



**Politecnico
di Torino**

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (37th cycle)

Practical Neural Architecture Search: Efficient Methods and Evaluation

Luca Robbiano

* * * * *

Supervisor

Prof. Barbara Caputo

Doctoral Examination Committee:

Prof. Manuel Roveri, Referee, Polytechnic of Milan

Prof. Paolo Garza, Polytechnic of Turin

Prof. Nicoletta Noceti, University of Genoa

Prof. Nicola Strisciunglio, University of Twente

Prof. Pietro Zanuttigh, University of Padua

Politecnico di Torino
2026

Declaration

I hereby declare that the contents and organization of this dissertation constitute my own original work and do not compromise in any way the rights of third parties, including those relating to the security of personal data.

Luca Robbiano
2026

This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

*I dedicate this thesis to my family and
friends*

Acknowledgements

I would first like to express my gratitude to my supervisor, Prof. Barbara Caputo, for giving me the opportunity to conduct research and pursue a PhD within the VANDAL laboratory. Being part of this research group gave me the chance to work and exchange ideas with outstanding researchers.

Among the people I had the opportunity to work with, I would like to thank Prof. Giuseppe Averta for his advice and guidance, and for giving me the opportunity to gain first-hand experience in research areas that I would never have expected to explore.

My sincere thanks also go to Prof. Jose Cano Reyes and Prof. Manuel Roveri for their careful evaluation of this thesis and for their valuable comments and suggestions.

My academic journey would not have been the same without all the people it allowed me to meet and who contributed to my growth as a researcher, but most importantly as a person. For this reason, I want to thank my lab colleagues and friends Andrea, Antonio A., Antonio T., Claudia, Debora, Gabriele×5, Leonardo, Mirco, Niccolò, Paolo, Riccardo, Silvia, Simone and all the others.

I believe that an academic and scientific journey is, first and foremost, a human one, and as such it cannot be separated from the people who became part of it, both within and outside the research environment. This is especially true when, as it is often the case, such journeys are not free from difficulties. In this respect, I feel it is particularly important to acknowledge the people without whom these obstacles would have been harder to overcome. For this reason, although a complete list would be impossible for reasons of space, I would like to explicitly mention and thank Andrea, Elia, Ilaria, Silvia, Teresa, Diana, Francesca, Giovanni, Sonia and Beatrice. Each of you, in your own ways, helped support and encourage me along the way.

Last, but certainly not least, I would like to thank my family, whose unconditional support has been invaluable since the very first chapter of my studies.

Summary

Deep learning has become a central technology in modern artificial intelligence, enabling substantial progress in perception, representation learning, and decision-making. Its impact is particularly evident in computer vision, where neural networks have achieved strong performance across recognition and large-scale visual understanding tasks. At the same time, the design and deployment of these models increasingly involve constraints that go beyond predictive accuracy alone. Models may need to operate under limited computational budgets, adapt to settings where annotations are few or unavailable, and be evaluated under conditions that better reflect real deployment scenarios. Neural Architecture Search (NAS) provides a systematic framework to automate architecture design, but its practical adoption is still limited by the cost of candidate evaluation, assumptions on the availability of validation labels and simplified benchmarking protocols.

This thesis investigates practical Neural Architecture Search for computer vision, focusing on efficient search methods and realistic evaluation. Rather than treating NAS as a generic optimization procedure, it studies how architecture search can be adapted to settings in which optimization cannot rely directly on the final validation objective, either because target labels are unavailable or because candidate training must be avoided, and in which evaluation must account for deployment-relevant criteria beyond in-domain predictive accuracy. This perspective is developed along two main directions: first, the design of search strategies that remain effective under practical constraints, such as limited target supervision and limited computational budgets; second, the construction of evaluation settings that better reflect modern architectures, large-scale data, and deployment-relevant trade-offs.

The first part of the thesis addresses NAS in the context of Unsupervised Domain Adaptation (UDA). In this setting, labeled data are available only from a source domain, while labels from the target domain cannot be used during training or model selection. Moreover, many effective UDA pipelines rely on pre-trained backbones, making full-network search undesirable in practical scenarios. To address these limitations, the thesis introduces Adversarial Branch Architecture

Search (ABAS), a framework that restricts the search to the auxiliary adversarial branch attached to a pretrained backbone. ABAS combines multi-fidelity Bayesian optimization with an ensemble-based model selection strategy that estimates target-domain performance from complementary unsupervised signals, enabling architecture search without target supervision.

The second part focuses on reducing the computational cost of NAS through training-free evaluation. Instead of training each candidate architecture, the proposed methods rely on proxy metrics computed at initialization. FreeREA combines complementary zero-cost indicators within a modified evolutionary algorithm, showing that training-free evaluation can guide the search toward competitive architectures under both unconstrained and resource-constrained settings. This direction is further extended with the introduction of Entropic Score, a data-agnostic metric designed to capture topology-related properties of candidate networks. By decoupling the optimization of topology and model size, the proposed strategy assigns dedicated proxy signals to different architectural dimensions, showing that efficient NAS can benefit from a structured use of training-free estimates.

The final part addresses the evaluation of NAS methods. Many existing benchmarks rely on small datasets and predominantly convolutional search spaces, which only partially reflect modern vision architectures and deployment constraints. To overcome these limitations, the thesis introduces HyViTas-Bench, a benchmark for hybrid CNN-ViT architectures trained on a reduced, yet large-scale, ImageNet-based dataset. The benchmark includes repeated training runs, latency measurements across heterogeneous hardware platforms, out-of-distribution evaluations, and precomputed training-free proxies. Its analysis highlights that architectural quality is multidimensional, since in-distribution accuracy, robustness, latency and proxy-based rankings do not always align.

Overall, this thesis contributes methods and evaluation tools for Neural Architecture Search under realistic constraints, highlighting the need for efficient search procedures, appropriate performance estimators, architecture-aware proxy metrics, and benchmarks capable of exposing practical trade-offs.

Keywords: deep learning, computer vision, automl, neural architecture search, efficient neural networks

Contents

List of Tables	XI
List of Figures	XV
1 Introduction	1
1.1 Contributions	3
1.2 Outline	4
1.3 Publications	5
2 Background and Related Works	7
2.1 Problem setting and formulation	7
2.1.1 Neural Architecture Search as an optimization problem . . .	7
2.1.2 From the true objective to estimators	8
2.1.3 Practical NAS scenarios	8
2.1.4 Evaluation dimensions and sources of mismatch	9
2.2 Neural Architecture Search: method families	10
2.2.1 Search spaces	10
2.2.2 Search strategies	12
2.2.3 Architecture evaluation and efficient performance estimation	17
2.2.4 Benchmark-based NAS evaluation	19
2.3 Architectures for Computer Vision	21

2.3.1	Convolutional architectures and residual design	21
2.3.2	Vision Transformers	23
2.3.3	Efficient and hybrid vision architectures	25
3	NAS for Unsupervised Domain Adaptation	27
3.1	Introduction	27
3.2	Related works	29
3.2.1	Unsupervised Domain Adaptation (UDA)	29
3.2.2	Model Selection for UDA	29
3.3	Method	30
3.3.1	Background	30
3.3.2	Search space	32
3.3.3	Ensemble Model Selection (EMS)	33
3.3.4	Adversarial branch optimization	37
3.4	Experiments	38
3.4.1	Datasets	38
3.4.2	Training protocol	38
3.4.3	Search space sensitivity	40
3.4.4	Performance estimation	40
3.4.5	Comparison with baseline methods	41
3.5	Conclusions	46
4	Training-free NAS for tiny and hybrid architectures	47
4.1	Introduction and motivation	47
4.2	Related works	49
4.3	FreeREA: combining training-free proxies for evolutionary NAS . . .	50

