids extracted using BGM for clustering are a promising approach to initialize a population-based algorithm. In the scenario of selecting models trained only 36 epochs, this approach used with GA shows significant long-term improvements (after 2000 iterations, in test) over random initialization and LHS, when using a Short encoding. When used with EA, it shows faster convergence (in validation) and significant long-term improvements over random initialization and LHS, when using a Short encoding and for all training budgets. Additional investigations on the distributions of the solutions found by the algorithms suggest that centroids enable retrieving local optima (maxima) of high fitness and similar configurations. Besides, we also investigate how an initial population gathered with our proposal on a tabular benchmark can be used for augmenting search on a real world problem, the So2Sat LCZ-42 scene classification dataset. Our early results show that this set of solutions help accelerate the convergence of EA on the target dataset, despite a short training budget.

Moreover, the cost of the approach lies in the size of the sample to collect (10k individuals), to serve as input to the initialization technique. More precisely, the computationally costly steps are the training and evaluation of the sample, while the sampling and clustering only require a few minutes to run. We argue that this drawback (computational cost) can be alleviated by using tabular or surrogate NAS benchmarks for obtaining free fitness evaluations of the sample. We demonstrate that once collected, the sample help initialiaze search for other applications. In order words, this cost can either be avoided (use of available benchmarks) and or limited (transfer to other applications).

As future work, we propose to investigate performances of this approach when selecting models on the Long Encoding. We also propose to study in depth the obtained clusters to gain more insights on obtained performances , and to explore different sampling strategies to select the models for the clustering. One might also explore the benefits of such data-driven initialization method on other families of algorithms (Bayesian optimization, local search, etc.).

Acknowledgements Authors acknowledge support by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. [ERC-2016-StG-714087], Acronym: *So2Sat*), by the Helmholtz Association through the Framework of Helmholtz AI [grant number: ZT-I-PF-5-01] - Local Unit "Munich Unit @Aeronautics, Space and Transport (MASTr)" and Helmholtz Excellent Professorship "Data Science in Earth Observation - Big Data Fusion for Urban Research"(W2-W3-100), by the German Federal Ministry of Education and Research (BMBF) in the framework of the international future AI lab "AI4EO – Artificial Intelligence for Earth Observation: Reasoning, Uncertainties, Ethics and Beyond" (Grant number: 01DD20001), the grant DeToL, and a DAAD Research fellowship.

Funding Open Access funding enabled and organized by Projekt DEAL. The funding is stated in Acknowledgment section.

Data Availability https://github.com/kalifou/data-driven-initialization-to-search.

Code Availability https://github.com/kalifou/data-driven-initialization-to-search.

Declarations

Consent for Publication All authors have checked the manuscript and have agreed to the submission.

Conflict of Interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Haykin, S.: Neural Networks and Learning Machines, vol. 3. Pearson Upper Saddle River, Hoboken (2009)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
3. Elsken, T., Metzen, J.H., Hutter, F., et al.: Neural architecture search: A survey. J. Mach. Learn. Res. **20**(55), 1–21 (2019)
4. Ojha, V.K., Abraham, A., Snášel, V.: Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng. Appl. Artif. Intel. **60**, 97–116 (2017)
5. Hanxiao, L., Karen, S., Yiming, Y.: Darts: Differentiable architecture search. International Conference on Learning Representations (2019)
6. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01), 4780–4789 (2019). https://doi.org/10.1609/aaai.v33i01.33014780
7. Camero, A., Wang, H., Alba, E., Bäck, T.: Bayesian neural architecture search using a training-free performance metric. Applied Soft Computing 107356 (2021)
8. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the 24th International Conference on Artificial Intelligence. IJCAI'15, pp. 3460–3468. AAAI Press (2015)
9. Camero, A., Toutouh, J., Alba, E.: Low-cost recurrent neural network expected performance evaluation. arXiv:1805.07159 (2018)
10. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: NAS-bench-101: Towards reproducible neural architecture search. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7105–7114. PMLR, Long Beach (2019). http://proceedings.mlr.press/v97/ying19a.html
11. Dong, X., Yang, Y.: Nas-Bench-201: Extending the scope of reproducible neural architecture search. In: International Conference on Learning Representations (ICLR) (2020)

12. Back, T.: Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Oxford (1996)
13. Holland, J.H.: Outline for a logical theory of adaptive systems. J. ACM (JACM) **9**(3), 297–314 (1962)
14. Hutter, F., Kotthoff, L., Vanschoren, J.: Automated Machine Learning - Methods, Systems, Challenges. Springer, Berlin (2019)
15. Engel, J.: Teaching Feed-Forward neural networks by simulated annealing. Complex Systems **2**, 641–648 (1988)
16. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. Proceedings of the 11th International Joint Conference on Artificial intelligence **1**(89), 762–767 (1989)
17. Alba, E., Aldana, J., Troya, J.: Genetic algorithms as heuristics for optimizing ANN design. In: Artificial Neural Nets and Genetic Algorithms, pp. 683–690. Springer, Berlin (1993)
18. Alba, E., Aldana, J., Troya, J.M.: Full automatic ann design: A genetic approach. In: International Workshop on Artificial Neural Networks, pp. 399–404. Springer (1993)
19. Yao, X.: A review of evolutionary artificial neural networks. Int. J. Intell. Syst. **8**(4), 539–567 (1993)
20. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
21. Camero, A., Toutouh, J., Alba, E.: Random error sampling-based recurrent neural network architecture optimization. Eng. Appl. Artif. Intel. **103946**, 96 (2020)
22. Zhining, Y., Yunming, P.: The genetic convolutional neural network model based on random sample. Int. J. u-and e-Service Sci. Technol. **8**(11), 317–326 (2015)
23. Rosa, G., Papa, J., Marana, A., Scheirer, W., Cox, D.: Fine-Tuning Convolutional Neural Networks Using Harmony Search. In: Pardo, A., Kittler, J. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, pp. 683–690. Springer, Cham (2015)
24. Van Stein, B., Wang, H., Bäck, T.: Automatic configuration of deep neural networks with parallel efficient global optimization. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2019)
25. Ororbia, A., ElSaid, A., Desell, T.: Investigating recurrent neural network memory structures using neuro-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 446–455. ACM (2019)
26. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al.: Evolving deep neural networks. In: Artificial Intelligence in the Age of Neural Networks and Brain Computing, pp. 293–312. Elsevier, Netherlands (2019)
27. Wang, C., Xu, C., Yao, X., Tao, D.: Evolutionary generative adversarial networks. IEEE Trans. Evol. Comput. **23**(6), 921–934 (2019)
28. Yang, A., Esperança, P.M., Carlucci, F.M.: Nas evaluation is frustratingly hard. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=HygrdpVKvr. Accessed 01 April 2022
29. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: Nas-bench-101: Towards reproducible neural architecture search. In: International Conference on Machine Learning, pp. 7105–7114. PMLR (2019)
30. Zela, A., Siems, J., Frank, H.: Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. International Conference on Learning Representations (2020)
31. Siems, J., Zimmer, L., Zela, A., Lukasik, J., Keuper, M., Hutter, F.: Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. arXiv:2008.09777 (2020)
32. Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., Burnaev, E.: NAS-Bench-NLP: Neural architecture search benchmark for natural language processing. arXiv:2006.07116 (2020)
33. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning, pp. 4095–4104. PMLR (2018)
34. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks. arXiv:1708.05344 (2017)
35. Camero, A., Toutouh, J., Alba, E.: Comparing deep recurrent networks based on the mae random sampling, a first approach. In: Conf of the Spanish Association for Artificial Intelligence, pp. 24–33. Springer (2018)
36. Lin, M., Wang, P., Sun, Z., Chen, H., Sun, X., Qian, Q., Li, H., Jin, R.: Zen-nas: a zero-shot nas for high-performance image recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 347–356 (2021)
37. Dürr, P., Mattiussi, C., Floreano, D.: Neuroevolution with Analog Genetic Encoding. In: Parallel Problem Solving from Nature-PPSN Ix, Pp, pp. 671–680. Springer, Berlin (2006)

