SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis, in Informatics

Landscape Analysis for Multi-Objective Hardware-Aware Neural Architecture Search in Earth Observation Applications

Emre Demir

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis, in Informatics

Landscape Analysis for Multi-Objective Hardware-Aware Neural Architecture Search in Earth Observation Applications

Landschaftsanalyse für die multikriterielle Suche nach hardware-gestuerten neuronalen Architekturen in Erdeobachtungsanwendungen

Author:	Emre Demir
Supervisor:	Prof. Dr. Ph.D. Ghoshdastidar Debarghya
Advisor:	Msc. Traoré Kalifou René Bala
Submission Date:	16.10.2023

I confirm that this master's thesis, is my own work and I have documented all sources and material used.

Munich, 16.10.2023 Emre Demir

Acknowledgments

I thank the following individuals for their invaluable support and guidance during my academic journey:

Supervisor:	*Prof. Dr. Ph.D. Ghoshdastidar Debarghya*, for his valuable feedback and guidance.
Advisors:	*Dr. Ph.D. Andrés Camero*, for his inspirational ideas and support. *Msc. Traoré Kalifou René Bala*, for his support and theoretical insights.

I am also deeply thankful for the unwavering support from:

- My beloved girlfriend, *Zeynep Güliz İskender*, who always stood by my side throughout this journey. She has undeniably played a pivotal role in my achievements.
- My dearest friend, *Oğuzhan Elmalı*, for always being there with words of encouragement and emotional support, helping me to find relief in times of stress.
- My one and only family, whose love and unwavering support have served as my anchor throughout this journey.

Your presence and support have played a pivotal role in my academic journey, and I am truly grateful for your significant contributions to my success.

Abstract

Recent advancements in the field of AutoML, particularly in Neural Architecture Search (NAS), offers exciting possibilities for both researchers and industry practitioners by automating the design of neural networks.

While many NAS studies have focused on well-established datasets, Earth Observation (EO) datasets introduce unique challenges and complexities distinct from those datasets.

This study's primary objective is to establish a dedicated benchmark dataset tailored specifically for EO data and to conduct a landscape analysis on the created benchmark dataset. This dataset will be valuable for the EO researchers who wish to apply NAS methods in their research.

Within the NAS framework, Hardware Aware Neural Architecture Search (HW-NAS) holds particular significance as it enables the optimization of network designs tailored to specific hardware configurations, especially in resource-constrained settings.

Considering the possible needs of HW-NAS on EO domain, our work also includes hardware-dependent metrics in the benchmark dataset.

Additionally, we explore the potential of developing compact surrogate models for data-centric initialization in HW-NAS, thereby trying to enhance the efficiency of the architecture search process.

This research not only addresses the unique demands posed by EO data in the context of NAS but also holds a promise for applications spanning various domains.

Contents

1 Introduction

The field of Automated Machine Learning (AutoML) (Elsken et al., 2019a) is continuously evolving, with a recent focus on automating neural network design, known as NAS.

To support research in NAS, benchmark datasets have been created within predefined search spaces, such as NASBench101 (Ying et al., 2019), which includes all the unique combinations of neural network architectures within a constrained search space. This space includes a range of neural network operations and a specific number of connections between these operations. All of the architectures in the search space have been trained and their metrics have been recorded to a tabular dataset.

While many benchmarks use the CIFAR-10 dataset for evaluation, complex datasets like EO datasets demand more sophisticated approaches due to the unique characteristics of the remote sensing data.

A parallel development is HW-NAS, which tailors network architectures for specific hardware configurations, posing a multi-objective optimization challenge (Benmeziane et al., 2021).

This research has two primary objectives:

- Firstly, it aims to create a specialized dataset for HW-NAS, specifically for EO data, within the NASBench101 search space and conducting a landscape analysis on the created benchmark dataset.
- Secondly, it explores the effect of using compact surrogate models(colin et al., 2023) to sample the search starting points during the initialization of HW-NAS process.

The work is structured as follows: We begin with a review of the current state of the field and related research. Next, we detail the methodology in the third chapter and present experiment results in the fourth chapter. Finally, the fifth chapter concludes and outlines future directions.

2 Related Works

2.1 Neural Architecture Search(NAS)

NAS refers to a collection of approaches aimed at automating the complex task of designing neural network architectures. Each technique automates network structure optimization while offering distinct advantages and insights(colin et al., 2023). NAS seeks to optimize neural network architectures for specific tasks through automated design techniques like random search, bayesian optimization, evolutionary methods, and reinforcement learning (Elsken et al., 2019b). This approach of framing neural architecture design as an optimization problem makes it easier to find good network structures.

NAS has also emerged as a pivotal area within Automated Machine Learning (AutoML) (Elsken et al., 2019b) community. The intersection of NAS and AutoML holds potential to revolutionize deep learning model building. By automating architecture engineering, NAS could make deep learning accessible and applicable across diverse applications.

In this work we will use Pareto Local Search (PLS) (Dubois-Lacoste et al., 2012) as a main NAS algorithm. PLS stands as a robust optimization approach, tailored for addressing multi-objective optimization challenges. It's primary goal is to identify a collection of solutions that collectively constitute the pareto-front, where no single solution dominates all others across all objectives.

2.2 NAS Benchmarks

Benchmarks hold a pivotal role in the advancement of NAS (colin et al., 2023). These datasets are meticulously curated to represent specific search spaces, enabling researchers to systematically assess the performance of various architectures. Notable examples include NASBench101, NASBench201(Dong and Yang, 2020), and NAS-BenchNLP(Klyuchnikov et al., 2020), each serving as valuable repositories of tabulated data that showcase the performance of trained architectures within predefined search spaces.

NASBench101, for instance, distinguishes itself as being the first most known bench-

mark dataset in the field with a compilation of **423,624** unique architectures, evaluated using the CIFAR-10 dataset for classification tasks.

Recently, benchmarks had a switch from being tabular to being surrogate. One of the first main examples of it is NASBench301 (Zela et al., 2020). In tabular benchmarks aim is to train the architectures in the very limited search space exhaustively and write the results into a tabular dataset. The promise of surrogate benchmarks is training a surrogate model which can predict the performance of the architectures in a much larger search space than the tabular benchmarks. This means creating a benchmark with less effort and cost in a wider search space. The search space of NASBench301 (Zela et al., 2020) consists of 10^{21} architectures.

The benchmark used in this work will be the NASBench101 since it has a relatively smaller search space and has real training results.

2.3 Hardware-Aware NAS (HW-NAS)

HW-NAS is a technique that optimizes neural network design to achieve both task performance and adhere to hardware constraints (Benmeziane et al., 2021). This optimization task can be framed as a multi-objective problem, considering objectives like model accuracy and inference latency. The goal of HW-NAS is to discover the best architecture within a specified search space that optimizes both of these metrics.

An example of benchmark dataset designed for HW-NAS is HW-NAS-Bench, tailored to provide metrics related to both performance and hardware within the NB201 search space (C. Li et al., 2021).

The focus of this work is creating a HW-NAS EO Benchmark Dataset containing the hardware-related metrics. Additionally, it explores the data-centric initiation of a PLS for HW-NAS using surrogate models.