4.3.1	Metrics	51
4.3.2	Search strategy	55
4.3.3	Experiments	56
4.4	Decoupling topology and size in training-free NAS	62
4.4.1	Method	64
4.4.2	Experiments	69
4.4.3	Ablation study	71
4.5	Conclusions	75
5	A hybrid CNN-ViT architecture search benchmark	77
5.1	Introduction	77
5.2	Related works	79
5.3	HyViTas-Bench	81
5.3.1	Search space	81
5.3.2	Datasets	82
5.3.3	Training and evaluation protocol	85
5.4	Benchmark evaluation	86
5.4.1	Baseline experiments	89
5.4.2	Accuracy and latency across hardware platforms	100
5.5	Conclusions	102
6	Conclusions	105
6.1	Summary of contributions	105
6.2	Limitations and future work	108
	Bibliography	111

List of Tables

3.1	Rank correlation between different metrics and target accuracy. EMS is evaluated in a cross-dataset setting, where the linear regressor is trained on a dataset different from the one reported in the table, and consistently outperforms the other individual performance estimators.	36
3.2	A set of configurations was randomly sampled from the ABAS search space and trained with ALDA on Office31 [106] (reported in the first six rows), and Office-Home [129] settings. The results highlight the strong impact of the auxiliary branch configuration on final performance. Div. denotes the fraction of runs that did not converge, Min and Max report the minimum and maximum accuracy, Mean gives the average accuracy with standard deviation, and Range is computed as Max - Min.	43
3.3	ABAS results on Office31 using a ResNet-50 backbone. “+ EMS” denotes experiments evaluated with the proposed model selection strategy. For the baselines and their EMS variants, we report the mean and standard deviation over four runs.	44
3.4	ABAS results on Office-Home with a ResNet-50 backbone. Official ALDA results could not be reproduced with the released code. “+ EMS” denotes runs evaluated with the proposed model selection strategy.	44
3.5	ABAS results on PACS with a ResNet-50 backbone, “+ EMS” denotes runs evaluated with the proposed model selection strategy.	45
4.1	Kendall- τ and Spearman- ρ correlations between training-free metrics and test accuracy on the three datasets considered in NATS-Bench [26]. Each metric is evaluated over three different initializations, and the resulting scores are averaged before computing the correlations.	51

4.2	Ablation study of the metrics included in the fitness function. The table reports the test accuracy of the final selected model. Each configuration is evaluated over 30 runs, with a search budget of 45 seconds on NATS-Bench and 12 minutes on NAS-Bench-101. The two metrics introduced in this work, Skipped Layers and LogSynflow, appear to provide the strongest contribution, while NASWOT has a noticeable effect mainly on NAS-Bench-101, likely because its discriminative ability is more effective in larger search spaces.	52
4.3	Kendall- τ and Spearman- ρ correlations between Synflow, LogSynflow, and test accuracy on NAS-Bench-101 [143]. The values show that LogSynflow provides a more reliable proxy for test accuracy than the original Synflow metric.	54
4.4	Test accuracy and search time, measured in seconds, on NATS-Bench. Results for training-based baselines are reported from [26]. .	60
4.5	Ablation study on NATS-Bench [26] for different values of N and n . The table reports the test accuracy of the final selected model. Each configuration is evaluated over 30 runs with a search budget of 45 seconds.	60
4.6	Test accuracy and search time, measured in seconds, for training-free methods and REA on NAS-Bench-101 [143].	61
4.7	FreeREA test accuracy on NATS-Bench under constrained search settings. Regret represents the gap between the optimal and average test accuracy obtained during search. Each run is stopped after 45 seconds.	63
4.8	Division of searchable dimensions between topology and size. Following the convention adopted in NATS-Bench [26], kernel size is considered a topological dimension rather than a size-related one, as it characterizes the operator choice rather than its dimensionality. .	69
4.9	Hyper-parameters used to train the models on ImageNet-1k.	72
4.10	ImageNet-1k classification results. Models are evaluated at the standard 224×224 resolution, except for MobileViTs, which use 256×256 inputs. $\diamond$ denotes training with distillation, $\dagger$ indicates training with the original configuration without distillation, and $\ddagger$ marks conservatively estimated search times when the original papers do not report them. Search time is reported in GPU hours/days. The symbols $\uparrow$ and $\downarrow$ indicate whether higher or lower values are better.	73

4.11	Kendall and Spearman rank-correlation values between training-free scores and CIFAR-10 top-1 accuracy, computed a) on the topological search space of NAS-Bench-101 and b) on the size search space of NATS-Bench. Entropic Score is mainly effective for topology-related choices, while LogSynflow better captures variations in model dimension.	74
4.12	Ablation study of the proposed search strategy on the extended search space, where Residual Bottleneck blocks are included as additional architectural choices. ImageNet-1k top-1 accuracy is reported. The symbols $\downarrow$ and $\uparrow$ indicate whether lower or higher values are better, respectively.	74
5.1	Summary of representative NAS benchmarks. HyViTas-Bench is the first benchmark in this set to address hybrid CNN-ViT architectures on large-scale data, while also providing out-of-distribution accuracy, hardware-related measurements, and precomputed training-free proxies within a unified evaluation framework.	81
5.2	Searchable dimensions of the HyViTas-Bench search space. Kernel size and expansion factor are optimized independently for the first and last pairs of stages, yielding a search space containing 6,561 different architectures.	83
5.3	Kendall- τ and Spearman- ρ correlations of architecture rankings between different datasets and ImageNet-1k, computed across 20 randomly sampled architectures.	89
5.4	Top-1 in-distribution and out-of-distribution accuracies, reported as mean $\pm$ standard deviation, for Random Search, Regularized Evolution (REA), REINFORCE, and BANANAS over 500 independent trials and under different optimization objectives. Accuracy-driven searches optimize either the in-distribution validation accuracy alone or a joint “ID+OoD” criterion, computed as the average between the ID accuracy and the mean accuracy on four OoD evaluation sets: RedImageNet-C at corruption levels 1 and 5, RedImageNet-Sketch, and RedStylized-ImageNet. For zero-shot NAS, the optimization objective is instead given by standard training-free proxies. The table also includes the one-shot methods DARTS and SPOS, evaluated over five runs with ID accuracy as target. The “Optimum” row reports, for each dataset, the best architecture available in the benchmark, while “Search space mean” summarizes the average accuracy and standard deviation over the entire search space.	94

5.5	Kendall- τ and Spearman- ρ rank correlations between several training-free proxies and the final top-1 accuracy, evaluated both in-distribution on RedImageNet validation and under out-of-distribution shifts, including RedImageNet-C across all corruption severities, RedImageNet-Sketch, and RedStylized-ImageNet. NASWOT shows the strongest correlation with in-distribution accuracy, but it decreases under high-severity corruptions (RedImageNet-C 5), while remaining comparatively informative for domain-shifted data such as RedImageNet-Sketch.	95
5.6	Mean Kendall- τ and Spearman- ρ correlations between Entropic Score and final accuracy across subspaces obtained by varying one family of architectural choices at a time while fixing the others. Standard deviations across subspaces are reported together with the mean values, along with the percentage of positive correlations.	97
5.7	Inference latency measured on a representative subset of the hardware platforms included in HyViTas-Bench. For each device, latency is reported in milliseconds as the mean, standard deviation, minimum, and maximum across all architectures in the search space. The selected devices cover the main hardware classes considered in the benchmark and highlight that single-image inference does not always follow hardware tier, with the RTX 5090 achieving lower latency than the H100.	100