38. Ning, X., Zheng, Y., Zhao, T., Wang, Y., Yang, H.: A generic graph-based neural architecture encoding scheme for predictor-based nas. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16, pp. 189–204. Springer (2020)
39. Chu, X., Zhang, B., Ma, H., Xu, R., Li, Q.: Fast, accurate and lightweight super-resolution with neural architecture search. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 59–64. IEEE (2021)
40. Nunes, M., Fraga, P.M., Pappa, G.L.: Fitness landscape analysis of graph neural network architecture search spaces. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '21, pp. 876–884. Association for Computing Machinery (2021). https://doi.org/10.1145/3449639.3459318
41. Traoré, K.R., Camero, A., Zhu, X.X.: Fitness Landscape Footprint: A Framework to Compare Neural Architecture Search Problems (2021)
42. Zhang, T., Lei, C., Zhang, Z., Meng, X.-B., Chen, C.P.: As-nas: Adaptive scalable neural architecture search with reinforced evolutionary algorithm for deep learning. IEEE Transactions on Evolutionary Computation (2021)
43. Maaranen, H., Miettinen, K., Mäkelä, M.M.: Quasi-random initial population for genetic algorithms. Comput. Math.. Appl. **47**(12), 1885–1895 (2004)
44. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Quasi-oppositional differential evolution. In: 2007 IEEE Congress on Evolutionary Computation, pp. 2229–2236. IEEE (2007)
45. Clerc, M.: Initialisations for particle swarm optimisation Online at http://clerc.maurice.free.fr/pso. Accessed 01 April 2022 (2008)
46. Helwig, S., Wanka, R.: Theoretical analysis of initial particle swarm behavior. In: International Conference on Parallel Problem Solving from Nature, pp. 889–898. Springer (2008)
47. Kazimipour, B., Li, X., Qin, K.: A review of population initialization techniques for evolutionary algorithms. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014 (2014). https://doi.org/10.1109/CEC.2014.6900618
48. Kazimipour, B., Li, X., Qin, A.K.: Initialization methods for large scale global optimization. In: 2013 IEEE Congress on Evolutionary Computation, pp. 2750–2757. IEEE (2013)
49. Kimura, S., Matsumura, K.: Genetic algorithms using low-discrepancy sequences. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 1341–1346 (2005)
50. Morrison, R.W.: Dispersion-based population initialization. In: Genetic and Evolutionary Computation Conference, pp. 1210–1221. Springer (2003)
51. Ma, Z., Vandenbosch, G.A.: Impact of random number generators on the performance of particle swarm optimization in antenna design. In: 2012 6th European Conference on Antennas and Propagation (EUCAP), pp. 925–929. IEEE (2012)
52. Poles, S., Fu, Y., Rigoni, E.: The effect of initial population sampling on the convergence of multi-objective genetic algorithms. In: Multiobjective Programming and Goal Programming, pp. 123–133. Springer, Berlin (2009)
53. Mousavirad, S.J., Bidgoli, A.A., Rahnamayan, S.: Tackling deceptive optimization problems using opposition-based de with center-based latin hypercube initialization. In: 2019 14th International Conference on Computer Science & Education (ICCSE), pp. 394–400. IEEE (2019)
54. Medeiros, H.R., Izidio, D.M., Ferreira, A.P.D.A., Da, S., Barros, E.N.: Latin hypercube initialization strategy for design space exploration of deep neural network architectures. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 295–296 (2019)
55. Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., Gagné, C.: DEAP: Evolutionary Algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)
56. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
57. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987). https://doi.org/10.1016/0377-0427(87)90125-7
58. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. **3**(1), 1–27 (1974). https://doi.org/10.1080/03610927408827101
59. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1(2) 224–227 (1979)
60. Zhu, X.X., Hu, J., Qiu, C., Shi, Y., Kang, J., Mou, L., Bagheri, H., Haberle, M., Hua, Y., Huang, R., et al.: So2sat lcz42: a benchmark data set for the classification of global local climate zones [software and data sets]. IEEE Geosci. Remote Sens. **8**(3), 76–89 (2020)

This page is intentionally left blank.

We Won't Get Fooled Again: When Performance Metric Malfunction Affects the Landscape of Hyperparameter Optimization Problems

Kalifou René Traoré[1,2]( ), Andrés Camero[2,3], and Xiao Xiang Zhu[1]

[1] Data Science in Earth Observation, Technical University of Munich, Arcisstrasse 21, Munich, Germany
xiaoxiang.zhu@tum.de

[2] Remote Sensing Institute, German Aerospace Center (DLR), Münchener Strasse 20, Weßling, Germany
{kalifou.traore,andres.camerounzueta}@dlr.de

[3] Helmholtz AI, Neuherberg, Germany

Abstract. Hyperparameter optimization (HPO) is a well-studied research field. However, the effects and interactions of the components in an HPO pipeline are not yet well investigated. Then, we ask ourselves: *Can the landscape of HPO be biased by the pipeline used to evaluate individual configurations?* To address this question, we proposed to analyze the effect of the HPO pipeline on HPO problems using fitness landscape analysis. Particularly, we studied over 119 generic classification instances from either the DS-2019 (CNN) and YAHPO (XGBoost) HPO benchmark data sets, looking for patterns that could indicate evaluation pipeline malfunction, and relate them to HPO performance. Our main findings are: (i) In most instances, large groups of diverse hyperparameters (i.e., multiple configurations) yield the same *ill* performance, most likely associated with majority class prediction models (predictive accuracy) or models unable to attribute an appropriate class to observations (log loss); (ii) in these cases, a worsened correlation between the observed fitness and average fitness in the neighborhood is observed, potentially making harder the deployment of local-search-based HPO strategies. (iii) these effects are observed across different HPO scenarios (tuning CNN or XGBoost algorithms). Finally, we concluded that the HPO pipeline definition might negatively affect the HPO landscape.

Keywords: Hyperparameter Optimization · Fitness Landscape Analysis · Benchmarking

1 Introduction and Related Work

Modern data-driven approaches dealing with large-scale data require domain, data science, and technical expertise. The variety of application tasks (e.g., classification and object detection) often require designing models that are not necessarily reusable in other tasks, and this process is both resource-demanding and error-prone [3,8,12]. Thus, automating the design of ML pipelines, a.k.a. *AutoML* [6], is much more desirable.

AutoML is usually split into four main activities: data preparation, feature engineering, model generation, and model estimation [5]. *Hyperparameter optimization* (HPO [1]) is an important task in model generation. HPO aims at automatically tuning the hyperparameters of learning algorithms, and as with all optimization problems, it is facing the process of minimizing/maximizing a target function (e.g., the performance metric of the model) subject to a set of constraints. HPO is a well-studied field [1], but the effects and interaction between the components of its pipeline are not yet well investigated. Recently, authors [10] have proposed to characterize the search space of AutoML pipelines using *fitness landscape analysis* (FLA [11]). In the same line, [15] proposed a FLA-base framework to characterize NAS problems, and applied it to a multi-sensor data fusion problem [14]. Despite the great results and insights provided by these studies, the relation between HPO and the rest of the HPO pipeline remains barely explored.

Therefore, in this study, we pose the following research question: ***Can the landscape of HPO be biased by the pipeline used to evaluate individual configurations?*** To address this question, we propose to study HPO in the context of AutoML using FLA. Particularly, using *fitness distance correlation* (FDC [7]), *locality* and *neutrality* [2], we aim at patterns that arise from evaluation pipeline issues and assess how they could alter the landscapes of HPO problems. The results on over 119 instances from either the DS-2019 [13] or the YAHPO [9] HPO benchmarks show the existence of large groups of diverse HP configurations that yield the same *ill* fitness value. This *illness* could be explained by the fitness metric selection (e.g., predictive accuracy and log loss), that *induce* various *suboptimal* model behavior, in scenarios of *different natures* (tuning CNN and XGBoost algorithms). More precisely, for the predictive accuracy, we suspect the generation of majority class predictors. In the case of the log loss criterion, the generation of models unable to classify. A complementary analysis of locality shows that the resulting landscapes are more rugged, with lesser correlation between the observed fitness and the fitness in the neighborhood. In other words, these problems are hard to tackle using a local-search strategy.

The rest of the paper is as follows: The next section introduces the methodology used in the study, Sect. 3 presents results of landscape analysis on HPO problems, and Sect. 4 provides conclusions.

2 Methodology

Given a HPO problem, let S be the HP configuration space, f the fitness function that assigns a value $f(x) \in \mathbb{R}$ to all configurations $x \in S$, and $N(x)$ a neighborhood operator that provides a structure to S. Then, the fitness landscape is defined as $\mathcal{L} = (S, f, N)$.

We are interested in exploiting the landscape definition to study the relation between the HPO landscape and the HPO pipeline, and check whether the pipeline may bias the HPO landscape. Particularly, we propose to use the FDC and *locality* to characterize this relation. The motivation is that issues related to the evaluation pipeline should affect the fitness of configurations irrespectively of their configuration, and thus their distance to the optimum. In other words, repetitive or grouping patterns (such as lines) might appear when visualizing distributions of distances to the optimum. Moreover, the locality of the configuration space should be arbitrarily affected, i.e., some configurations should present an unexpected or *random* behavior (in relation to the neighborhood).

Without loss of generality, we consider the problem of tuning the HPs of a fixed model, e.g. neural network architecture or XGBoost ensemble, to perform a task (e.g., classification). Typically, the HP configuration space consists of mixed type features (continuous, discrete or categorical). Thus, we propose to evaluate the distance between individuals using a dedicated similarity function, $\delta(x, y)$, introduced by [4]. Then, we define a neighborhood function $N(x) = \{y \in S \mid \delta(x, y) < \Delta\}$.

The FDC is often interpreted as a measure of the existence of search trajectories from randomly picked configurations to the known global optimum. In practice, the FDC is not collected as a correlation score but visualized as the distribution of fitness versus distance to the global optimum. It writes as: $\mathrm{FDC}(f, x^*, S) = \{(\delta(x^*, y), f(y)) \mid \forall y \in S\}$, where $x^* \in S$ is the global optimum. On the other hand, *locality* corresponds to the relationship between the observed fitness and the distribution of average fitness in the neighborhood [2].

Moreover, the *neutrality degree* [2] provides an additional picture of the interaction between solutions in the landscape. It is defined as $N_d(x) = |\{x^{'} \in N(x) \mid | f(x^{'}) - f(x) | < \epsilon\}|$, and is interpreted as the number of neighbors of x that have a *similar* fitness. In this case, we set $\epsilon = \mathtt{max}_{\mathrm{fitness}}/C$. Besides, Table 1 and 2 give a description of abbreviations and symbols used in the paper.