The use of surrogate models in HW-NAS typically involves one surrogate model per objective. Some of the recent studies have shown promising results by using a single model with an integrated loss function for all objectives. (Benmeziane et al., 2023).

3 Methodology

3.1 Earth Observation (EO) NAS Benchmark Dataset

This section is dedicated to the cost-effective creation of a specialized EO NAS Benchmark dataset. Architectures in this dataset will be sampled from the search space of NASBench101 with a systematic sampling strategy rooted in a random walk pattern (Xia et al., 2020).

3.1.1 Earth Observation Data

We chose "So2Sat LCZ42" (Zhu et al., 2020) dataset to create an EO benchmark. EO datasets are essential for various applications, including climate change monitoring, urban development analysis, and disaster management. Traditional land use/land cover (Land Use Land Cover (LULC)) maps often rely on subjective, culture-dependent categorizations like urban and non-urban, which can vary globally. To address this, local climate zones (Local Climate Zones (LCZs)) offer an objective and culture-independent classification system.

Further information about the dataset can be found on appendix (Dataset Details).

3.1.2 Sampling Strategy

In order to establish an unbiased and representative sample, firstly a random sampling strategy was employed to select an initial set of 30 architectures from the extensive NASBench101 search space. The selection process was conducted through the utilization of a random number generator, aimed to emulate a uniform distribution within the search space. Each architecture has been represented as an object in an array, and 30 indexes have been randomly sampled from among all these architectures.

After selecting the initial 30 architectures, random walks (Xia et al., 2020) on each of them for 14 steps has been conducted. At each step, we explored the neighborhood of the current architecture within a hamming distance of 1. The decision to move to the next architecture was made randomly, ensuring a stochastic exploration process.

A representation of the random walk process has been visualized on figure 3.1.

- **a)** in Figure 3.1, represents the state of the search space after the initial sampling. Each square represents one architecture, and the ones marked with the red color are the initial randomly sampled points.
- **b)** in Figure 3.1, represents the state of the search space after taking the first random walk step. Architectures sampled after the first step are colored with blue. These are randomly chosen neighbors of the initially sampled points.
- **c)** in Figure 3.1, represents the state of the search space after taking the second random walk step. Architectures sampled after the second step are colored with orange. New steps are sampled randomly from the neighborhood of the recent architecture in the previous step. The process continues by randomly sampling architectures from the neighborhood of the recently sampled architecture in the previous step for that specific walk, until it reaches a total of 14 steps.

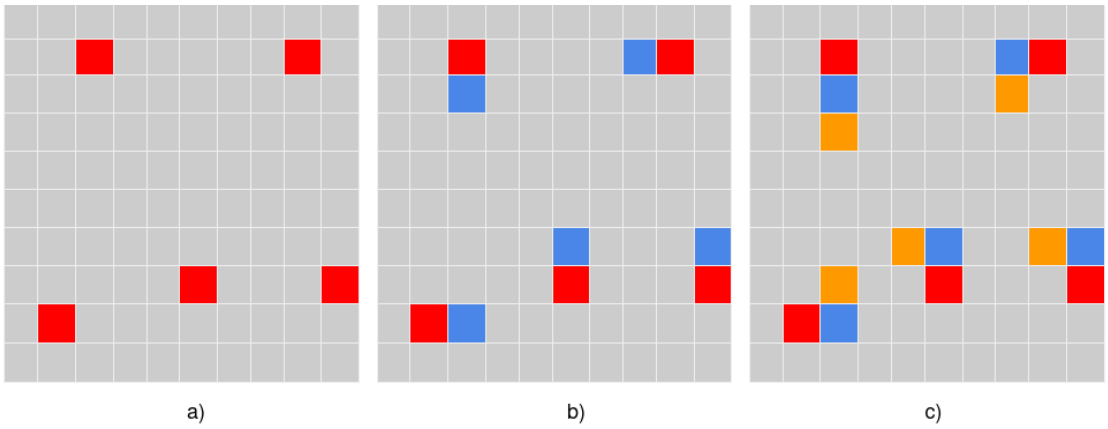

Figure 3.1: Random Walk

With sampling 30 points initially and taking 14 random walk steps on them, we have sampled 450 unique architectures in total$((30) + (14 * 30) = 450)$. The selection of 450 architectures was determined after considering the trade-off between computational resources and the need for a representative sample. While there is no fixed standard for sample size in NAS research, this size was deemed adequate for achieving our research objectives.

Details on representing the search space as a graph and calculating the distances between different architectures has been added to the appendix (Search Space Representation).

Following the successful sampling of these architectures, the next phase involves training. Leveraging the So2Sat LCZ42 dataset (Zhu et al., 2020), the architectures are sub-

jected to a classification task. The performance of these architectures are documented, including metrics such as test and validation accuracy. Additionally, hardware dependent metrics like average inference time, standard deviation of the inference time, architecture size, number of parameters of the architecture and Multiply–accumulate (MAC)s are also recorded. "MAC" operation entails the multiplication of two numbers, followed by the accumulation of the resulting product into an accumulator (Abdelgawad, 2013).

Furthermore, at regular intervals of every 2 epochs, checkpoints of the architectures are preserved. During the training process, all sampled architectures trained for 108 epochs in total. This aligns with the number of epochs employed in the original NASBench101.

Details of the implementation and recording process has been added to the appendix (Implementation Details).

3.1.3 Landscape Analysis

In the first part of the analysis, the focus will be the distributions of the macro and micro accuracies of the final model. Since there are 17 classes in the So2Sat (Zhu et al., 2020) dataset, assessing the both accuracies separately gives more insights. Then, the distributions of MACs across the all models and the inference latency will be assessed.

3.2 Data-Centric Initilization of NAS

This part focuses on the explanation of methodologies on data-centric initilization of a HW-NAS strategy on NASBench101's search space.

3.2.1 Benchmark Dataset

In this experiment, we will exclusively examine the impact of data-centric initialization of HW-NAS. To achieve this, we will utilize the pre-existing tabular NASBench101 dataset that has already been assessed on CIFAR-10. Rather than assessing solutions for every sample in the search space, we will query the benchmark dataset.

3.2.2 Surrogate Models

Since the problem here is HW-NAS a multi-objective optimization problem, the method aims to create 2 different XGboost (Chen and Guestrin, 2016) surrogate models firstly. One for the performance metric and the other for the hardware related metric. The number of objectives in this search can be increased.

In this demonstration, "validation accuracy" of the models after the 108th's epoch will be the performance metric and the "train time" of the models will be accustomed as the hardware dependent metric. The metric "train time" has been chosen as hardware dependent metric since there are limited hardware dependent metrics in existing NASBench101. It is just a representative of a hardware dependent metric, it can be any other hardware dependent metric.

Proposed method trains the models on very limited data (corresponding to less than 1% of the whole search space). Those surrogate models predicts the performance of the all architectures in the search space (423,624 in total) to find the pareto-front. Then, we randomly pick 20 of those non-dominated pareto-front points as starting points for the HW-NAS.

In the end, this work aims to observe if there any computing cost benefits in terms of number of solution evaluations to find the best pareto-front compared to the HW-NAS initialized with the randomly picked points.