List of Figures

3.1	General framework shared by several UDA methods. The auxiliary branch D is attached to the feature extractor and can be used either to implement a domain discriminator [15], [33] or to compute a self-supervised auxiliary loss [9], [119].	31
3.2	Integration of the ABAS search space into the main UDA architecture. The searchable auxiliary branch is attached after the feature extractor bottleneck and consists of a variable number of fully connected layers followed by the final domain classifier.	32
3.3	Model configurations trained on Office31 (left), and Office-Home (right). The two axes correspond to different unsupervised validation metrics used by EMS, while color encodes target-domain accuracy. A single metric alone does not provide a reliable prediction of model performance, while combining multiple metrics helps separate well-performing and poorly performing configurations. This highlights the complementarity of the validation metrics and their usefulness for estimating model quality. The similarity between corresponding plot pairs (first and second columns) suggests that an estimator learned on one dataset can transfer to another.	35
3.4	Accuracy distributions obtained by training 40 randomly sampled architectures for the ALDA auxiliary branch on three representative settings, excluding the 10% of runs that did not converge. Compared with many search spaces [141], the proposed one is sufficiently broad for architectural choices to produce a substantial variation in final performance, around 20% across all settings. The default ALDA branch performs well on Office31 and Office-Home, where it was originally evaluated, but degrades on PACS, while ABAS adapts to the different settings and consistently outperforms random sampling.	41

3.5	Sensitivity analysis of the ABAS search space. For each dataset, 40 configurations are randomly sampled and trained on one representative setting. The results show that the best hyperparameter values vary substantially across tasks. In each plot, the x-axis corresponds to the search-space hyperparameter under analysis.	42
4.1	Relationship between test accuracy and the two training-free metrics used in the method, computed over all architectures in NATS-Bench: NASWOT on the left panel and LogSynflow on the right. Accuracy values refer to the ImageNet-16-120 dataset.	53
4.2	Mean test accuracy and corresponding standard deviation on the ImageNet-16-120 dataset of NATS-Bench. The shaded regions group training-based methods and training-free methods separately.	57
4.3	Average test accuracy versus time on NATS-Bench (ImageNet-16-120). The time axis is reported in logarithmic scale. For training-based methods, both accuracy and runtime values are taken from the original NATS-Bench paper [26].	59
4.4	Evolution of test accuracy over search time for REA, FreeREA ⁻ , and FreeREA on NAS-Bench-101. The time axis is shown in logarithmic scale, and standard deviations are reduced by a factor of 10 for readability. FreeREA exhibits higher variability during the early search phase, between 10^1 and 10^2 seconds, but converges to lower uncertainty near the end of the search, while REA and FreeREA ⁻ maintain comparable variance during the process.	62
4.5	Alternative locally-enhanced FFN block designs. The left panel shows an Inverted Bottleneck [107] block, while the right panel illustrates a ConvNeXt [73] block.	64
4.6	Relationship between training-free metrics and CIFAR-10 test accuracy. The left plot reports Entropic Score on the topological search space of NAS-Bench-101 [143], while the right plot shows LogSynflow on the dimensional search space of NATS-Bench [26]. Entropic Score is particularly effective for topology selection, while LogSynflow is better suited to architectural dimensioning.	65
4.7	Comparison between model size and ImageNet-1k top-1 accuracy for state-of-the-art NAS approaches. Marker size encodes the computational cost of each model in terms of MACs.	70

5.1	Schematic representation of the hybrid CNN-ViT backbone underlying the HyViTas-Bench search space. The network is organized into four stages and includes a fixed Stem module inspired by EdgeNeXt [82], two searchable convolutional stages built from ConvNeXt [73] blocks, two searchable transformer-based stages derived from LocalViT [67], and a fixed classification head. The search dimensions comprise kernel size, the expansion factor shared across stages of the same type, and the output depth, which is optimized independently for each stage. The structure of the encoder blocks is shown in Figure 5.2.	82
5.2	Structure of the transformer-based encoder block used in the attention-driven stages of the search space. The block consists of an attention module followed by a ConvNeXt [73] block, with residual connections around both components. The inset on the right details the ConvNeXt sub-block, which adopts an inverted bottleneck structure and includes a depthwise convolution.	83
5.3	Relationship between model size and final validation accuracy for the architectures in the benchmark, with accuracy averaged over three training runs. Architectures with accuracy below 50% are omitted for visualization purposes, as they account for less than 1% of the search space. The highlighted curve denotes the Pareto front, while the star marker (★) identifies the best-performing architecture in the search space.	87
5.4	Distribution of the final validation top-1 accuracy across the architectures in the benchmark, reported as the average over three training runs, together with the corresponding standard deviation across runs. The left plot summarizes average performance, while the right plot shows sensitivity to random initialization, with lower standard deviations indicating more stable architectures. A limited number of outliers is omitted for visualization clarity.	88

5.5	Relationship between in-distribution accuracy on RedImageNet and out-of-distribution accuracy on reduced versions of ImageNet-Sketch, Stylized-ImageNet and ImageNet-C at corruption severities 1 and 5. For RedImageNet-C, each accuracy value is averaged over the 15 corruption types. Star markers (★) denote the best-performing architectures for each OoD dataset, showing that the OoD optima generally differ from the architecture with the highest ID accuracy. The Pareto front is reported for each OoD setting. Kendall- τ correlations between ID and OoD accuracy are 0.68 for RedImageNet-Sketch, 0.25 for RedStylized-ImageNet, 0.57 for RedImageNet-C level 1, and 0.09 for RedImageNet-C level 5.	92
5.6	Distribution of Kendall- τ rank correlations between Entropic Score and final accuracy across the subspaces obtained by fixing all architectural choices except the considered family. Lighter bars highlight positive correlation values, while the red dashed line denotes the mean correlation for each distribution. Kernel-size variations mostly exhibit positive correlations, whereas the other architectural choices are shifted towards negative correlation values.	98
5.7	Scatter plots relating six training-free metrics to the corresponding top-1 accuracy of the architectures included in HyViTas-Bench, considering both in-distribution and out-of-distribution evaluations. Each point represents one architecture under a specific evaluation setting, while colors distinguish the in-distribution dataset from the considered OoD benchmarks.	99
5.8	Relationship between in-distribution top-1 accuracy and inference latency across representative hardware platforms. Each point corresponds to one architecture in the search space, while the highlighted curve denotes the Pareto front. The plots show that different platforms induce different latency distributions and Pareto-optimal sets.	101

Chapter 1

Introduction

Over the past decade, Artificial Intelligence (AI) has first attracted progressively growing attention and then experienced an unprecedented surge, with AI models becoming increasingly pervasive. Historically, neural networks research has largely prioritized task performance, devoting comparatively less effort to efficiency and deployability [4], [108]. Yet real-world applications impose constraints, often in terms of computational budget and memory footprint [70], but also in terms of the amount, quality and accessibility of data [146].

These constraints are becoming ever more pressing as modern models scale to hundreds of billions or even trillions of parameters [32], [130]. While such large models are typically hosted in large-scale datacenters, where compute availability and energy supply may appear less limiting, this centralization is not a definitive solution. Rather than eliminating resource constraints, it shifts them to the data-center level: the resources required to train and run these models are increasingly challenging even in that setting [50], [92]. Energy consumption can be enormous, hardware cooling demands substantial water [65], and the production and supply of specialized hardware are struggling to keep pace with the growth of datacenter infrastructure [89].

Moreover, offloading computation to remote systems is not always viable nor desirable. Remote execution introduces latency that may be unacceptable for time-sensitive applications [14], [83], and it transfers control over both computation and data to third parties that may not be fully trusted [14], [110]. In many scenarios, deploying AI locally is therefore preferable and, even when remote computation is feasible, the underlying resources should not be treated as effectively infinite.