Table 1. Table of abbreviations used in the paper.

Abbreviation	Description
HP	Hyperparameter
HPO	Hyperparameter Optimization
CV	Computer Vision
FLA	Fitness Landscape Analysis
FDC	Fitness Distance Correlation
CNN	Convolutional Neural Network
MMCE	Mean Misclassification Error

3 Results

To evaluate the proposed methodology, we propose to analyze the **DS-2019** and **YAHPO** HPO benchmark data sets. DS-2019 consists of a tabular benchmark for the scenario of tuning the HPs of a (fixed) convolutional neural network

(CNN), a ResNet-18, on ten instances of CV classification. For each instance, 15 hyperparameters should be optimized, including the batch size, number of epochs, and momentum, among others.

Table 2. Table of symbols used in the paper.

Symbol	Description
S	Hyperparameter Configuration Space
N	Neighborhood operator
f	Fitness evaluation function
$\mathcal{L}$	Fitness landscape derived from the combination of S, N, f
$\mathtt{max}_{\text{fitness}}$	Maximum fitness value observed in S
$\mathtt{max}_{\text{dist}}$	Maximum pairwise distance (to the optimum) measured in S
$N_d(x)$	Neutrality degree of a HP solution x
$\delta(x, y)$	Gower distance between HP solutions x and y
Δ	Threshold (in Gower distance) used by N to assign neighbors
ϵ	Threshold (in fitness) used by N_d to assign neutral neighbors
C	Constant used to define the fitness neutrality threshold ϵ

YAHPO consists of a tabular benchmark for the scenarios of tuning various learning algorithms (e.g. XGBoost, Neural Networks) for 119 instances of classification, in various domains of applications. Many of the instances were obtained from the collaborative open-source OpenML platform, gathering an ever-growing number of machine learning instances. In this study, we focus on the specific scenario tuning a set of 15 HPs for the XGBoost learning algorithm.

The code used for the experiments is available following the anonymized link: https://github.com/anonymous-for-open-review/late-breaking-automlConf-2022.

3.1 Classification Accuracy

The following paragraphs introduce results when evaluating solutions using the metric of *predictive accuracy* (DS-2019). In the case of YAHPO, the available metric is the *mean misclassification error* ($MMCE = 1 - predictive\ accuracy$), and we focus on the instances *WDBC, YEAST, MINIBOONE, and ISOLET*. Similar observations are made for the rest of the 119 instances.

Fitness Distance Correlation (FDC)
First, for each instance, we randomly sampled 1000 HP configurations and computed the FDC. Results are shown in Figs. 1 and 2, respectively, for DS-2019 and YAHPO.

Overall, the distances to the global optimum cover a wide range of values: the distribution of distances is wide and uniform in most instances. This is the case for both benchmarks. This suggests a large diversity in the HP configurations (with respect to the optimum), for the sample and potentially the whole configuration space. This is true for both benchmarks, with slightly more narrow distributions of distances for YAHPO. This suggests slightly less diversity within the XGBoost HP configuration space (YAHPO), than one of the CNN classifiers (DS-2019).

In most instances of DS-2019, the fitness also covers a wide range of values, as opposed to relatively more narrow distributions of fitness on YAHPO. This suggests a larger influence on HP configuration on the fitness of CNN classifiers (DS-2019) than on more classical and notoriously robust ensembles of models, i.e XGBoost (YAHPO). Overall, the distributions of fitness all seem to be multimodal, with a principal mode for large fitness values (i.e., good configurations), and another mode for *odd* values, for both benchmarks.

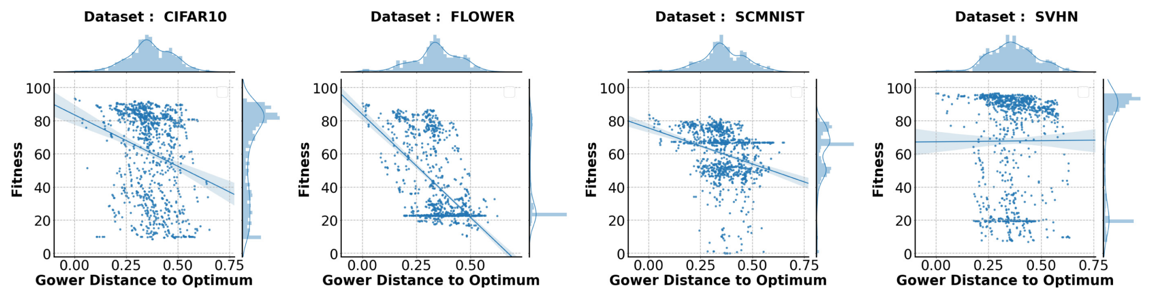


Fig. 1. FDC plot for instances from the DS2019 benchmark, and the corresponding regression line in blue. The fitness function is the *predictive accuracy*. (Color figure online)

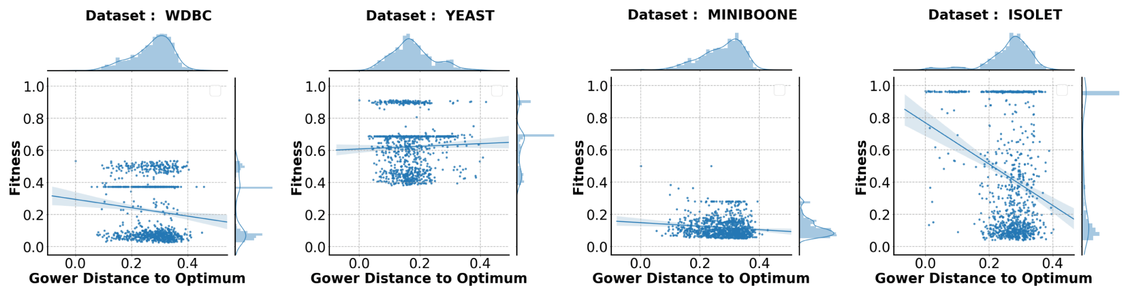


Fig. 2. FDC plot for a few instances from the YAHPO HPO benchmark, and the corresponding regression line in blue. The fitness function is the MMCE. (Color figure online)

Besides, we checked the data distribution for each instance, and we notice that the *odd* modes could be correlated to the majority class. Note that the fitness metric used is predictive accuracy for DS-2019, and its opposite the MMCE for YAHPO. For example in DS-2019, on FLOWER it is around 25%, SCMNIST around 65% and SVHN around 20%. In YAHPO, on WDBC it is around 38%, on YEAST around 70%, and ISOLET around 97%, among others. In particular, configurations are affected regardless of the distance to the optimum. In other words, very diverse configurations yield the same fitness value. This phenomenon could be attached to issues with the learning process, failing to properly fit the data and being stuck in poor local optima (i.e., majority class prediction), preventing them to reach the fitness that their HP configuration would normally yield. Besides, there is no clear global correlation between the observed fitness and distance to the global optimum. This could be caused by the multi-modal nature of the distributions of fitness.

Neighborhood

Next, we seek to identify how the observed artifacts, i.e., the *majority* class predictors, affect the locality of landscapes. Figures 3 and 4 show the distribution of average neighbor fitness as a function of the observed fitness, respectively, for instances from DS-2019 and YAHPO. The black dash-dotted line represents the bisector, i.e., the line connecting all points of equal value on both axis. To generate the plots, we used the previously sampled configurations, and identified the maximal pairwise distance (of any individual) to the optimum $\texttt{max}_{\text{dist}}$, and maximum observed fitness $\texttt{max}_{\text{fitness}}$. Given a constant $C = 40$, we discretize the range of fitness values into intervals, where a step is equal to the maximum observed fitness $\texttt{max}_{\text{fitness}}$ divided by C. In order to decide if a configuration is a neighbor, we set $\varDelta = \texttt{max}_{\text{dist}}/C$.

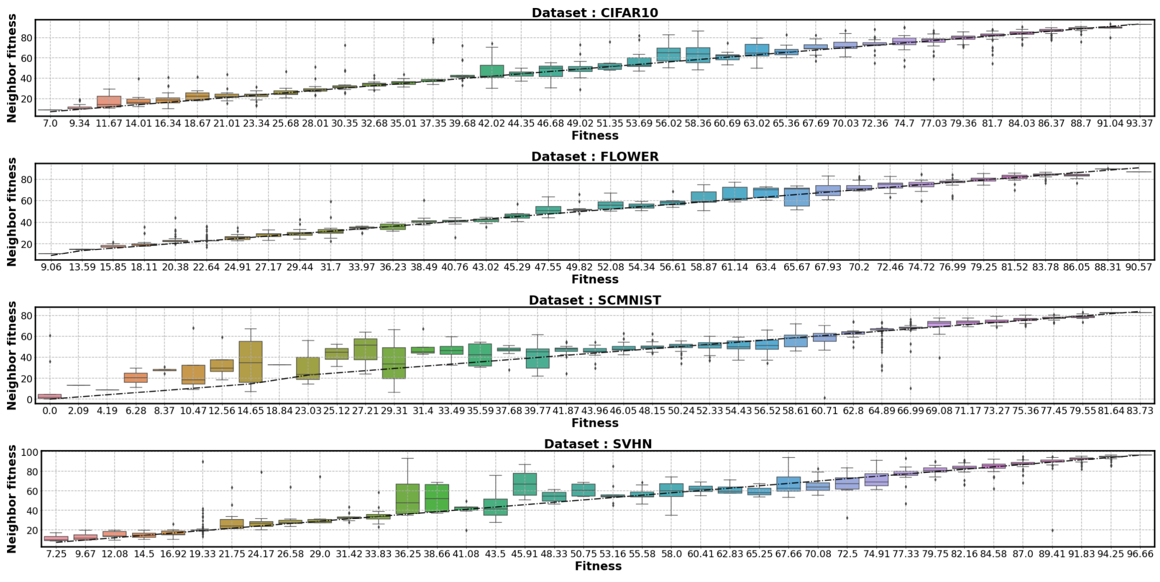


Fig. 3. Distribution of the average fitness of neighbors as a function of the observed fitness (*predictive accuracy*), for a few instances of the DS2019 benchmark.