3.2.3 Search Method

The chosen search method for the HW-NAS is PLS. There will be 20 initialization points for the search algorithm. These initialization points will be from either the pareto-front of the trained surrogate models or randomly sampled.

In PLS as described in (Dubois-Lacoste et al., 2012), in every step all neighbours of the one sampled point will be trained. If there are non-dominated neighbours among those, non-dominated neighbours will be added to the non-dominated points archive(pareto-front) and also to a queue which is keeping the non-dominated points according to the order they have been added. Then in every step the oldest neural network architecture in the queue will popped out and it's neighborhood will be explored. This process will be repeated till it reaches the solution evaluation limit.

In this context, "non-dominated" means that a solution performs better than or equal to another solution in at least one aspect without performing worse in any other aspect.

Pseudo-code of the PLS can be found on appendix (Implementation Details).

3.2.4 Comparison with Randomly Initialized HW-NAS

While comparing, 30 different experiments will be made with 30 different random seeds to see if the effect is significant.

After conducting the PLS initialized with the pareto-front from the surrogate models, another PLS will be initialized with the randomly picked starting points.

All searches will have a limit of solution evaluations. Since the number of solutions can vary with the number of steps a lot and the final goal is observing the cost efficiency

for model training, the limiter for these searches will be number of solution evaluations.

In this context, "solution evaluation" means the final metrics of a fully trained neural network architecture in the search space.

On this regard, for the fair comparison of the solution evaluations, search solution limits are set like below:

> **Randomly Initialized PLS Solution Limit = Number of Solutions Used in Surrogate Model Training + Search Solution Limit**
>
> **Surrogate Initialized PLS Solution Limit = Search Solution Limit**

This process will be made for 4 different values of solution evaluation limits for PLS.

4 Results and Experiments

4.1 Landscape Analysis of EO HW-NAS

Once the dataset was established, we recorded various metrics, including accuracy, average inference latency, MACs, the total number of model parameters, and the model size. In this part a few analysis will be conducted using those metrics.

	Macro Acc.↑	Micro Acc.↑
So2Sat LCZ42	41.13%	58.67%
CIFAR-10	89.62%	89.62%

Table 4.1: Comparison of the average micro and macro classification accuracy between the So2Sat LCZ42 and CIFAR-10 landscapes

On figure 4.1, unlike So2Sat dataset, there is not a visible difference between macro and micro accuracy scores of CIFAR-10 dataset, which is perfectly balanced. Additionally, the higher micro classification accuracy implies that the trained models tend to prioritize the most prevalent class and struggle to effectively address the class imbalance. These findings further corroborate the assertion that the So2Sat LCZ42 landscape presents a significantly more challenging scenario.

For the purpose of double-checking the results, the class distributions of So2Sat dataset has been plotted. Figure 4.1, supports the first ideas of class imbalance. Then, it is possible to say that the reason behind the lower performance metrics is having different class distributions for train, validation and test sets. It is visible that in the train set, class number 16 dominates the whole dataset, however for the validation and test sets class 7 dominates the sets.

Another visible insight from the class distributions is some classes are definitely underrepresented in the dataset. Underrepresented classes might has the biggest effect on performance drop between training and test accuracies. Since the aim of this work is creating an EO-NAS Benchmark dataset, fine tuning the model and data has not been made.

In figure 4.2, it is evident that there is a notable concentration of macro accuracy values around 0.5 for the test set, whereas the micro accuracy values tend to cluster

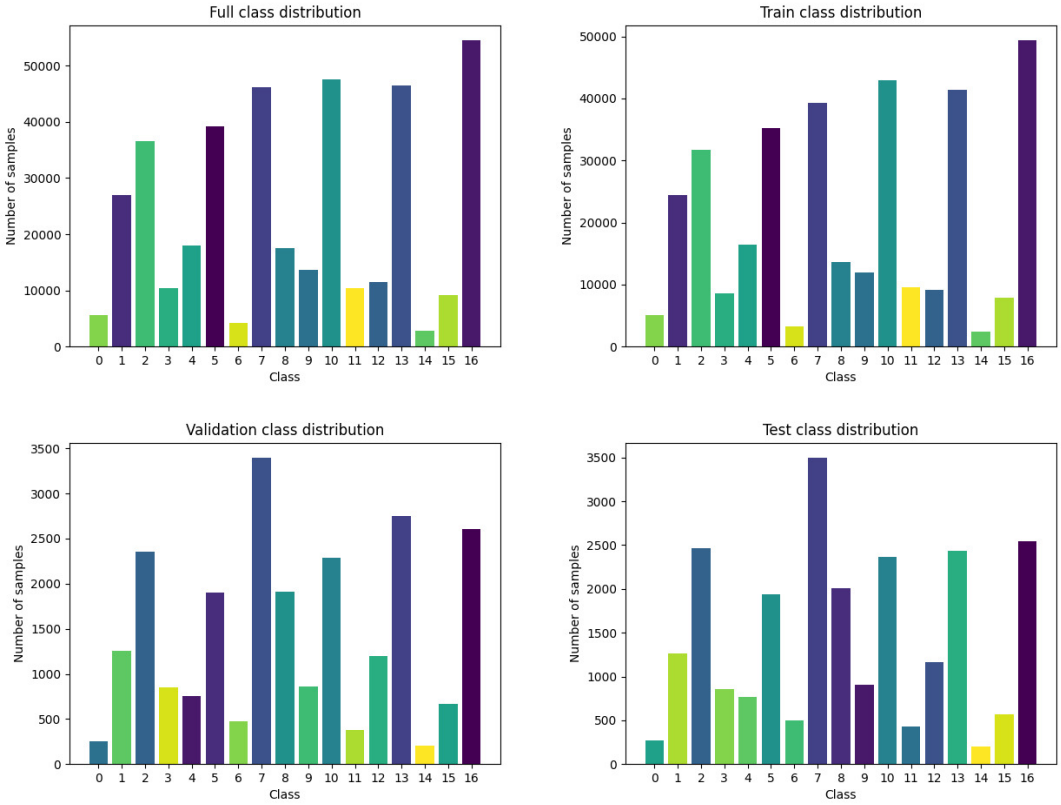

Figure 4.1: Class distributions of the So2Sat LCZ42 dataset.