Traditionally, the development of new machine learning models has relied heavily on an iterative trial-and-error process that demands substantial time, expertise

and computational resources. This reliance initially encouraged the adoption of standardized, well-established architectures, which offered strong performance and predictable training behavior across a range of tasks. As the demand grew for models tailored to specific data modalities, problem settings and deployment constraints, the limitations of manual design became increasingly evident. This shift motivated the development of automated approaches for optimizing the machine learning pipeline [44]. In particular, the architecture-design stage began to be addressed through algorithmic procedures, giving rise to the field of Neural Architecture Search (NAS) [153], which aims to automatically discover neural network architectures that satisfy task objectives and practical constraints.

At a high level, NAS formulates architecture design as an optimization problem. Candidate networks are drawn from a search space, typically defined by a set of building blocks (i.e., operators) and composition rules that describe the family of architectures of interest. A search strategy then explores this space in order to identify high-performing designs, guided by an evaluation criterion that reflects the target objective, such as predictive accuracy, possibly combined with additional terms accounting for latency, memory footprint or other deployment-related requirements.

This thesis investigates the automation of neural network design with the goal of explicitly accounting for practical constraints, including the deployment target, available computational budget, and data availability. In this context, it is important to note that NAS has historically been associated with substantial computational cost, largely because candidate architectures must be evaluated repeatedly during search. Although NAS can identify architectures that are efficient at inference time, its benefits may be undermined if achieving them requires training and evaluating thousands of candidate networks.

To address this limitation, the community has developed one-shot [40], [69] and zero-shot (often referred to as training-free) [1], [85] NAS paradigms. One-shot approaches approximate the search by training a single over-parameterized supernet, while zero-shot methods aim to rank or select architectures by using heuristic performance metrics, without training candidate networks at all. Since search efficiency is itself a practical constraint, a significant part of this thesis focuses on training-free NAS paradigms that enable architecture optimization under tight computational budgets.

1.1 Contributions

This thesis focuses on the development of practical and computationally efficient methodologies for automating deep learning model design via Neural Architecture Search (NAS), addressing key challenges related to supervision availability, computational efficiency, and realistic evaluation settings.

In the first part of this thesis, we introduce a NAS framework for Unsupervised Domain Adaptation (UDA). UDA addresses the problem of learning models that generalize to a target data distribution for which labeled samples are not available at training time, while labeled data are accessible from a related but distinct source distribution. The objective of UDA is to mitigate the distributional shift between source and target domains, enabling effective knowledge transfer without target supervision. Several UDA approaches rely on auxiliary architectural branches, such as domain discriminators or self-supervised pretext-task modules, which are integrated into the network to promote domain-invariant feature representations. The architecture of such auxiliary branches is often designed in a heuristic or simplistic manner. While these designs can be effective in practice, they are not guaranteed to be optimal. A natural objective is therefore to automatically optimize the architecture of the auxiliary branch. However, this poses a fundamental challenge: during training, and consequently during architecture optimization, the model performance on the target domain is unknown, as target labels are not available. As a result, target-domain accuracy cannot be directly used as an optimization signal to guide the architecture search. To address this issue, the proposed framework introduces a cross-domain performance predictor, which estimates the model effectiveness on target data and enables architecture optimization in the absence of target supervision.

The second part of the thesis advances zero-shot NAS by introducing evolution-based strategies that leverage combinations of training-free metrics as proxies for final model accuracy. These approaches enable fast and resource-efficient architecture search, requiring little computational power, while still achieving performance competitive with, and in some cases superior to, training-based NAS baseline methods. Building on this perspective, we explore how different training-free metrics capture complementary aspects of network quality, such as trainability and expressivity. These metrics are combined and embedded within evolutionary search algorithms to efficiently guide the exploration of the architecture space. Overall, this part of the thesis demonstrates that carefully designed training-free evaluation strategies, coupled with tailored evolutionary search mechanisms, can effectively replace costly training-based NAS pipelines, enabling fast, scalable and practically deployable architecture search under strict computational constraints.

The final part of the thesis addresses the challenge of realistic and reproducible evaluation of NAS methods. While substantial progress has been made in reducing the computational cost of NAS through one-shot and zero-shot approaches, the evaluation of these techniques, and of NAS methods in general, is still predominantly conducted on benchmarks based on small-scale datasets and search spaces that are largely limited to purely convolutional networks. To overcome these limitations, this thesis introduces a benchmark specifically designed for hybrid convolutional and transformer-based architectures and based on a large-scale dataset. The proposed benchmark provides a comprehensive evaluation framework for both training-based and training-free NAS methods on a large-scale dataset. In addition, it enables the systematic analysis of out-of-distribution generalization across different architectures, and it includes hardware latency measurements to assess inference efficiency on real computing platforms.

1.2 Outline

The remainder of the thesis is structured as follows. In Chapter 2, we present an overview of the fundamental concepts in deep learning for computer vision, together with an introduction to Neural Architecture Search. More specifically, Section 2.2 reviews the main classes of Neural Architecture Search methods, and Section 2.3 provides an overview of the key architectural patterns adopted throughout the thesis.

Chapter 3 presents an application of Neural Architecture Search to the Unsupervised Domain Adaptation (UDA) setting, investigating the automated optimization of auxiliary branch architectures commonly employed in domain adaptation methods. In particular, the chapter introduces the Adversarial Branch Architecture Search (ABAS) framework and Ensemble Model Selection (EMS), which enable architecture optimization in the absence of target-domain labels.

Chapter 4 focuses on efficient Neural Architecture Search through training-free metrics. In Section 4.3, we introduce FreeREA, a novel evolutionary method that explores the search space without training candidate networks, leveraging the training-free performance proxy LogSynflow. In Section 4.4, we extend the discussion to transformer-based architectures, showing how the search over model size and topology can be decoupled, and we propose Entropic Score as a dedicated metric for topological spaces.

In Chapter 5 we address the problem of benchmarking NAS methods, with a particular focus on hybrid architectures that combine convolutional networks and vision transformers, as well as on the representativeness of the evaluation data. We

introduce HyViTas-Bench, to the best of our knowledge the first NAS benchmark based on a hybrid CNN-ViT search space. We show that architectures achieving the highest in-distribution performance do not necessarily exhibit the best generalization capability to shifted data distributions. Hence, we include Out-of-Distribution evaluation metrics as part of the benchmark. Finally, to support deployability-oriented studies, HyViTas-Bench provides latency measurements across a broad range of hardware platforms.

1.3 Publications

The following list provides a comprehensive chronological list of the author’s publications. Please note that only the papers marked with a $\star$ are included in this thesis.

- M. R. Loghmani, L. Robbiano, M. Planamente, K. Park, B. Caputo, and M. Vincze, **Unsupervised domain adaptation through inter-modal rotation for RGB-D object recognition**, *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6631-6638, 2020. Presented at IROS 2020 [74]
- L. Robbiano, M. R. Ur Rahman, F. Galasso, B. Caputo, and F. M. Carlucci, **Adversarial branch architecture search for unsupervised domain adaptation**, in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 1008-1018 [103] $\star$
- N. Cavagnero, L. Robbiano, B. Caputo, and G. Averta, **FreeREA: Training-free evolution-based architecture search**, in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 1493-1502 [11] $\star$
- N. Cavagnero, L. Robbiano, F. Pistilli, B. Caputo, and G. Averta, **Entropic score metric: Decoupling topology and size in training-free nas**, in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2023, pp. 1459-1468 [12] $\star$
- L. Robbiano, F. Pistilli, and G. Averta, **A CNN-ViT hybrid architecture search benchmark on a large-scale dataset**, *IEEE Access*, vol. 13, pp. 209 965–209 979, 2025 [102] $\star$

Chapter 2

Background and Related Works

2.1 Problem setting and formulation

2.1.1 Neural Architecture Search as an optimization problem

Neural Architecture Search (NAS) aims to automate the design of neural network architectures by framing architecture design as an optimization problem over a predefined search space. Let $\mathcal{A}$ denote the search space, that is, the set of candidate architectures considered during the search. Each candidate architecture $\Theta \in \mathcal{A}$ is evaluated by an objective function $J(\Theta)$, which quantifies its quality according to the target criterion of interest, such as predictive performance on a validation set. The NAS problem can then be formalized as the search for the architecture that maximizes such an objective:

$$\Theta^* = \arg \max_{\Theta \in \mathcal{A}} J(\Theta) \quad (2.1)$$

In this formulation, $\mathcal{A}$ may be a fully discrete space, for example when it is defined by a finite set of architectural choices, or a hybrid space including both discrete design variables and continuous hyperparameters. More generally, the definition of $\mathcal{A}$ determines which aspects of the architecture can be optimized, such as the network topology, the depth and width of its stages, the type of computational blocks, or their internal configuration.