Overall, we observe in many instances of DS-2019 a strong correlation between the observed fitness and the average fitness in the neighborhood. Indeed, the box-plots are aligned with the bisector. From the perspective of local search, it is easy to navigate the configuration space by consistently improving the fitness, from randomly distant and bad configurations to configurations of high fitness, for instances from DS-2019. This is less the case in YAHPO, as show in Fig. 4. Indeed, we observe a weaker correlation between the two variables. In the presence of unexpected mode in the distributions of fitness (see Fig. 2), e.g. YEAST and ISOLET, we find that a majority of neighbors tend to have the fitness of the observed mode, respectively around 70% and 97%. This suggests that the respective landscapes might have many local optima surrounded by plane areas at the odd fitness value. Thus, the chances for local search of being stuck are higher in such instances. Besides, the instances with more uniform and wider distribution of fitness (Fig. 1) tend to have a near perfect correlation. On the other hand, the more the distributions are multi-modal and with peaky modes, the worse the correlation between the variables of interest. This is the case for both benchmarks.

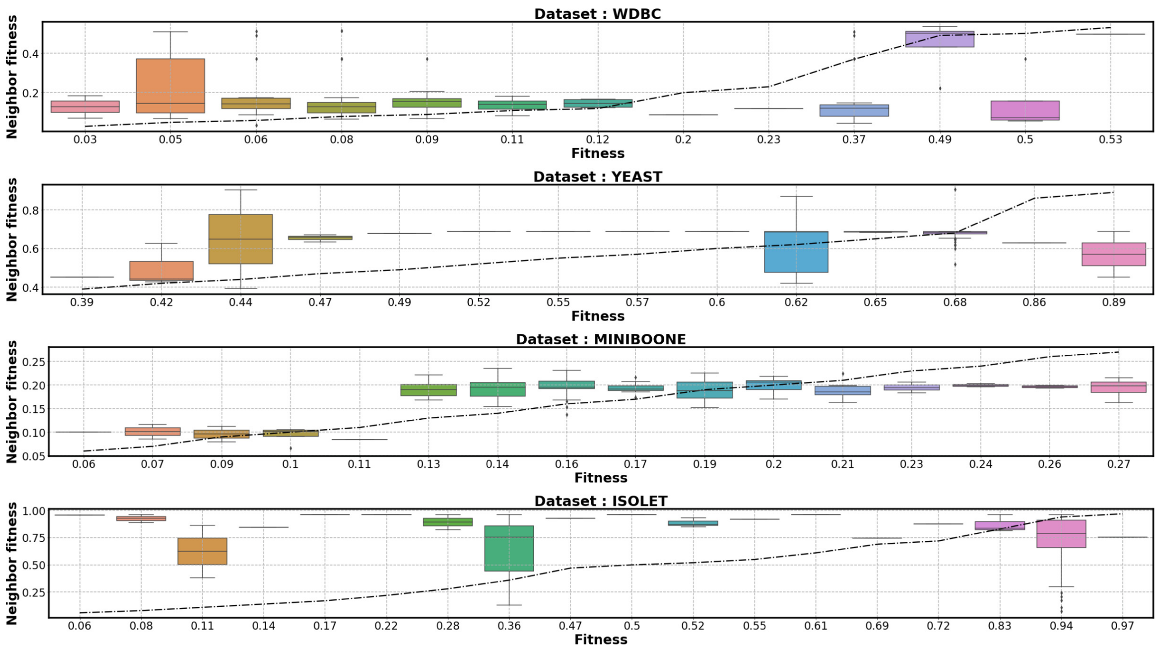


Fig. 4. Distribution of the average fitness of neighbors as a function of the observed fitness (MMCE), for a few instances from the YAHPO benchmark.

To summarize, results indicate that the evaluation protocol could have an impact on the easiness and practicability of HPO landscapes, assessed by the correlation.

Neutrality

Next, we look into the neutrality of the landscape. Figures 5 and 6 show the distribution of neutral neighbor counts as a function of the current fitness, respectively, for instances from DS-2019 and YAHPO.

For instances from DS-2019, the neutrality degree is equal to or greater than one, for most ranges of fitness values. In order words, most configurations have at least one neutral neighbor. Also, note that the FDC and *locality* results for CIFAR-10 are *good*, while for SCMNIST adn SVHN, with a multi-modal distribution of fitness (Fig. 1), coupled with lower *local* correlation (i.e., between the fitness and the fitness in the neighborhood, Fig. 3), the results are *bad*. Regarding CIFAR-10, the neutrality degree is on average consistently greater than two. In other words, most configurations have two or (many) more *neutral* neighbors. On the other hand, for SCMNIST and SVHN, the neutrality degree is inconsistent and with lower values on average. In particular, N_d is lower for fitness values ranging from 6.28 to 43.98% for SCMNIST, i.e., generally bad configurations have fewer neutral neighbors than *mid* and *good* configurations. Also, as expected, there is a huge number of neutral neighbors around the majority class prediction fitness: around 65% for SCMNIST and 20% for SVHN.

In YAHPO, we find that the range of fitness for which one can find solutions with neutral neighbors is limited. In practice, it represents a fraction of the range covered by all evaluated solutions. Besides, the neutrality degree is generally above 2, with larger counts associated to the modes in the distributions of fitness (See Fig. 2). These two facts suggest that the landscapes might be highly rugged with many local optima (no neutral neighbors), with areas centered towards values of the observed modes.

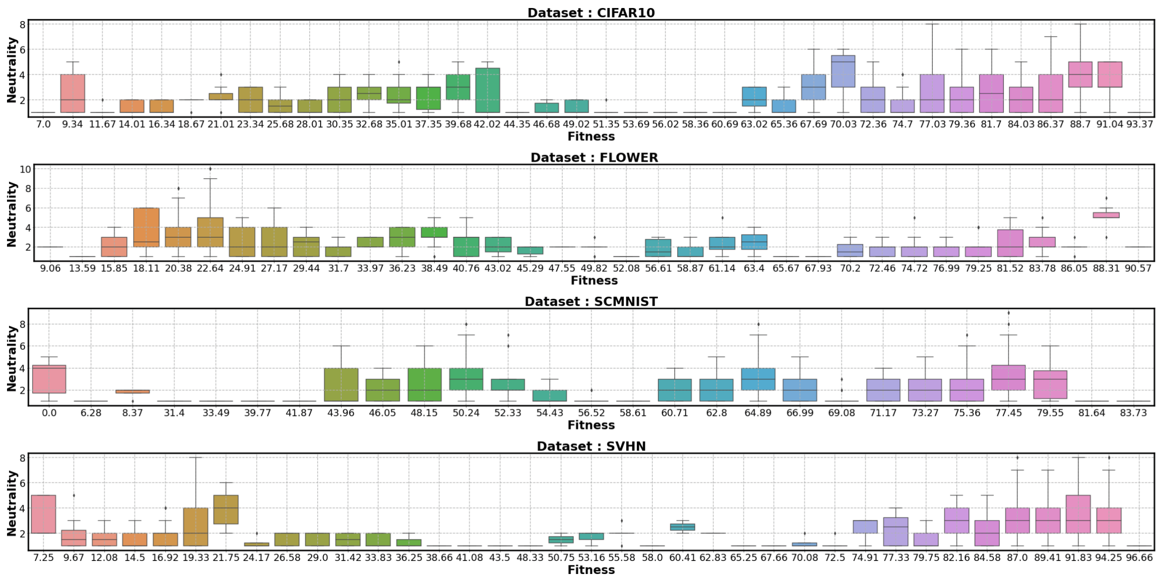


Fig. 5. Neutrality degree as a function of the observed fitness (*predictive accuracy*), for a few instances from the DS2019 benchmark.