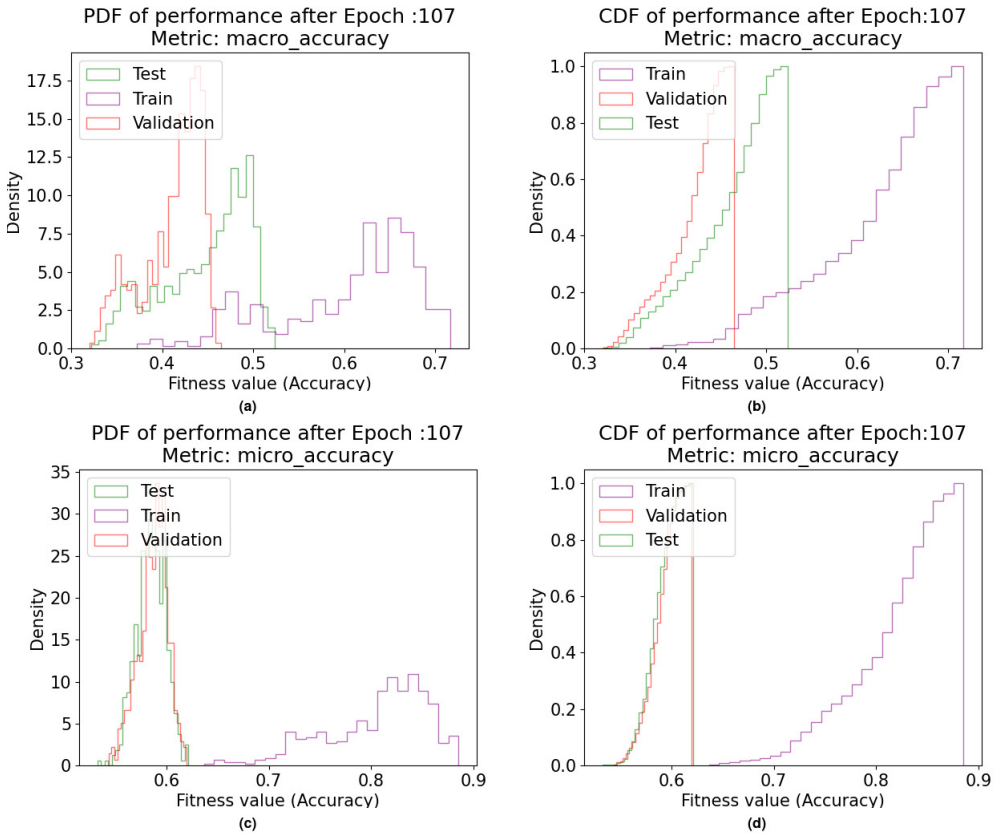

Figure 4.2: Comparison of macro and micro accuracies with PDF and CDF distributions

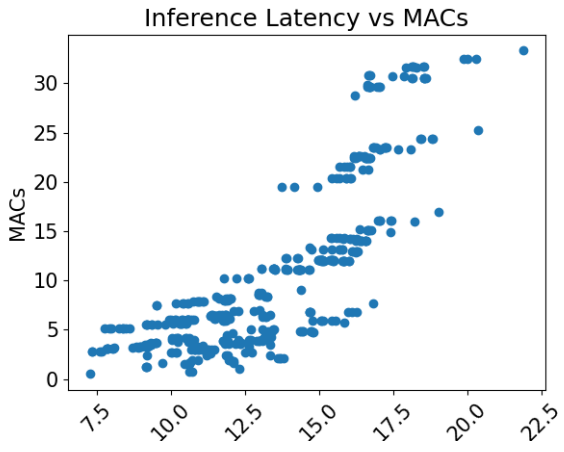

Figure 4.3: Inference Latency and MACs comparison.

more closely around 0.6. This observed trend raises the hypothesis that there might exist a class imbalance issue within the So2Sat dataset. This observed disparity in distributions can be attributed to the distinct calculation methods of macro accuracy, which computes the mean of individual class accuracies, and micro accuracy, which computes the mean of all predictions.

Upon examining the relationship between the count of (MACs) and Inference Latency, an observable correlation between inference latency and the quantity of MACs per model becomes apparent. This observed correlation is inherently logical, as MACs represent the number of computational operations, and it is reasonable to expect that latency may increase in tandem with the growth in computational operations.

In our landscape analysis, the final metric we consider is ruggedness. Ruggedness is calculated using the inverse of autocorrelation, as outlined in (Traoré et al., 2021). In the context of NAS, ruggedness serves as an informative metric.

For NAS, a positive ruggedness value suggests higher correlation between successive steps in the search process, potentially indicating a less diverse search space. Conversely, a negative ruggedness value implies a more challenging search space with greater variation between configurations. Ruggedness provides insights into NAS landscape characteristics and informs optimization strategies.

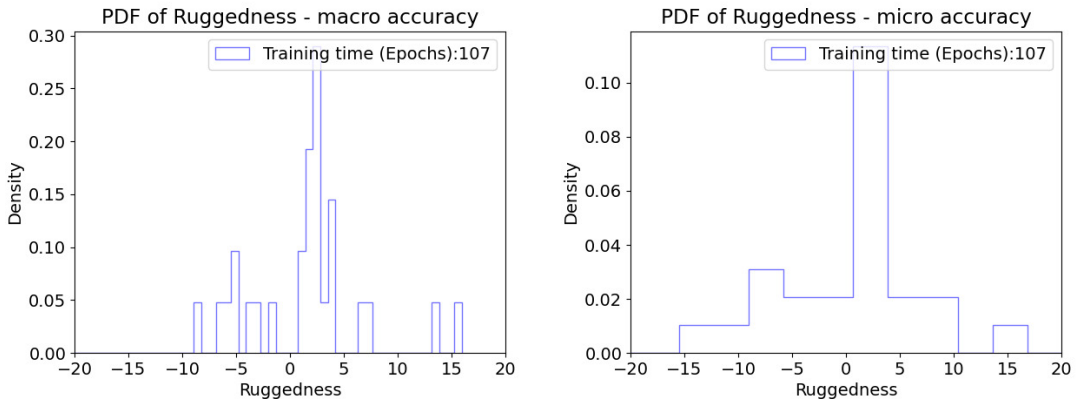

Figure 4.4: Ruggedness distributions for micro and macro accuracies.

In the generated EO HW-NAS dataset, there is a high density for ruggedness values between 0 and 5 Figure 4.4. This indicates that the search space was neither highly homogeneous or diverse. Furthermore, variations in the density distribution suggest the presence of both more homogenous and more diverse regions within the search space. The diversity of the value densities in for micro and macro accuracies can be interpreted as the class imbalance of the data.

4.2 Data Centric Initialization of HW-NAS

This second part will focus on the results obtained by the experiments of data-centric initialization of HW-NAS.

Firstly, a few surrogate models with different training dataset sizes will be trained to see what is the most cost-effective training set size for a surrogate model in Neural Architecture Search Benchmark 101 (NB101) search space. After deciding on the most cost-effective surrogate model size, a pareto-front from the whole search space will be gathered using the two surrogate models one for hardware related metric and the other for the performance metric.

Using this pareto-front, a PLS will be initialized and will explore the search space till it reaches the limit of solution evaluations. The pareto-front of the PLS will be assesed using the metrics like Hypervolume, Diameter, Pareto Optimal Distance Average (Liefooghe, 2022) and Best Training Time, Best Accuracy.

The results of those metrics will be compared with the randomly initialized PLS to see if they are making a difference in terms of performance.

4.2.1 Training Set Size Analysis for the Surrogate Models

To pick a best cost-effective training dataset size, few surrogate models has been trained on NB101 data. Following data sizes has been used for training a surrogate model: **[300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 5000]**.