This formulation highlights the main conceptual components of a NAS problem. First, the search space specifies the set of admissible architectures. Second, the objective function defines the property to be optimized. Third, a search procedure

is required to explore or optimize over $\mathcal{A}$, since exhaustive evaluation is generally infeasible except for very small spaces. Different NAS methods implement this search procedure in different ways. Some approaches explicitly compare individual candidate architectures, whereas others optimize a relaxed or over-parameterized representation of the space and derive the final architecture only after this optimization process. Despite these differences, they can all be interpreted within the same general formulation given above.

Under this perspective, NAS should be regarded as a general optimization framework rather than a specific algorithmic pipeline. What changes across methods is not the underlying problem definition, but the way in which the search space is parameterized, the optimization is carried out, and candidate architectures are evaluated. In this thesis, such a framework is considered primarily in the context of computer vision, with a predominant focus on image classification, although the same formulation naturally extends to broader application domains.

2.1.2 From the true objective to estimators

While the NAS problem can be expressed in terms of an objective function $J(\Theta)$, directly computing such an objective is often impractical. Typically, assessing the actual quality of an architecture would require a complete training and evaluation pipeline, whose cost may be prohibitive during search. As a consequence, NAS methods often rely on an alternative evaluation signal $\tilde{J}(\Theta)$, used as a proxy for the target objective:

$$\Theta^* = \arg \max_{\Theta \in \mathcal{A}} \tilde{J}(\Theta), \quad \tilde{J}(\Theta) \approx J(\Theta). \quad (2.2)$$

The distinction between $J(\Theta)$ and $\tilde{J}(\Theta)$ is central in NAS. The former represents the true quantity of interest, whereas the latter denotes the signal that is effectively accessible and used during optimization. Depending on the method and on the experimental setting, $\tilde{J}(\Theta)$ may correspond to a partial training evaluation, a weight-sharing estimate, a predictor-based surrogate or a training-free metric computed at initialization.

2.1.3 Practical NAS scenarios

The formulation introduced above naturally covers different NAS settings, depending on the nature of the optimization objective and on the presence of additional constraints. In the simplest case, NAS can be posed as an unconstrained optimization problem, where the goal is to identify the architecture that maximizes

the target objective within the search space as defined in (2.1). In this setting, the search is driven exclusively by the performance criterion of interest, typically predictive accuracy on a validation set, without imposing explicit limitations on the computational or structural properties of the resulting model. This formulation is common in many benchmark studies, where the primary goal is to assess the effectiveness of a search strategy in identifying high-performing architectures.

A more realistic formulation is given by constrained NAS, in which the search must satisfy one or more resource-related requirements in addition to maximizing performance. In this case, the problem can be expressed as

$$\Theta^* = \arg \max_{\Theta \in \mathcal{A}} J(\Theta) \quad \text{s.t.} \quad C_i(\Theta) \leq b_i, \quad i = 0, 1, \dots, m, \quad (2.3)$$

where $C_i(\Theta)$ denotes the i -th cost associated with architecture Θ , and b_i is the corresponding admissible budget. Depending on the target application, such constraints may involve the number of parameters, floating-point operations, memory usage, inference latency, or energy consumption. Constrained NAS is particularly relevant when architectures are intended for resource-limited environments, where predictive performance alone is not sufficient to determine the suitability of a model.

Beyond this distinction, many applications lead to practical NAS settings in which the search problem is affected by additional limitations that are not fully captured by standard resource constraints. In such scenarios, the challenge may stem from the lack of complete supervision, from the need to preserve pretrained components, from the high cost of architecture evaluation, or from the fact that the benchmark objective only partially reflects the conditions of the final deployment setting. As a consequence, NAS should not be interpreted only as the optimization of predictive performance under optional resource budgets, but more generally as the design of architectures under the constraints and approximations imposed by real application contexts. This broader perspective is particularly relevant for the contributions presented in this thesis, which address NAS under limited supervision, under limited computational budgets, and under realistic evaluation conditions.

2.1.4 Evaluation dimensions and sources of mismatch

The quality of a neural architecture is inherently multidimensional. In the most common formulation of NAS, the target objective is defined in terms of in-distribution predictive performance, such as validation accuracy in image classification. However, this criterion captures only one aspect of architectural quality. Depending on the target application, additional dimensions of interest may also be optimized, including performance under domain shift, computational and memory

efficiency, inference latency, and robustness beyond the training distribution. More generally, an architecture cannot be regarded as good in an absolute sense, but only with respect to a given set of evaluation dimensions.

In principle, NAS could be formulated by jointly accounting for several such dimensions. In practice, however, many methods reduce the problem to a scalar objective, or to a constrained optimization setting in which one target metric is maximized under resource budgets. A related line of research addresses this issue more explicitly through multi-objective NAS, where multiple criteria are optimized simultaneously. Even in this case, however, the relevance and interpretation of each evaluation dimension remain strongly dependent on the final application context.

A further complication is that the quantities used to evaluate architectures during search or benchmarking do not always coincide with the quantities that ultimately matter at deployment time. This gives rise to several forms of mismatch. A first source is objective mismatch, which emerges when the score used during search is only a proxy for the final trained performance, or when validation performance does not fully reflect deployment behavior. A second source is data mismatch, which arises when architectures are assessed on reduced-scale datasets rather than on the intended full-scale setting, or when in-distribution evaluation is used as a surrogate for robustness under distribution shift. Finally, the adopted evaluation protocol may itself introduce distortions, since simplified or partial procedures can affect the conclusions drawn about the relative quality of candidate architectures. For these reasons, evaluating architectures in NAS should not be interpreted as the measurement of a single universal and well-defined property, but rather as the approximation of a broader and application-dependent notion of quality.

In this thesis, these aspects are addressed from complementary perspectives. While multi-objective NAS is not treated explicitly, the discussion of different evaluation dimensions, including in-distribution performance, out-of-distribution behavior, and efficiency, is taken up more directly in the final part of the thesis, where the problem of realistic benchmarking is addressed (Chapter 5).

2.2 Neural Architecture Search: method families

2.2.1 Search spaces

The search space defines the set of candidate architectures that a Neural Architecture Search algorithm is allowed to explore. Its design is a central aspect of NAS, as it determines both the expressive power of the search process and its

computational cost, along with the difficulty of optimization and the extent to which the discovered architectures can generalize beyond a specific benchmark. In practice, NAS methods do not explore the space of all conceivable neural networks. Rather, they operate on constrained design spaces, in which only specific architectural choices are left variable while the remaining aspects are fixed a priori.