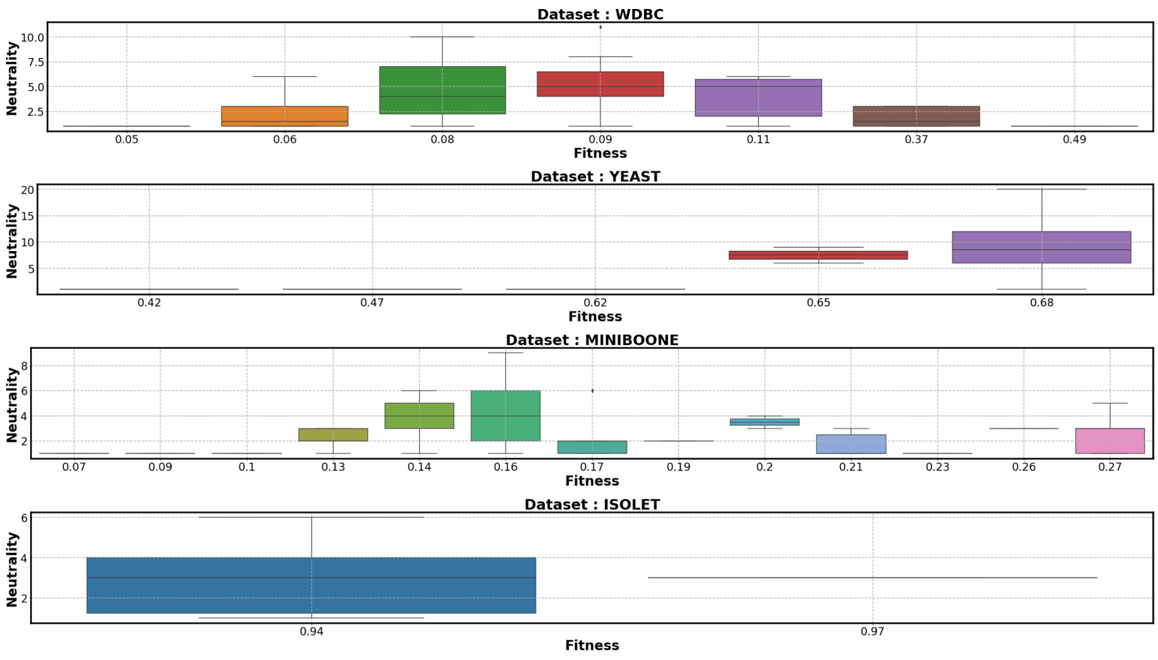


Fig. 6. Neutrality degree as a function of the observed fitness (MMCE), for a few instances from the YAHPO benchmark.

As a summary, the evaluation pipeline *malfunction* is responsible for an *imbalanced* landscape, i.e., the AutoML pipeline generates arbitrary *peaks* of fitness (low N_d) in areas of expected continuous fitness.

3.2 Log Loss

The following paragraphs present the results of the analysis when evaluating solutions using the Log Loss for classification. This is done on a sub-sample of the YAHPO benchmark, namely *SEMEION, VEHICLE, SEGMENT, KC1*. Similar observations are made for the remaining 115 instances.

Fitness Distance Correlation (FDC)

First, we look at the FDC for the four instances *SEMEION, VEHICLE, SEGMENT, KC1*, as shown in Fig. 7.

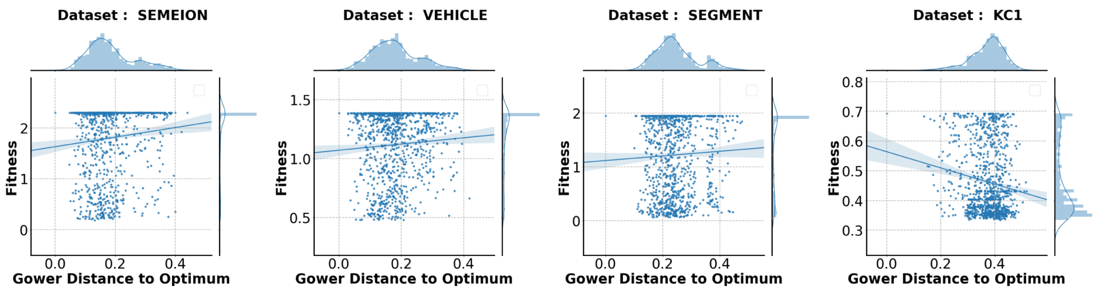


Fig. 7. FDC plot for a few instances from the YAHPO HPO benchmark, and the corresponding regression line in blue. The fitness function is the Log Loss. (Color figure online)

Similarly to results gathered in Sect. 3.1 (*predictive classification accuracy*), we find that the distributions of fitness are also covering a wide range of values, and are multi-modal. Several distributions also have the artifact identified previously: an unexpected mode around large (*poor*) fitness values, associated with HP configurations of highly variable *Gower* dissimilarity to the optimum. For instance, it is around the Log Loss value of 2.25 for SEMEION, 1.55 for VEHICLE, and 1.98 for SEGMENT. It affects HP configurations at Gower distances 0.15 to 0.45, i.e. covering the whole range of dissimilarity to the optimal HP. This phenomenon is also observed for a majority of the 119 analyzed instances of YAHPO. Besides, another mode can exist (around 0.35 for KC1) at low fitness values, i.e *good* HP configurations.

When looking at the nature of the instances (number of classes), we find that the unexpected mode correlates with the fitness value yielded by the Log Loss metric when attributing an *equal* probability of occurring to all classes, for all observations. In other words, solutions associated with the mode are likely solutions with *no classification ability*. This phenomenon could occur since the Log Loss metric does not penalize such behavior, a local optimum to which many solutions could naturally converge to.

Neighborhood

Next, we look at the neighborhood in the landscapes generated by the Log Loss metric. This is shown in Fig. 8. Overall, we find that the correlation between the average fitness of neighbors and the observed fitness, is weak in the case of distributions of fitness with the identified artifact (*peaks*). Most neighbors have a fitness value of the unexpected mode, e.g. a Log Loss value of 2.25 for SEMEION

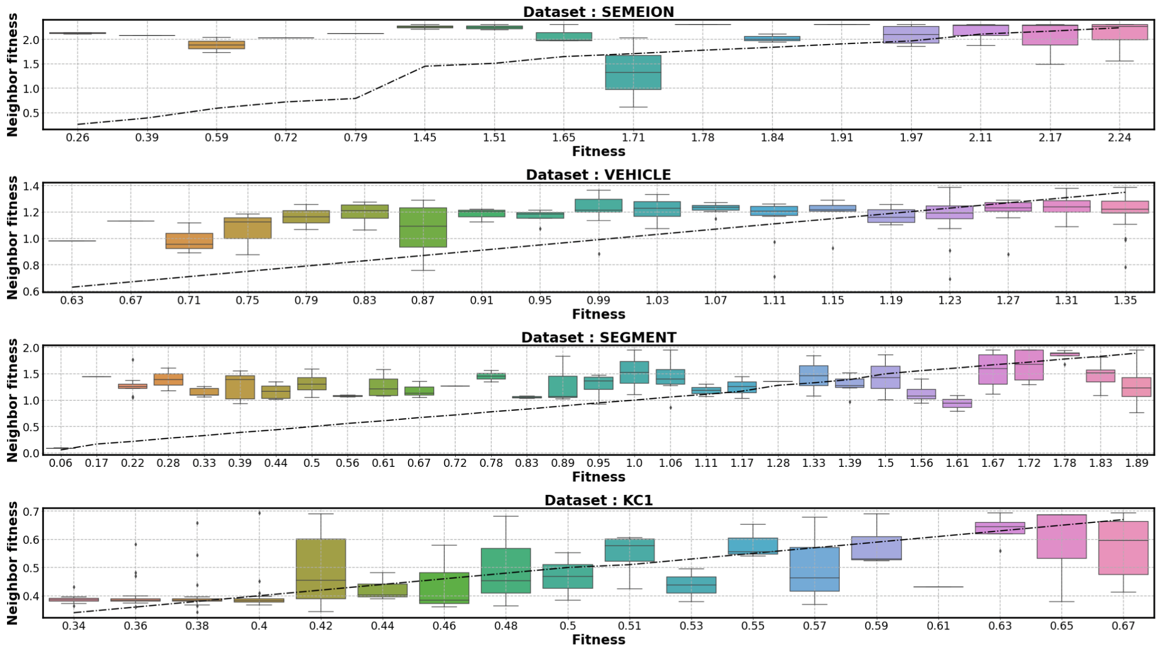


Fig. 8. Distribution of the average fitness of neighbors as a function of the observed fitness (Log Loss), for a few instances from the YAHPO benchmark.

and 1.35 for VEHICLE. This observation is in line with those made when using the metric of MMCE (see Figs. 1 and 2).

To summarize, using the Log Loss as an evaluation metric might also negatively impact the easiness of the associated landscapes, by increasing their ruggedness and decreasing their fitness potential.

Neutrality

Next, we look into the neutrality of the landscapes, as shown in Fig. 9. Similar to the analysis provided when evaluating with the MMCE, we find that few solutions have a neutral neighbor, and these are found within a restricted range of fitness. This suggests highly rugged landscapes. Besides, the highest counts of neutral neighbors are for solutions associated with the modes in the distributions of fitness (see Fig. 7). For instance, at a Log Loss value of 2.2 for SEMEION, and between 1.23 and 1.35 for VEHICLE. In other words, the landscapes are highly rugged (numerous local minima), and surrounded by a plateau of HP configurations associated with the high Log Loss values of the observed mode.