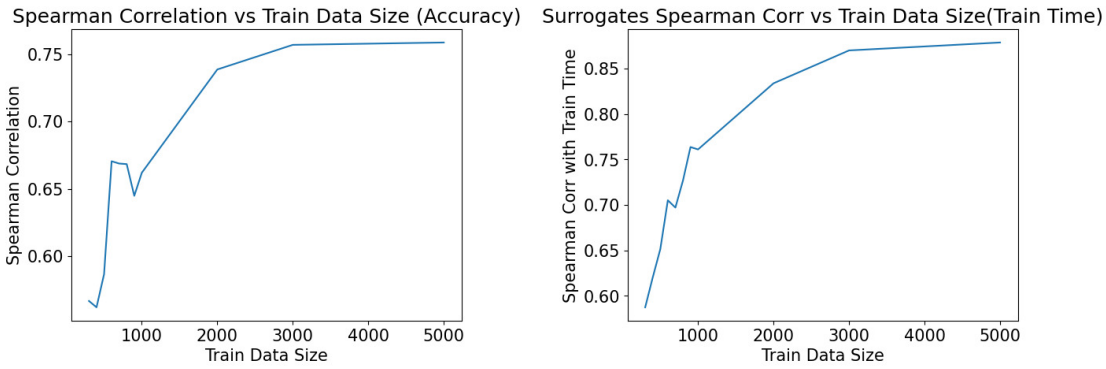

Figure 4.5: Spearman correlations of surrogate models with the test set.

As it seen in the train data size analysis Figure 4.5, after the train data size 2000 the performance starts to being a plateau. Smaller training data size would be better, however for the sizes smaller than 2000 it seems like performance is still not so

promising. As it seems that surrogate model which predicts the train time has higher performance than the accuracy model.

4.2.2 Analysis of HW-NAS

To observe the results of the proposed solution better, this experiment has been repeated 30 times with different random seed values.

The important metrics to compare for us are, best training time and the best validation accuracy of the pareto-front. Best training times from the 30 different pareto-fronts from data-centric initialized PLS and randomly initialized PLS has been plotted for different number of solution evaluation limits.

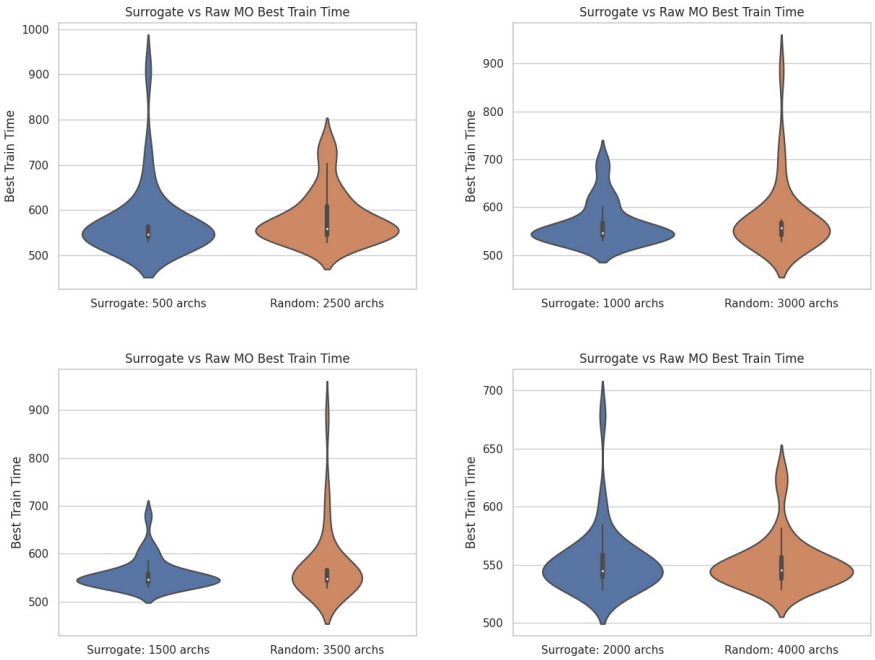

Figure 4.6: Violin Plot of "Best Train Time" from 30 different Runs.

Overall, in the Figure 4.6 the violin plot shows that surrogate initialized PLS has a lower median "best train time" than the randomly initialized PLS. It seems like with the increasing solution evaluation limit, the median for both of the PLS is reducing. It is also important to note that there is 1 outlier for the surrogate initialized runs around the value 700.

To observe if there is a real significant difference between the distributions we have applied Mann-Whitney U-test (Neuhäuser, 2011). Mann-Whitney U-test has been

chosen because it does not assume a specific distribution for the data and is robust against outliers. P value for all the tests in this section is: 0.05.

It turned out that only significant difference in various distributions, between surrogate initialized 500 and random initialized 2000 solution evaluation limits. Rest of the distributions are not significantly different.

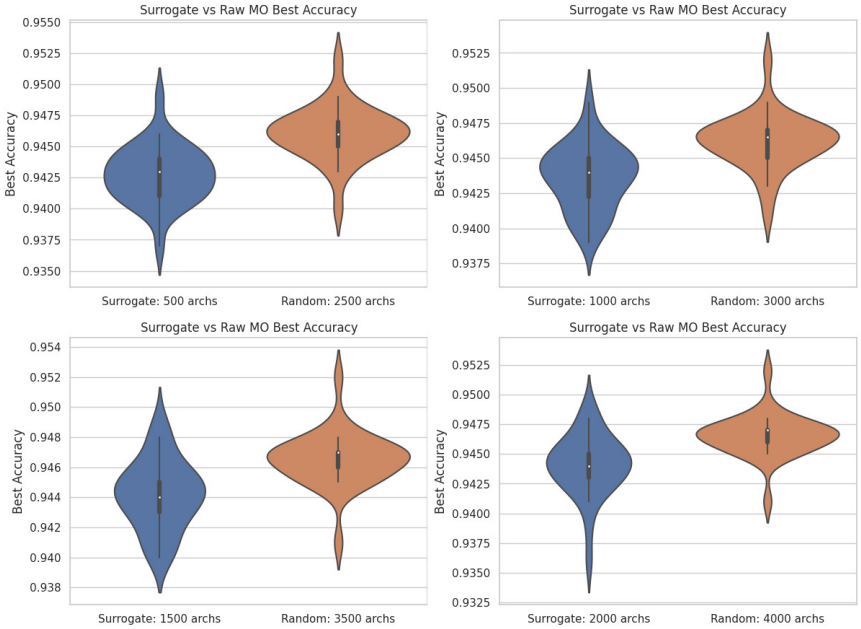

Figure 4.7: Violin Plot of "Best Validation Accuracy" from 30 different Runs.

For the second objective "best validation accuracy", it is noted that the randomly initialized runs performs better than the surrogate initialized ones in figure 4.7. One reason behind this might be the performance of the validation accuracy predicting surrogate model. In figure 4.5, the highest spearman correlation value for the accuracy predicting surrogate model was around 0.75. This might cause the initialization stuck on some low accuracy areas of the search space. It is also important to note that, the metric "best validation accuracy" doesn't have a wide range in pareto-fronts, namely it is from 0.935 to 0.955.

Mann-Whitney U-test showed that all distribution comparison pairs are significantly different from each other for the "best validation accuracy" metric.