A common distinction is between cell-based and stage-wise search spaces. In cell-based NAS, the search is restricted to a small computational cell, which is then repeated multiple times to form the final, complete architecture. This design greatly reduces the complexity of the search and has historically enabled efficient comparison across methods on standard benchmarks. However, the resulting flexibility is limited, since the global structure of the network is largely predetermined and architectural diversity is concentrated within the repeated cell. By contrast, stage-wise search spaces allow the architecture to vary at the level of broader network stages, typically enabling different configurations across depth, resolution levels, or groups of blocks. This setting is more expressive and better suited to modern architectures, where different stages may play distinct representational roles, but it also leads to a substantially larger and more complex optimization problem.

An orthogonal distinction concerns the type of architectural decisions represented in the search space. Some spaces primarily encode topological choices, such as the presence of connections, the arrangement of blocks, or the selection of operations and pathways. Others emphasize dimensional choices, including network depth, channel width, expansion ratios, hidden dimensions, or the size of attention-related components. Although these two aspects are often entangled in practical search spaces, separating them conceptually is useful, since topology and dimensionality affect model behavior in different ways and may benefit from different search or evaluation strategies. This distinction is particularly relevant in the context of this thesis, with the work presented in Section 4.4 explicitly exploiting the idea that different zero-cost metrics may be more informative for topology than for model size.

It must be noted that the search space does not necessarily need to cover the entire architecture. In some scenarios, it may be preferable to preserve part of a network, for instance a pretrained backbone, and restrict the search to selected architectural components. This choice reduces the complexity of the problem and may better reflect practical settings in which only specific modules are architecture-sensitive.

The specific search spaces adopted in the remainder of this thesis are described in the corresponding chapters, while the present section is limited to the main conceptual distinctions needed to frame them.

2.2.2 Search strategies

Given a search space, the search strategy defines how candidate architectures are generated, explored, and refined during the optimization process. Different strategies induce different trade-offs between exploration efficiency, computational cost, and the ability to handle complex or high-dimensional search spaces. Over the years, NAS research has explored a broad range of search strategies, from simple uninformed baselines to highly structured optimization procedures. The most representative method families are described in the following paragraphs.

Random search

Random search represents the simplest possible NAS strategy, as candidate architectures are sampled from the search space without exploiting any explicit model of past observations. Despite its simplicity, it plays an important role as a reference baseline, since it provides a direct way to assess how much a more sophisticated search strategy actually contributes beyond the design of the search space itself. In sufficiently constrained spaces, random search can already yield competitive results, which has made it a common point of comparison in the NAS literature [63]. Representative examples include methods such as NASWOT [85], where the focus is placed more on the architecture ranking criterion than on the search policy itself.

Reinforcement learning-based methods

Reinforcement learning was among the earliest successful strategies for NAS. In this setting, architecture design is formulated as a sequential decision process in which a controller selects operations or connections and receives a reward based on the performance of the resulting architecture. This formulation is highly flexible and can naturally accommodate complex design spaces, but it is often associated with high computational cost, since many candidate networks must be evaluated before the controller becomes effective. Seminal examples include [153], which established NAS as a practical research direction, and later cell-based approaches such as NASNet [154], where reinforcement learning was used to discover reusable architectural patterns.

One-shot and differentiable methods

A major shift in NAS was introduced by one-shot and differentiable methods, which were proposed to reduce the cost of repeatedly training separate candidate architectures. The core idea is to relax the search problem so that many architectural choices can be embedded into a single over-parameterized model, often referred to as a supernet, and optimized jointly through shared weights. In differentiable NAS, architectural decisions are parameterized continuously, allowing gradient-based optimization; in one-shot variants, the search is similarly performed within a shared model. These approaches can dramatically accelerate search, but they also introduce a potential mismatch between the ranking induced by the shared supernet and the true standalone performance of the sampled architectures. Prominent examples include DARTS [69] and ENAS [96], which have strongly influenced subsequent NAS research.

Bayesian optimization

Bayesian optimization (BO) is particularly suited to optimization problems in which evaluating a candidate solution is expensive. In NAS, this problem arises naturally, since, unless proxy objectives are used, each evaluation of the objective function may require partial or full training, or at least the computation of a non-trivial performance estimate. Building on the NAS formulation introduced in (2.1), BO approaches architecture search by treating $J(\Theta)$, i.e. the objective associated with a candidate architecture, as an expensive black-box function. Rather than evaluating a large number of candidates uniformly, BO aims to identify promising architectures with as few evaluations as possible by learning, from past observations, a probabilistic surrogate of the objective. In the NAS literature, BO has therefore emerged [113], [133], [143] as a sample-efficient alternative to more expensive exploration strategies, especially when the search budget is limited and each evaluation is expensive.

At iteration t , let $\mathcal{D}_t = \{(\Theta_i, y_i)\}_{i=1}^t$ denote the set of evaluated candidates, where Θ_i is an architectural configuration and y_i is the corresponding observed value of the objective. Bayesian optimization fits a probabilistic surrogate model to $\mathcal{D}_t$, which defines, for each candidate Θ , a predictive distribution over its unknown performance. Then, the next candidate is selected by maximizing an acquisition function:

$$\Theta_{t+1} = \arg \max_{\Theta \in \mathcal{A}} \alpha(\Theta, \mathcal{D}_t), \quad (2.4)$$

where $\alpha(\Theta, \mathcal{D}_t)$ balances the exploitation of candidates with high predicted performance and exploration of candidates whose performance is still uncertain. In this

sense, BO does not search directly on the true objective in (2.1), but on a sequence of auxiliary optimization problems induced by the surrogate and the acquisition rule. A common acquisition criterion is the Expected Improvement (EI), which measures the expected gain over the best value observed so far, denoted by y^+ :

$$\alpha_{\text{EI}} = \mathbb{E} \left[\max \left(0, J(\Theta) - y^+ \right) \mid \mathcal{D}_t \right]. \quad (2.5)$$

A candidate may be selected either because it has a high predicted objective value or because its uncertainty leaves open the possibility of substantial improvement. Other choices, such as upper confidence bound criteria, follow the same principle. The key idea is that candidate selection is guided not only by what has already been observed, but also by how uncertain the model remains in different regions of the search space.

The effectiveness of BO in NAS depends crucially on the surrogate representation of architectures. Unlike standard hyperparameter optimization, NAS typically involves discrete, structured and variable-topology objects, which are not immediately suited to conventional continuous domain BO. For this reason, BO-based NAS methods typically require an encoding of the architecture into a representation suitable for surrogate modeling, or alternatively a surrogate specifically designed for graph-structured inputs. This aspect is one of the main practical challenges of BO in architecture search: while the optimization itself is conceptually sample-efficient, its success depends on whether the surrogate can capture meaningful similarity relations between architectures. Early methods such as NASBOT [55] explicitly addressed this issue through architecture-aware modeling, while later approaches such as BANANAS [133] combined BO with learned predictors tailored to neural architectures.

Another important aspect is scalability. Classical BO is most effective when evaluations are very expensive and the number of observations remains moderate. As the search space grows in dimensionality and combinatorial complexity, maintaining an accurate probabilistic model becomes more difficult, and the optimization of the acquisition function may itself become non-trivial. In NAS, this limitation is particularly evident in large search spaces with heterogeneous design variables, where architectural topology, depth, width and training hyperparameters may all vary simultaneously. For this reason, BO is often most attractive in settings where the search problem is either moderately sized or intentionally restricted to a subset of architectural decisions. In such cases, its ability to prioritize informative evaluations may compensate for the additional modeling complexity.