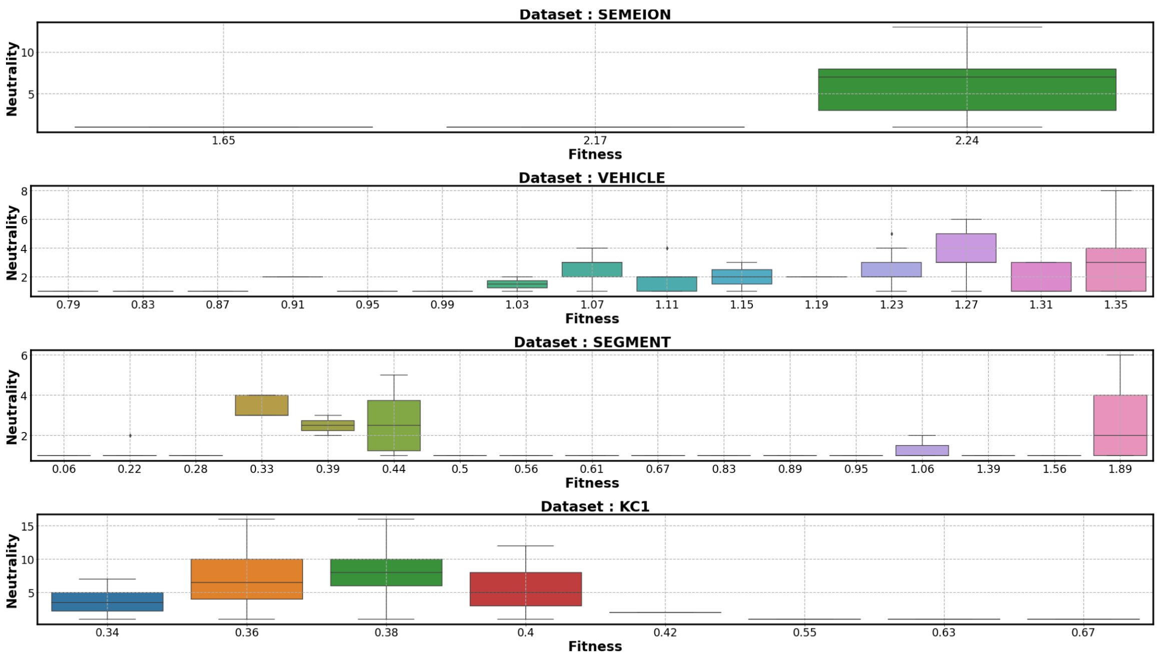


Fig. 9. Neutrality degree as a function of the observed fitness (Log Loss), for a few instances from the YAHPO benchmark.

4 Conclusions and Future Work

In this paper, we investigate if AutoML pipelines can negatively affect the landscape of HPO problems. More precisely, we address the following question: *Can the landscape of HPO be biased by the pipeline used to evaluate individual configurations?* To tackle this question, we have studied the fitness landscape of over 119 HPO instances obtained from either the DS-2019 (CNN) or YAHPO

(XGBoost) HPO benchmark data sets, using the concepts of *fitness distance correlation*, *locality*, and *neutrality*.

The FDC analysis shows unhealthy patterns in many HPO instances, with large groups of very diverse HP configurations with the same *ill* fitness value. These resulting peaks in fitness appear to be outliers in the respective distributions. Looking at the locality (fitness versus fitness in the neighborhood), we observe two things: First, there is a correlation between both variables of interest in healthy landscapes, suggesting that an *easy* path from randomly picked HP configurations could lead to the best performers, i.e., local-search may *do the job*. Second, for HPO problems negatively affected by the mentioned *illnesses* (i.e., the majority class predictors, or the inability to classify), the correlation between the current fitness and fitness in the neighborhood is worsened, indicating more rugged local landscapes.

Even though the *majority class prediction* problem for models trained and evaluated using some metrics (e.g., accuracy) is well known, the results show that the problem may not be taken seriously into account. This is also the case with the *inability to classify* arising when using the Log Loss as a training and evaluation criterion.

Thus, a great amount of resources is wasted when addressing HPO (i.e., many *simple* majority class or *inable* models are evaluated). Furthermore, the evidence shows that *the landscape of HPO problems could be negatively affected by the evaluation pipeline being used.*

Future work will further investigate how such artifacts affect HPO algorithms in practice.

Acknowledgements. Authors acknowledge support by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. [ERC-2016-StG-714087], Acronym: *So2Sat*), by the Helmholtz Association through the Framework of Helmholtz AI [grant number: ZT-I-PF-5-01] - Local Unit "Munich Unit @Aeronautics, Space and Transport (MASTr)" and Helmholtz Excellent Professorship "Data Science in Earth Observation - Big Data Fusion for Urban Research" (W2-W3-100), by the German Federal Ministry of Education and Research (BMBF) in the framework of the international future AI lab "AI4EO – Artificial Intelligence for Earth Observation: Reasoning, Uncertainties, Ethics and Beyond" (Grant number: 01DD20001) and the grant DeToL. The authors also acknowledge support by DAAD for a Doctoral Research Fellowship.

References

1. Bischl, B., et al.: Hyperparameter optimization: foundations, algorithms, best practices and open challenges (2021). https://doi.org/10.48550/ARXIV.2107.05847, https://arxiv.org/abs/2107.05847
2. Clergue, M., Verel, S., Formenti, E.: An iterated local search to find many solutions of the 6-states firing squad synchronization problem. Appl. Soft Comput. **66**, 449–461 (2018). https://doi.org/10.1016/j.asoc.2018.01.026, https://www.sciencedirect.com/science/article/pii/S1568494618300322

3. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: a survey. J. Mach. Learn. Res. **20**(1), 1997–2017 (2019)
4. Gower, J.C.: A general coefficient of similarity and some of its properties. Biometrics **27**(4), 857–871 (1971). http://www.jstor.org/stable/2528823
5. He, X., Zhao, K., Chu, X.: AutoML: a survey of the-state-of-the-art. Knowl.-Based Syst. **212**, 106622 (2021)
6. Hutter, F., Kotthoff, L., Vanschoren, J.: Automated Machine Learning - Methods, Systems, Challenges. Springer, Berlin (2019). https://doi.org/10.1007/978-3-030-05318-5
7. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann Publishers Inc., San Francisco (1995)
8. Ojha, V.K., Abraham, A., Snášel, V.: Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng. Appl. Arti. Intell. **60**, 97–116 (2017). https://doi.org/10.1016/j.engappai.2017.01.013, https://www.sciencedirect.com/science/article/pii/S0952197617300234
9. Pfisterer, F., Schneider, L., Moosbauer, J., Binder, M., Bischl, B.: YAHPO gym - an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization (2021)
10. Pimenta, C.G., de Sá, A.G.C., Ochoa, G., Pappa, G.L.: Fitness landscape analysis of automated machine learning search spaces. In: Paquete, L., Zarges, C. (eds.) EvoCOP 2020. LNCS, vol. 12102, pp. 114–130. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43680-3_8
11. Pitzer, E., Affenzeller, M.: A comprehensive survey on fitness landscape analysis. In: Fodor, J., Klempous, R., Suárez Araujo, C.P. (eds.) Recent Advances in Intelligent Engineering Systems. Studies in Computational Intelligence, vol. 378, pp. 161–191. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-23229-9_8
12. Ren, P., Xiao, Y., Chang, X., Huang, P.y., Li, Z., Chen, X., Wang, X.: A comprehensive survey of neural architecture search: challenges and solutions. ACM Comput. Surv. **54**(4) (2021). https://doi.org/10.1145/3447582
13. Sharma, A., van Rijn, J.N., Hutter, F., Müller, A.: Hyperparameter importance for image classification by residual neural networks. In: Kralj Novak, P., Šmuc, T., Džeroski, S. (eds.) DS 2019. LNCS (LNAI), vol. 11828, pp. 112–126. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33778-0_10
14. Traoré, K.R., Camero, A., Zhu, X.X.: Landscape of neural architecture search across sensors: how much do they differ ? ISPRS Ann. Photogr. Remote Sens. Spat. Inf. Sci. **V-3-2022**, 217–224 (2022). https://doi.org/10.5194/isprs-annals-V-3-2022-217-2022, https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-3-2022/217/2022/
15. Traoré, K.R., Camero, A., Zhu, X.X.: Fitness landscape footprint: a framework to compare neural architecture search problems (2021)

This page is intentionally left blank.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7. Available from http://www.i6doc.com/en/.

Compact Neural Architecture Search for Local Climate Zones Classification

Kalifou Rene Traore[1,2] Andrés Camero[2]
Xiao Xiang Zhu[1,2]

1- Technical University of Munich, Data Science in Earth Observation
2- German Aerospace Center (DLR), Remote Sensing Technology Institute (IMF)

Abstract. State-of-the-art Computer Vision models achieve impressive performance but with an increasing complexity. Great advances have been made towards automatic model design, but accounting for model performance and low complexity is still an open challenge. In this study, we propose a neural architecture search strategy for high performance low complexity classification models, that combines an efficient search algorithm with mechanisms for reducing complexity. We tested our proposal on a real World remote sensing problem, the Local Climate Zone classification. The results show that our proposal achieves state-of-the-art performance, while being at least 91.8% more compact in terms of size and FLOPs.

1 Introduction

Modern data-driven approaches to deal with large-scale data, particularly Deep Learning (DL), require domain and data science expertise. The variety of application tasks (e.g., classification and object detection) often require designing models that are not necessarily reusable in other tasks [1]. Moreover, manually designing models is time consuming and error prone. Thus, automating the Machine Learning (ML) pipeline design (a.k.a. AutoML [2]) is much desirable.

Neural Architecture Search (NAS) [3], a sub-field of AutoML focusing on the design of ML models, has already proven to be successful for a wide variety of problems, including Computer Vision (CV) [3].