The last metric to compare will be "average pareto distance", it is the average distance value between the solutions in pareto-front. This metric can give insights about the diversity of solutions in the pareto-front. In figure 4.8, it is obvious that the diversity of the solutions in surrogate-initialized run is distinctly higher and kept being higher even

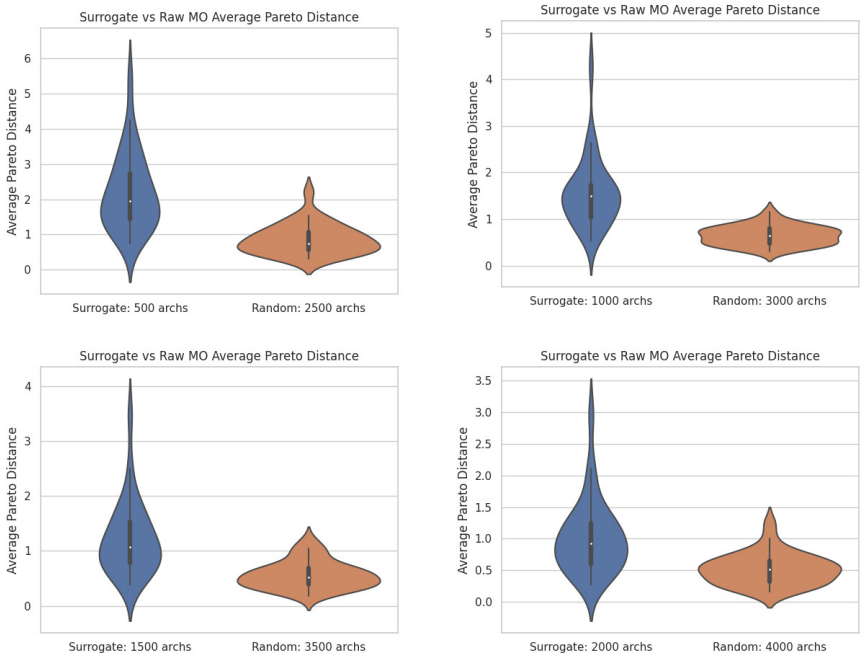

Figure 4.8: Violin Plot of "Pareto Distance Average" from 30 different Runs.

after exploring the search space more. This can mean that, if the surrogate-initialization of the search already has non-dominated points in their local areas then the further exploration progressed in the local areas which are distinct from each other. On the other hand, in the randomly initialized run one solution might be dominated all the other solutions and the search progressed on the locality of that one solution.

Mann-Whitney U-test showed that all distribution comparison pairs are significantly different from each other for the "average pareto distance" metric.

5 Conclusion and Future Work

5.1 Conclusion

In this master's thesis of "Landscape Analysis for Multi-Objective Hardware-Aware Neural Architecture Search in Earth Observation Applications", an EO HW-NAS dataset on NASBench101's search space has been created and an analysis of data-centric initialization of HW-NAS has been made.

The dataset will be helpful for the EO researchers who would like to optimize their neural network designs based on their data and their hardware.

In the landscape analysis and the creation of the EO HW-NAS dataset, a comprehensive landscape study was conducted, exploring various metrics which is giving valuable insights about the EO benchmark dataset and the search space.

As a further step, this dataset can be used by EO researchers to develop surrogate models or to improve their neural networks specific to their dataset and hardware.

On the second part, an analysis explored data-centric initialization in HW-NAS using compact surrogate models with experiments on the NASBench101.

Surrogate-initialized HW-NAS showed lower median than the randomly initialized HW-NAS on "best train time" for small solution evaluation limits. Randomly initialized HW-NAS outperformed the surrogate-initialized HW-NAS on "validation accuracy". The possible reason might be non-converging surrogate model and the short range of "validation accuracy" metric in the dataset.

Additionally, surrogate-initialized runs exhibited higher diversity in pareto-front considering the "average pareto distance" metric. This indicates the discovery of a broader range of solutions for HW-NAS.

5.2 Future Work

As a future directions for this work, a new EO HW-NAS benchmark dataset can be generated in a wider search space with a well-balanced EO dataset to help EO researchers to optimize their networks for their specific data and hardware. Additionally, a second version of the EO HW-NAS would be created as a learning curve prediction based surrogate model just like in the (Zela et al., 2020).

One of the causes of performing worse than randomly initialized PLS for "validation accuracy" might be a non-converging surrogate model. As a future work, data-centric initialization of HW-NAS would be done with different compact surrogate models than the "XGboost". Recent advances in the learning curve prediction seems to give promising results (Zela et al., 2020) and hence it would give better initialization.

All code material for these works are publicly available:

- EO HW-NAS dataset: *https://github.com/emreds/tum-dlr-automl-for-eo*.
- Data-Centric initialization of PLS: *https://github.com/emreds/data-centric-nas*.

6 Appendix

6.1 Dataset Details

LCZs consist of 10 built and 7 natural classes based on climate-relevant surface properties, such as building height, vegetation, and heat output. LCZs provide a consistent classification methodology applicable worldwide for tasks like infrastructure planning and disaster mitigation.

To create the dataset, 400,673 pairs of Sentinel-1 and Sentinel-2 images were labeled. These satellites provide data with various spectral bands, which were pre-processed into 32x32 dimensions. The dataset includes diverse regions from different continents, ensuring its quality and applicability. A meticulous labeling process involved a learning phase, manual labeling with the guidance of a classification map, and validation steps to minimize errors and personal bias.

After labeling, the dataset was fine-tuned, including correcting outliers and balancing class sizes. The final dataset comprises 56GB of data, split into training, test, and validation sets. Trained models achieved overall accuracy between 0.51 and 0.61, indicating promise but still leaving room for improvement to meet land cover mapping requirements (0.85-0.9 accuracy).

6.2 Search Space Representation

6.2.1 NASBench101 Details

In the NASBench101 the neural network architecture search space is defined in three main parts: the head (input layer), body (comprising three identical block structures with down-sampling modules), and tail (output layer). Each block structure consists of three cells, and these cells are represented as Directed Acyclic Graphs (DAGs)(Ying et al., 2019). A visualized representation of the neural network can be seen in figure 6.1.

A single cell's DAG contains up to seven nodes: five intermediate nodes, one input node, and one output node. These nodes are labeled with specific fixed operators, including max-pooling 3x3, convolution layer 3x3, and convolutional layer 1x1. To represent a solution within this search space, an upper-triangular adjacency binary matrix of size 7x7 is used, derived from the DAG.

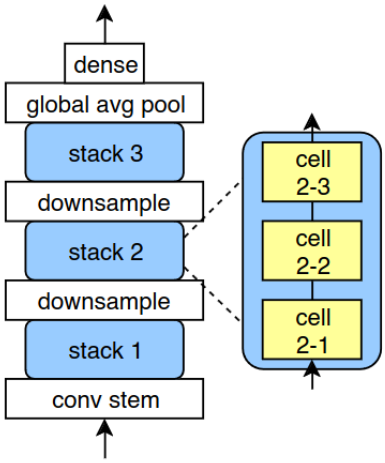

Figure 6.1: Visualization of a cell based neural network architecture in NAS-Bench101 search space. Adopted without changes from (Ying et al., 2019). This figure is under CC BY-NC-SA 4.0 license, https://creativecommons.org/licenses/by-nc-sa/4.0/.

6.2.2 2D Representation Details

With the aim of sampling the architectures using random walk, the whole search space should have been represented as a graph. To represent the whole search space as a 2D graph, our first aim was representing an architecture in the search space as binary matrix. Binary matrix representation specifically chosen because it will be also helpful to calculate hamming distances between different architectures.