An extension for NAS, particularly relevant for this thesis, is multi-fidelity Bayesian optimization, in which candidate architectures can be evaluated under different budgets. Letting $\mathcal{D}_t = \{(\Theta_i, r_i, y_i)\}_{i=1}^t$, where r_i denotes the budget associated with the i -th evaluation, the search process for (2.4) can exploit cheap

low-fidelity observations to discard poor candidates early, while reserving more expensive high-fidelity evaluations for the most promising ones. In practice, the budget may correspond to the number of training epochs, the amount of data used for evaluation, or another resource-related quantity. This setting is especially natural in NAS, since architecture evaluation is rarely all-or-nothing: partial information can often be obtained much more cheaply than a full assessment. Among the methods in this family, BOHB [31] is particularly notable because it combines model-based candidate selection with Hyperband-style [64] resource allocation. Instead of relying exclusively on expensive full-budget evaluations, BOHB progressively allocates larger budgets to promising configurations, thereby improving efficiency while retaining the principled exploration mechanism of Bayesian optimization. This combination is especially useful when the search jointly involves architectural and hyperparameter choices, as it enables the algorithm to exploit partial evaluations without abandoning probabilistic guidance. In this thesis, such a strategy is particularly relevant because it underlies the search procedure adopted in Chapter 3.

Evolutionary algorithms

Building on the NAS formulation introduced in Section 2.1.1, evolutionary algorithms [72] approach architecture search as a black-box optimization problem over structured and typically discrete design variables. Rather than exploiting gradient information or an explicit probabilistic model of the objective, they maintain a population of candidate architectures and iteratively improve it through selection and variation. This perspective is particularly natural in NAS, where architectures may differ in connectivity patterns, operations, depth, or width, and where the objective associated with a candidate is often expensive, noisy, and not differentiable. For these reasons, evolutionary methods have become one of the most established search paradigms in NAS, especially in the setting where the search space is combinatorial and difficult to optimize through continuous relaxations.

At iteration t , let $P_t = \{\Theta_0, \Theta_1, \dots, \Theta_N\}$ denote the current population of N candidate architectures, and let $\hat{J}(\Theta)$ denote the fitness associated with a candidate Θ , where $\hat{J}$ may correspond either to the true objective or to an estimate obtained through partial training or other proxies. An evolutionary NAS algorithm alternates between parent selection, offspring generation, and population replacement. In abstract form, this can be written as

$$\Theta_t^{\text{parent}} \sim \mathcal{S}(P_t, \hat{J}), \quad \Theta_t^{\text{offspring}} \sim \mathcal{V}(\Theta_t^{\text{parent}}) \quad (2.6)$$

or, when recombination is used,

$$\Theta_t^{\text{offspring}} \sim \mathcal{V}(\Theta_t^{\text{parent}(1)}, \Theta_t^{\text{parent}(2)}), \quad (2.7)$$

where $\mathcal{S}$ denotes the selection rule and $\mathcal{V}$ the variation operator. After evaluating the offspring, a replacement step yields the next population P_{t+1} . Although this scheme is conceptually simple, it is flexible enough to accommodate different encodings, structural constraints, and evaluation procedures.

Mutation and crossover play complementary roles. Mutation applies local changes to an existing architecture, for example by modifying an operation, rewiring a connection, or altering a structural hyperparameter. Crossover instead combines information from two parents, potentially generating offspring that differ more substantially from either parent. In practice, evolutionary NAS often relies more heavily on mutation than on crossover [42], especially when architectures are represented as graphs or variable-length structures and recombination becomes harder to define robustly [72]. Selection mechanisms are similarly varied [72], and include elitist strategies [116], roulette-wheel selection [138], tournament selection [118], and age-based replacement [100].

A particularly influential instance of this family is regularized evolution, also known as aging evolution, introduced in NAS by [100]. In this approach, the population size is kept fixed, but replacement is governed by age rather than by worst fitness. More specifically, at iteration t , a subset $S_t \subset P_t$ of size n is randomly sampled, and the parent is chosen as the best individual within that subset:

$$\Theta_t^{\text{parent}} = \arg \max_{\Theta \in S_t} \hat{J}(\Theta), \quad |S_t| = n. \quad (2.8)$$

A mutated offspring $\Theta_t^{\text{offspring}}$ is then generated from Θ_t^{parent} , evaluated, inserted into the population, and the oldest individual currently in the population is removed:

$$P_{t+1} = (P_t \cup \{\Theta_t^{\text{offspring}}\}) \setminus \{\text{oldest}(P_t)\}. \quad (2.9)$$

Replacement by age has an important effect: by removing the oldest candidate rather than the worst-performing one, the algorithm avoids concentrating too early around a small set of highly fit architectures and promotes a continuous turnover of the population. In this sense, aging evolution acts as an implicit regularizer of the search dynamics, encouraging broader exploration while preserving the simplicity of mutation-based search.

Regularized evolution is especially well suited to NAS for several reasons. First, it preserves the black-box nature of evolutionary search, and therefore does not require differentiability or handcrafted continuous relaxations. Second, tournament-based parent selection provides a simple balance between exploration and exploitation: only a small subset of the population is compared at each step, but better candidates are still preferentially selected. Third, age-based replacement prevents old individuals from dominating the population indefinitely, which could cause premature convergence. In the original REA [100] setting, this strategy was applied

to a cell-based search space, where candidate cells were progressively evolved and later stacked to form the final network. More generally, the family of methods derived from regularized evolution has become one of the main reference points for mutation-based NAS.

Evolutionary search is attractive for NAS because it combines a simple search procedure with the ability to operate directly on discrete and structured representations. At the same time, its practical effectiveness depends critically on the quality and cost of the fitness evaluation, since the search itself is usually inexpensive compared with the repeated assessment of candidate architectures. In this respect, the main limitation of evolutionary methods in NAS is the fact that standard training-based evaluation can make the overall procedure computationally prohibitive. This observation is particularly relevant to this thesis, which focuses on the practical adoption of NAS methods. In this context, evolutionary search provides the conceptual foundation for the training-free approach presented in Chapter 4. Specifically, FreeREA builds on the regularized evolution framework, while introducing modifications to the evaluation stage and enhancing the offspring generation mechanism.

2.2.3 Architecture evaluation and efficient performance estimation

Once the search space and the search strategy have been defined, the practical effectiveness of a NAS method depends crucially on how candidate architectures are evaluated during the search. This aspect represents one of the main bottlenecks of Neural Architecture Search. In principle, the most faithful way to assess an architecture is to fully train the corresponding model and measure its validation performance under the target protocol. Such an evaluation provides the closest approximation to the true objective introduced in Section 2.1.1, since it reflects the behavior of the architecture after the complete optimization process. However, repeating full training for a large number of candidates is typically prohibitively expensive, especially when the search space is large or the models themselves are computationally expensive. As a consequence, a large part of the NAS literature can be interpreted as an attempt to construct evaluation procedures that remain informative enough to guide the search, while being substantially cheaper than full training.

A first way to reduce this cost is to evaluate candidate architectures under a limited budget, rather than through a complete training pipeline. In this case, the estimator $\hat{J}(\Theta)$ is obtained from a partial observation of architectural performance, for instance after a reduced number of optimization steps, on a smaller subset

of data, or under another simplified evaluation protocol. More generally, multi-fidelity approaches exploit the fact that architecture evaluation is often not an all-or-nothing process: low-cost observations may already be sufficient to discard clearly suboptimal candidates, while only a smaller subset of promising configurations is evaluated more thoroughly. This principle is particularly useful when some form of training-based assessment remains necessary, but a uniform full-budget evaluation of all candidates would be infeasible. At the same time, these approaches only alleviate the computational burden of NAS, rather than removing it entirely, since they remain tied to the training dynamics of the candidate models.