In Earth observation (EO), satellite remote sensing enables recovering contact-free large-scale information about the physical properties of the Earth from space. Thanks to ESA's Sentinel missions and NewSpace companies, petabytes of satellite data has become available, leveraging large-scale datasets that have boosted the study of tailored models, in particular for multi-spectral and Synthetic Aperture Radar (SAR) data classification [4]. However, the application of NAS for further improvements remains quite unexplored.

In this paper, we propose a NAS strategy to automatically design high-performance low complexity image classification models. Particularly, we combine a *differentiable search strategy* to optimize the elementary normal cells of a backbone architecture, with a *structural depth* and a *complexity reducing loss.* We validate our proposal on a real world problem, the *So2Sat LCZ42* [5] data set for classification of Local Climate Zones (LCZ). Our results show that the best found models are on par with state-of-the-art baselines, while accounting for at least 91.8% less FLOPs and size.

2 Related Work

The increase in remote sensing missions and sensors has allowed more data collection, which in turn has motivated the creation of larger EO datasets for scientific purposes [6]. In order to analyse these large data sets, the remote sensing community has embraced the use of DL models classically used in CV for similar tasks [4]. However, due to the particularities of these EO data sets (e.g., multi-modal, geo-located, time-variable data), it is necessary to develop domain-specific models [7]. Therefore, AutoML (and particularly NAS), has the potential to ease and improve the quality of the future tailored models.

Great advances have been made in the NAS field [3, 8]. However, most approaches have not yet been adopted because they require a lot of computational resources and time [3, 8]. To cope with these limitations, some authors have leveraged the *differentiable architecture search* (DARTS) [9], dramatically reducing the search budget time (GDAS [10]), without compromising the performance. On the other hand, some authors have explore NAS for low complexity models, including the design of models for constrained hardware [11, 12], compression-based approaches [13], and pruning techniques [14], among others.

3 NAS for Compactness

A Differentiable Search Strategy: Let $\alpha = \{\alpha_{cell}, \Delta\}$ be a candidate neural architecture defined by an elementary module α_{cell}, a backbone configuration Δ, and ω_{cell} and Ω_{depth} the respective weights. The objective is to find α^* that minimizes a validation loss, while identifying its optimal parameters ω_{cell*} in training:

$$\min_{(\alpha_{cell},\Delta)} \mathbb{E}_{(x',y')\sim\mathbb{D}_V} L(x^{'}, y^{'}) \text{ s.t. } \omega^*_{cell} = \text{argmin}_{\omega} \mathbb{E}_{(x,y)\sim\mathbb{D}_T} L(x, y) \quad (1)$$

The optimized loss L is the sum of a prediction loss L_{NLL} and a weighted complexity reducing loss L_{cpl}: $L(x, y) = L_{NLL}(x, y) + C_{cpl} \cdot L_{cpl}$. Here L_{NLL} is the negative log likelihood of predicting the correct label y, given a data sample x, α_{cell}, Δ and $\omega_{cell} \in \mathbb{W}$. Both α_{cell}, Δ are respectively sampled from τ_{A} and τ_{Δ} the probability distributions over the cells and depths configurations. In practice, τ_{A} and τ_{Δ} are encoded by the matrices A_{cell}, and Ω_{depth}. Thus, the distribution over α is $\mathbb{A} = \{\mathrm{A}_{cell}\} \cup \{\Omega_{depth}\}$. To sample, we use a continuous and smooth Gumbel-Softmax function (i.e., the sampling on α is differentiable), thus we can learn τ_{A} via Gradient Descent. To tackle our Equation 1, we propose to use Algorithm 1 alternating between updating the trainable parameters of an architecture, sampled in $\mathbb{W}$, and its structural parameters sampled in $\mathbb{A}$ [10] .

Cells and Depth Search Space: All considered model α use the backbone architecture introduced in Figure 1 [10]. As in [13], each block i of α is made of a sequence of identical cells α_{cell}, of variable length δ_i.Therefore, α is parametrized by $\Delta = [\delta_1, \delta_2, ..., \delta_N]$, s.t. $\delta_i < \delta_{max}$, the set of depths for all block i. As in [10], we search for the topology of α_{cell}. Such module is defined as a directed acyclic graph $\mathbb{G}$, with an ordered sequence of $\mathbb{B}$ features (nodes). Each node is the resulting transformation of its $\mathbb{T} = 2$ preceding nodes as: $\mathrm{I_i} = f_{i,j}(\mathrm{I_j}) + f_{i,k}(\mathrm{I_k})$ s.t. $j < i$ & $k < i$ where $\mathrm{I_i}, \mathrm{I_j}, \mathrm{I_k}$ represent nodes of respective indices

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7. Available from http://www.i6doc.com/en/.

i, j, k. $f_{i,j}$, $f_{i,k}$ represent operations sampled from a set $\mathbb{F}$. The final α_{cell} obtained by selecting the function $f_{i,j}$ between each pair of nodes i and j with the highest probability to appear.

Complexity Reducing Loss: In order to reduce the complexity of a searched α, we formulate a complexity loss to minimize [11, 13]: $L_{cpl} = \sum_{\delta_k \in \Delta} L_{cpl-cell} \cdot \delta_k$, a function of the selected depth δ_k for all k blocks. At a cell-level, it writes as $L_{cpl-cell} = \frac{1}{N_{ops}} \sum_{o_{i,j} \in \alpha_{cell}} FLOPs(o_{i,j}) \cdot \omega_{i,j}$, where N_{ops} is the total number of operations, and $\omega_{i,j}$ the probability of selecting an operation $o_{i,j}$ in α_{cell}. It is made differentiable only with respect to the cells' weights $\omega_{i,j}$.

Architecture Depth Search: We look for the optimal depth of each block in α [13]. This aspect is capture by matrix: $\Omega_{depth} = [\Omega_1, ..., \Omega_i, .., \Omega_N]$, s.t. $\Omega_i \in \mathbb{R}^{\delta_{max}}$ where δ_{max} is the maximal depth per block, and Ω_i the depth parameters for block i. These parameters are involved in the training process as follows: $f_{block_{i+1}} = \sum_{\Omega_i^j \in \Omega_i} f_{block_i} \cdot \Omega_i^j$ where f_{block_i} is the feature map resulting from block i. The final feature map is a sum of the original block's feature map weighted by the parameters Ω_i^j of all potential depth j at layer i.

Algorithm 1: Searching for α_{cell} and Δ

```
Input: Two disjoint sets $\mathbb{D}_T$ and $\mathbb{D}_V$, randomly initialized $\mathbb{A}$ and $\mathbb{W}$, batch size;
while not at convergence do
    Sample a batch $\mathbb{D}_t = \{(x_i, y_i)\}_{i=1}^n$ from $\mathbb{D}_T$
    Calculate $L_T = \sum_{i=1}^n L(x_i, y_i)$
    Update $\mathbb{W}$ by gradient descent: $\mathbb{W} = \mathbb{W} - \nabla_{\mathbb{W}} L_T$
    Sample a batch $\mathbb{D}_v = \{(x_i, y_i)\}_{i=1}^n$ from $\mathbb{D}_V$
    Calculate $L_V = \sum_{i=1}^n L(x_i, y_i)$
    Update $\mathbb{A}$ by gradient descent: $\mathbb{A} = \mathbb{A} - \nabla_{\mathbb{A}} L_V$
end
Derive the final architecture $\alpha = \{\alpha_{cell}, \Delta\}$ from $\mathbb{A}$;
Optimize $\alpha = \{\alpha_{cell}, \Delta\}$ on the whole training set for future inference on the test set.
```

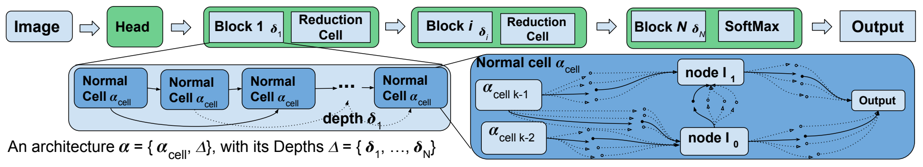


Image
Head
Block 1 δ_1
Reduction Cell
Block i δ_i
Reduction Cell
Block N δ_N
SoftMax
Output
Normal Cell α_{cell}
Normal Cell α_{cell}
Normal Cell α_{cell}
...
Normal Cell α_{cell}
depth δ_1
An architecture $\alpha = \{\alpha_{cell}, \Delta\}$, with its Depths $\Delta = \{\delta_1, ..., \delta_N\}$
Normal cell α_{cell}
$\alpha_{cell\ k-1}$
$\alpha_{cell\ k-2}$
node I_1
node I_0
Output

Fig. 1: An architecture α, with a reduction cell is fixed according to [10].

4 Experimental Setup

Data: The So2Sat LCZ42 classification benchmark [5] consist of samples from 42 urban areas (plus 10 smaller areas for validation and test) across all continents. Including 400,673 co-registered Sentinel-1 SAR (8 real-valued bands) and Sentinel-2 multi-spectral (10 real valued bands) image patches and its respective labels. Each patch is a 32x32 pixel image (320x320 meters spatial resolution). The data set defines 17 LCZ classes, ten urban classes and seven vegetation ones.

Data distribution: We consider two distributions using Sentinel-2 and the labels, exclusively. In $d1$, the training-set consist of samples from 42 cities (352,366 patches). Validation (24,188 patches) and test (24,199 patches) contain samples from 10 different cities, respectively taken from east and west part of each city. In $d2$, train and test sets are randomly draw from $d1$ training-set (80% and 20% respectively). Thus, both have the same data distribution. At search time, the train set is split into two halves: $\mathbb{D}_T$ (training) and $\mathbb{D}_V$ (validation).