In the original work of NASBench101 an architecture represented as 7x7 binary matrix(Ying et al., 2019), however this representation was not covering all changes. In the original representation a bit change corresponding to an operation was not actually corresponding to an architecture in 1 hamming distance, rather it was corresponding to an architecture in 2 hamming distance. Since changing one operation will also effect the input and output edges. To have more accurate representation this work uses 17x17 matrix representation.

Considering an architecture consisting of 7 layers, it's important to note that the initial and final layers are pre-defined as input and output, respectively. This leaves us with 5 layers within which architectural adjustments can be made. Among these 5 layers, there exist a total of 15 distinct variant operations, which is a result of having only three types of operations available: 1x1 convolutions, 3x3 convolutions, and 3x3 maxpooling.

It should be emphasized that the first and last layers are fixed and unchanged, resulting in 2 fixed operations. Consequently, the encoding of each architecture can be effectively represented using a matrix sized (15 variant operations + 2 fixed operations) x (15 variant operations + 2 fixed operations). In the end each architecture can be represented as 17x17 adjacency matrix(Traoré et al., 2023). Visualization of this representation can be seen in figure 6.2.

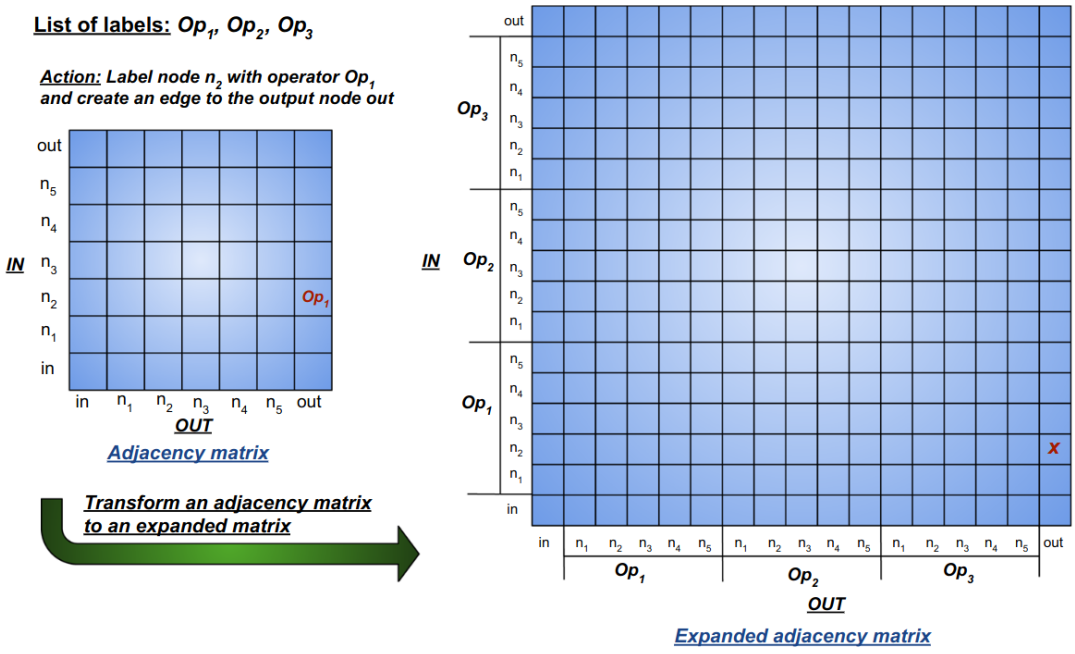

Figure 6.2: A visual description of the transformation of a standard adjacency matrix into an expanded adjacency matrix for a neural cell in NASBench-101. Adopted without changes from (Traoré et al., 2023). This figure is under CC BY 4.0 license, https://creativecommons.org/licenses/by/4.0/.

To find the hamming distance between 2 binary matrices effectively, XOR operations was applied.

As an additional note, it's important to mention that, during the experiment involving data-centric initialization of HW-NAS the neighborhood function for NASBench101 in NASLib (Ruchte et al., 2020) software library has been used for the sake of simplicity. This neighborhood function considers the architectures with one different operation or one different edge connection as neighbours.

6.3 Implementation Details

6.3.1 Architecture Training Pipeline

To create the EO NAS benchmark dataset, a standardized training pipeline was required to evaluate the neural network architectures in the search space. To do this, with considering the architecture and capabilities of our High-Performance Computing (HPC) cluster (Jülich Supercomputing Centre, 2021), the following pipeline has been created on figure 6.3.

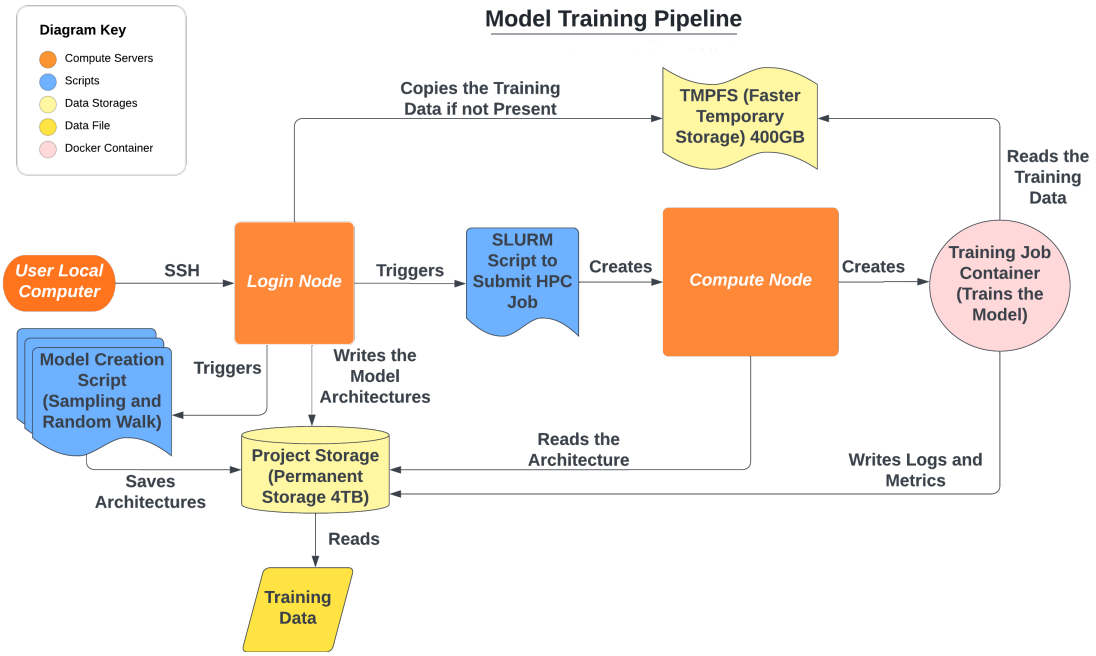

Figure 6.3: Our Neural Architecture training pipeline to create EO NAS benchmark dataset.

The pipeline works as follows,

- User logs in to the login node of our cluster with SSH connection.
- Triggers the scripts to sample architectures from NASBench101's search space with random sampling and random walk.
- After sampling, all these architectures saved in a disk in a PyTorch (Paszke et al., 2019) format.