Hyperparameters: We set the number of channels in the first layer 16, the number of nodes in α_{cell} to 4, the number of incoming cells to 2, and the initial depth δ in all blocks to 3. The max depth per block is set as 5. We use SGD (105 epochs) for training, and we set the learning rate to 0.025, weight decay to 3e−4, annealed to 1e−3, and momentum of 0.9. The Gumbel-Softmax sampler temperature is set to 10 (linearly reduced to 1). Also, we set $C_{cpl} = 0.01$, to enable balancing the optimization of both L_{NLL} and L_{cpl}. We observed that larger values tend prevent the convergence of the overall objective L. Additionally, following [10], we set $\mathbb{F}$ as: the (1) identity, (2) zeroize, (3) 3x3 depth-wise separate convolutions, (4) 3x3 dilated depth-wise separate convolutions, (5) 5x5 depth-wise separate convolutions, (6) 5x5 dilated depth-wise separate convolutions, (7) 3x3 average pooling, (8) 3x3 max pooling. The stride is set to 1 for normal cells, and to 2 for reduction cells.

Implementation: We used GDAS [10] and TAS [14] source code as basis, and *ResNet08*, *ResNet32* and *ResNet110* models (available on the same repository) as baselines. The experiment were run on a Nvidia V100 GPU.

5 Results

Performance in $d1$: Table 1 presents the mean performance on test data (4 independent runs). S stands for Single Cell Search, C for Complexity Loss, D for Depth Search, and '+' indicates the combination of these settings. The individual effect of Depth Search **(S + D)** and the Complexity Aware Loss **(S + C)** is positive in improving performance on both OA, AA, and Kappa metrics over Single Cell Search **(S)**. The combination **(S + C + D)** does not accumulate their benefice. The model **(S + D)** brings the most improvements while keeping the complexity low over, compared to **(S)**. When compared to manually designed baselines (Table 1 - left), **(S + D)** is a TOP-2 low complexity model (FLOPs and Size), a TOP-2 performer for two accuracy metrics (OA and Kappa), while best on the AA metric. The best selected model **(S + D)** outperforms manually designed baselines of similar complexity. Also, all selected models are performing well in test on $d1$ while being selecting on data distribution $d2$.

Method	*ResNet08*	*ResNet32*	*ResNet110*	*S*	*S+C*	*S+D*	*S+C+D*
OA	60.25	64.85	67.60	64.05	64.25	64.96	64.00
AA	47.86	52.24	53.34	50.39	51.63	54.02	52.20
Kappa	0.565	0.615	0.645	0.61	0.61	0.62	0.61
Size (MB)	0.08	0.27	1.73	0.224	0.165	0.14	0.130
FLOPs (M)	13.53	41.85	253.89	30.37	24.10	20.05	18.42
Block's Depth	NA	NA	NA	3x3	3x3	1x3	1x3

Table 1: Mean performance benchmark in test for setting $d1$.

Performance in $d2$: Table 2 presents the mean performance for 4 independent runs on $d2$. The individual effect of Depth Search **(S + D)** and the Complexity Aware Loss **(S + C)** are non beneficial over Single Cell Search **(S)**

in test. Combining with $(\mathbf{S} + \mathbf{C} + \mathbf{D})$ does not improve performances neither. However, overall performance of all models are very good, suggesting that the $d2$ data distribution is much easier to fit than $d1$. The best model found using the proposed approach is $(\mathbf{S} + \mathbf{D})$, with 98% OA, 91.44% AA and 0.98 Kappa.

Method	*ResNet08*	*ResNet32*	*ResNet110*	*S*	*S+C*	*S+D*	*S+C+D*
OA	96.18	96.89	98.59	98.15	97.81	98.00	97.74
AA	88.91	89.75	91.91	91.61	91.12	91.44	91.08
Kappa	0.958	0.966	0.984	0.980	0.975	0.980	0.975
Size (MB)	0.08	0.27	1.73	0.224	0.165	0.14	0.130
FLOPs (M)	13.53	41.85	253.89	30.37	24.10	20.05	18.42
Block's Depth	NA	NA	NA	3x3	3x3	1x3	1x3

Table 2: Mean performance benchmark in test for setting $d2$.

Compared to modern classification baselines (Table 2 - ResNets), the model $(\mathbf{S} + \mathbf{D})$ is TOP-2 in performance and is very close (0.59% - OA, 0.47% - AA and 0.004 - Kappa) to ResNet110 (TOP-1), while it is nearly 92% smaller (FLOPs and Size) than ResNet110. Also, it outperforms similar complexity competitors.

Confusion: Figure 2 depicts the confusion matrix of (S + D) on $d1$ (left) and $d2$ (right). The confidence increases from blue to yellow. In both cases, the confusion is higher for urban classes (1 to 10). This resemblance suggests that the 42 cities selected on $d1$ and the 10 introduced on $d2$ are similar. Also, the confusion for the vegetation classes (11 to 17) is lower on $d2$ than on $d1$. This difference could be explained by the vegetation disparity around the globe.

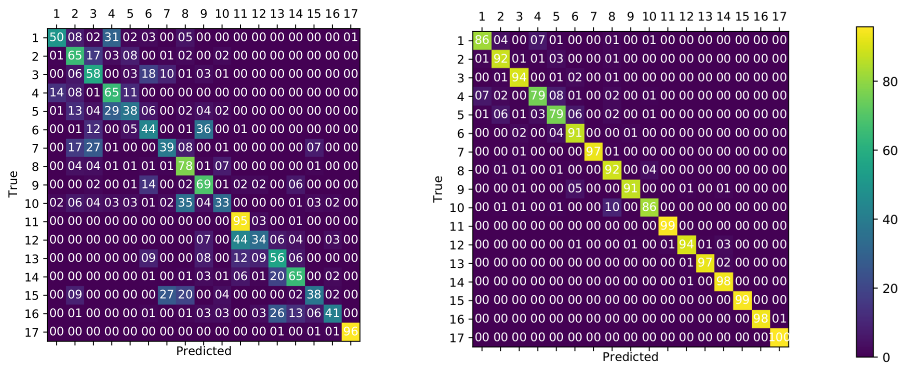


Fig. 2: Confusion matrix for model (S + D) on test $d1$ (left) and $d2$ (right).

6 Conclusions and Future Work

In this paper, we proposed a differentiable search strategy combined with two mechanisms to reduce the model complexity for finding high performing and low complexity classification models. We tested our proposal on a real World problem, the So2Sat LCZ42, using two different data distributions ($d1$ and $d2$). In test, our approach achieves state-of-the-art performance (ResNet-110), while being 91.8% smaller in both FLOPS and size, and outperforms all baselines of similar complexity (ResNet-08). However, in test, these cumulative improvements are not consistent, in particular on the data distribution $d2$. As future work, we propose to investigate how to improve model selection in validation to get the most of fully retrained models in test.

Acknowledgments

Authors acknowledge support by the European Research Council (ERC) under the EU Horizon 2020 (ERC-2016-StG-714087, *So2Sat*), the Helmholtz Association through the Helmholtz AI (ZT-I-PF-5-01), the German Federal Ministry of Education and Research (BMBF) international future AI lab "AI4EO" (01DD20001), and DeToL.

References

[1] Krizhevsky Alex, Sutskever Ilya, and E. Hinton Geoffrey. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[2] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning - Methods, Systems, Challenges.* Springer, 2019.

[3] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *JMLR*, 20(55):1–21, 2019.

[4] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geosci. Remote Sens.*, 5(4):8–36, 2017.

[5] Xiao Xiang Zhu, Jingliang Hu, Chunping Qiu, Yilei Shi, Jian Kang, Lichao Mou, Hossein Bagheri, Matthias Haberle, Yuansheng Hua, Rong Huang, et al. So2sat lcz42: A benchmark data set for the classification of global local climate zones [software and data sets]. *IEEE Geosci. Remote Sens.*, 8(3):76–89, 2020.

[6] John E Ball, Derek T Anderson, and Chee Seng Chan. Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *J. Appl. Remote Sens.*, 11(4):042609, 2017.

[7] Qingpeng Li, Lichao Mou, Qizhi Xu, Yun Zhang, and X. X. Zhu. R3-net: A deep network for multi-oriented vehicledetection in aerial images and videos. *IEEE Tran. Geosci. Remote Sens.*, 2019.

[8] Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Eng. Appl. Artif. Intell.*, 60(January):97–116, 2017.

[9] Liu Hanxiao, Simonyan Karen, and Yang Yiming. Darts: Differentiable architecture search. *ICLR*, 2019.

[10] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. *CVPR*, 2019.

[11] Cai Han, Zhu Ligeng, and Han Song. Proxilessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

[12] Xie Sirui, Zheng Hehui, Liu Chunxiao, and Lin Liang. Snas: Stochastic neural architecture search. *ICLR*, 2019.

[13] Chen Daoyuan, Li Yaliang, Qiu Minghui, Wang Zhen, Li Bofang, Ding Bolin, Deng Hongbo, Huang Jun, Lin Wei, and Zhou Jingren. Adabert: Task-adaptive bert compression with differentiableneural architecture search. *Arxiv*, 2019.

[14] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *NeurIPS*, 2019.