- Training data stays in the permanent storage, however another script triggers the copying data to fast storage.
- A script for training architectures, submits all the architecture trainings as a separate job to SLURM(resource management tool) (Yoo et al., 2003). Submitting separately gives us a possibility to train architectures in parallel and the process becomes more fault-tolerant.
- SLURM creates a job on HPC which reads the architectures in PyTorch format and starts the training process.
- Training job reads the data from temporary faster storage.
- Every training job runs on 4 GPUs using Data Distributed Parallel (S. Li et al., 2020) feature of PyTorch. This lets us train the architectures using the power of 4 GPUs. It significantly decreased our training duration, cutting it by approximately 70%, resulting in reduced expenses.
- For every trained architecture the metrics has been recorded to the permanent disk.
- Also in every 2 epochs, trained architecture saved to a disk as a checkpoint.
- We compile all the assessed metrics into a file that can be utilized for subsequent research.

6.3.2 PLS Implementation

Below there is a pseudo-code implementation of the PLS used in the data-centric initialization of the HW-NAS experiment.

Algorithm 1 ParetoLocalSearch input: initial_solution

Initialize an empty pareto-front P
Initialize an empty queue Q
Initialize evaluation limit L
Initialize evaluated solution count C
Add *initial_solution* to the set of current solutions S
Add *initial_solution* to Q
while Q is not empty && $C < L$ **do**
    Remove the first solution s from Q
    Generate a set N of neighboring solutions for s
    Evaluate the solutions in N and add non-dominated solutions to P
    Add the count of evaluated solutions to C
    Add non-dominated solutions in N to Q
    Remove dominated solutions from S and P
end while
Output: The pareto-front P

The full implementation for the PLS can be found under:
https://github.com/emreds/data-centric-nas/blob/main/src/pls.py

6.3.3 Hardware Details

An important thing to mention in this work is the hardware type that have been used for testing the models. After the training all models have been deployed on **NVIDIA A100** GPUs and their respective hardware related metrics calculated accordingly.

Abbreviations

NAS Neural Architecture Search

HW-NAS Hardware Aware Neural Architecture Search

EO Earth Observation

MAC Multiply–accumulate

NB101 Neural Architecture Search Benchmark 101

PLS Pareto Local Search

AutoML Automated Machine Learning

LULC Land Use Land Cover

LCZs Local Climate Zones

HPC High-Performance Computing

List of Figures

List of Tables

Bibliography

Elsken, T., Metzen, J. H., & Hutter, F. (2019a). Neural architecture search. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automatic machine learning: Methods, systems, challenges* (pp. 69–86). Springer.

Ying, C., Klein, A., Real, E., Christiansen, E., Murphy, K., & Hutter, F. (2019). Nas-bench-101: Towards reproducible neural architecture search. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 12334–12348.

Benmeziane, H., Maghraoui, K. E., Ouarnoughi, H., Niar, S., Wistuba, M., & Wang, N. (2021). A comprehensive survey on hardware-aware neural architecture search.

colin, C. W., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Hutter, F., White, C., & Dey, D. (2023). Neural architecture search: Insights from 1000 papers.

Elsken, T., Metzen, J. H., & Hutter, F. (2019b). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20*, 1–21.

Dubois-Lacoste, J., López-Ibáñez, M., & Stützle, T. (2012). Pareto local search algorithms for anytime bi-objective optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7245 LNCS*, 206–217. https://doi.org/10.1007/978-3-642-29124-1_18/COVER

Dong, X., & Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *8th International Conference on Learning Representations, ICLR 2020*.

Klyuchnikov, N., Trofimov, I., Artemova, E., Salnikov, M., Fedorov, M., Filippov, A., & Burnaev, E. (2020). Nas-bench-nlp: Neural architecture search benchmark for natural language processing. *IEEE Access*, *10*, 45736–45747. https://doi.org/10.1109/ACCESS.2022.3169897

Zela, A., Siems, J., Zimmer, L., Lukasik, J., Keuper, M., & Hutter, F. (2020). Surrogate nas benchmarks: Going beyond the limited search spaces of tabular nas benchmarks. *ICLR 2022 - 10th International Conference on Learning Representations*.

Li, C., Yu, Z., Fu, Y., Zhang, Y., Zhao, Y., You, H., Yu, Q., Wang, Y., & Lin, Y. (2021). Hw-nas-bench:hardware-aware neural architecture search benchmark. *ICLR 2021 - 9th International Conference on Learning Representations*.

Benmeziane, H., Ouarnoughi, H., El Maghraoui, K., & Niar, S. (2023). Multi-objective hardware-aware neural architecture search with pareto rank-preserving surro-

gate models. *ACM Trans. Archit. Code Optim.*, *20*(2). https://doi.org/10.1145/3579853

Xia, F., Liu, J., Nie, H., Fu, Y., Wan, L., & Kong, X. (2020). Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *4*(2), 95–107. https://doi.org/10.1109/tetci.2019.2952908

Zhu, X. X., Hu, J., Qiu, C., Shi, Y., Kang, J., Mou, L., Bagheri, H., Haberle, M., Hua, Y., Huang, R., Hughes, L., Li, H., Sun, Y., Zhang, G., Han, S., Schmitt, M., & Wang, Y. (2020). So2sat lcz42: A benchmark data set for the classification of global local climate zones [software and data sets]. *IEEE Geoscience and Remote Sensing Magazine*, *8*(3), 76–89. https://doi.org/10.1109/MGRS.2020.2964708

Abdelgawad, A. (2013). Low power multiply accumulate unit (mac) for future wireless sensor networks. *2013 IEEE Sensors Applications Symposium Proceedings*, 129–132.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Traoré, K. R., Camero, A., & Zhu, X. X. (2021). Fitness landscape footprint: A framework to compare neural architecture search problems. *CoRR*, *abs/2111.01584*.

Liefooghe, A. (2022). *Landscape analysis and heuristic search for multi-objective optimization*.

Neuhäuser, M. (2011). Wilcoxon–mann–whitney test. In M. Lovric (Ed.), *International encyclopedia of statistical science* (pp. 1656–1658). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_615

Traoré, K. R., Camero, A., & Zhu, X. X. (2023). A data-driven approach to neural architecture search initialization. *Annals of Mathematics and Artificial Intelligence*, 1–28. https://doi.org/10.1007/S10472-022-09823-0/METRICS

Ruchte, M., Zela, A., Siems, J., Grabocka, J., & Hutter, F. (2020). Naslib: A modular and flexible neural architecture search library.

Jülich Supercomputing Centre. (2021). JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Juelich Supercomputing Centre. *Journal of large-scale research facilities*, *7*(A138). https://doi.org/10.17815/jlsrf-7-183

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Yoo, A. B., Jette, M. A., & Grondona, M. (2003). Slurm: Simple linux utility for resource management. In D. Feitelson, L. Rudolph, & U. Schwiegelshohn (Eds.), *Job scheduling strategies for parallel processing* (pp. 44–60). Springer Berlin Heidelberg.

Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., & Chintala, S. (2020). Pytorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, *13*, 3005–3018. https://doi.org/10.14778/3415478.3415530