A more radical approach consists in replacing training-based evaluation during search with metrics that can be computed directly at initialization. In training-free or zero-cost NAS, the estimator is no longer obtained from the partially or fully trained performance of the model, but from a score derived from the architecture and from quantities available before training. Such an estimator may be written as

$$\hat{J}_{\text{tf}}(\Theta) = s(\Theta, \theta_0, \mathbf{x}_0), \quad (2.10)$$

where $s(\cdot)$ denotes a training-free scoring function, θ_0 represents the model weights at initialization and $\mathbf{x}_0$ is a small input batch. It must be noted that not all scoring functions require input data. For data-agnostic methods, the scoring function can be simply written as $s(\Theta, \theta_0)$. The key idea is that the resulting score does not directly measure the final predictive performance of the architecture, but serves as a proxy based on properties that may correlate with it. Depending on the specific metric, these properties may be related to gradient propagation, activation behavior, expressivity, or other structural characteristics of the network.

The appeal of training-free evaluation lies in its efficiency. By avoiding repeated training of the candidate architectures, it can reduce the computational cost of NAS by orders of magnitude and make search feasible even under limited hardware and time budgets. For this reason, training-free estimators are particularly relevant in the context of efficient NAS, and they play a central role in the methods discussed later in this thesis. Their use, however, also makes the distinction between the true objective and the accessible evaluation metric especially explicit. Unlike full-training evaluation, training-free scores are only indirectly related to the quality of the final trained model, and their predictive power is inherently dependent on the setting in which they are used.

More specifically, no training-free proxy should be regarded as universally reliable. Its usefulness depends on the search space, on the task, and on the target notion of quality. A metric that is informative in one architectural domain may become much less predictive in another, and a proxy that correlates with in-distribution accuracy may not be equally informative when the evaluation criterion involves behavior under distribution shift. Moreover, different zero-cost metrics

often reflect different aspects of the architecture rather than a single unified notion of quality. Some are more closely related to optimization and signal propagation, while others are better suited to capture structural or expressive properties of the model. As a result, the problem of efficient architecture evaluation is not only computational, but also methodological: the search behavior depends critically on which architectural properties are made visible to the optimizer through the adopted estimator.

For completeness, it should be noted that efficient NAS also includes other important paradigms, most notably weight-sharing or one-shot methods, and predictor-based approaches, already briefly described in Section 2.2.2. In the former, many candidate architectures are embedded into a shared supernet and assessed through inherited weights; in the latter, a learned predictor is used to estimate the performance of new candidates from previously observed ones. Although both directions are highly relevant in the broader NAS literature, they are not central to the methods proposed in this thesis and are therefore not discussed here in detail. What matters for the present discussion is that all efficient evaluation strategies introduce, in different forms, an additional layer of approximation between the signal optimized during search and the quality of the final discovered architecture.

2.2.4 Benchmark-based NAS evaluation

The cost of architecture evaluation affects not only the deployment of NAS pipelines in real-world scenarios, but also the development and validation of new NAS methods. This remains true even for training-free approaches, which avoid training candidate models during the search but still require reliable ground-truth performance values to assess the effectiveness of the proposed proxies. More specifically, zero-cost metrics can only be meaningfully validated by measuring how well their rankings correlate with the final accuracy of trained architectures. Similarly, search strategies can only be compared fairly when the architectures they select are evaluated under a common protocol. If different methods rely on different search spaces, training recipes, random seeds or data splits, their relative performance may reflect differences in the experimental setup rather than the quality of the search strategy itself. NAS benchmarks address these issues by providing fixed search spaces together with precomputed information about the candidate architectures, enabling reproducible and computationally inexpensive comparisons among search methods.

A NAS benchmark can be seen as an evaluation infrastructure associated with a predefined architecture space. For each architecture, the benchmark provides one or more quantities of interest, such as validation accuracy, test accuracy, training

time, number of parameters, FLOPs, or hardware latency. During the evaluation of a NAS algorithm, candidate architectures can then be queried from the benchmark rather than trained from scratch. This makes it possible to study the behavior of search strategies under controlled conditions and to repeat experiments many times at negligible computational cost. In this sense, NAS benchmarks are useful when the objective is not to train a final model, but to analyze how effectively a search procedure explores a space and identifies high-performing candidate architectures.

Two main benchmark families can be distinguished. **Tabular benchmarks** enumerate a finite search space and store the results obtained by training all candidate architectures under a common protocol. This design provides direct access to measured performance values and is particularly convenient for small or moderately sized spaces. However, it becomes impractical when the search space is too large to be exhaustively evaluated. **Surrogate benchmarks** address this limitation by training a predictive model on a subset of evaluated architectures and using it to estimate the performance of unseen candidates. This allows benchmark-based evaluation to scale to larger spaces, at the cost of introducing an additional approximation layer between the queried score and the actual trained model performance.

Several tabular benchmarks have become standard testbeds in the NAS literature. NAS-Bench-101 [143] is one of the earliest and most influential examples. It defines a cell-based search space for CIFAR-10 [56], where candidate cells are represented as directed acyclic graphs with a limited number of nodes and edges. The benchmark provides repeated training evaluations for more than 423,000 unique architectures, making it possible to compare search algorithms without retraining the sampled models. NAS-Bench-201 [27] later introduced a more compact cell-based search space evaluated on multiple datasets, namely CIFAR-10 [56], CIFAR-100 [56] and ImageNet-16-120 [19]. NATS-Bench [26] further extends this line of work by considering both a topology search space, inherited from NAS-Bench-201, and a size search space in which the number of channels is varied. This makes NATS-Bench particularly useful for studying not only topological decisions, but also dimensional choices that affect model capacity and computational cost.

Despite their usefulness, NAS benchmarks also introduce important limitations. Since they are tied to fixed search spaces and training protocols, conclusions drawn on one benchmark may not necessarily transfer to other architectural families or deployment scenarios. Many widely used benchmarks rely on small datasets such as CIFAR-10 or CIFAR-100, or on heavily reduced ImageNet variants such as ImageNet-16-120. Moreover, several of them are centered on cell-based convolutional spaces, which only partially reflect the structure of modern vision architectures. These limitations are especially relevant when the goal is to evaluate methods for efficient, hardware-aware, hybrid, or robustness-aware architecture search.

In this thesis, NAS benchmarks play two distinct roles. In Chapter 4, NAS-Bench-101 and NATS-Bench are used as controlled experimental environments to evaluate training-free search methods and to analyze the behavior of zero-cost proxy metrics. In Chapter 5, the focus shifts from using existing benchmarks to studying NAS benchmarking itself: the limitations of standard benchmarks motivate the introduction of HyViTas-Bench, which targets hybrid CNN-ViT architectures, larger-scale data, out-of-distribution evaluation, and hardware latency.

2.3 Architectures for Computer Vision

Although Neural Architecture Search is a general methodology, much of its development has been driven by computer vision, particularly image classification. This is also the main focus of the present thesis. Accordingly, after discussing NAS from a general perspective, this section briefly reviews the main architectural paradigms in computer vision that are most relevant to the contributions presented in the following chapters.

2.3.1 Convolutional architectures and residual design

At the most basic level, a neural network layer can be viewed as a parametric transformation that maps an input representation into a new feature space. A standard fully connected layer applies an affine transformation of the form

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}, \quad (2.11)$$

where $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, $\mathbf{y} \in \mathbb{R}^{d_{\text{out}}}$, $\mathbf{b} \in \mathbb{R}^{d_{\text{out}}}$, and $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. While this formulation is general and expressive, it does not exploit the spatial structure of images, since each output component depends on all input dimensions through independent weights. For visual data, this lack of spatial inductive bias is both statistically and computationally inefficient. A Convolutional Neural Network (CNN) addresses this limitation by imposing locality and weight sharing. Given an input tensor $\mathbf{X} \in \mathbb{R}^{C_{\text{in}} \times H \times W}$, a convolutional layer produces an output tensor $\mathbf{Y} \in \mathbb{R}^{C_{\text{out}} \times H' \times W'}$ by applying learnable kernels over local spatial neighborhoods. Denoting by $\mathbf{K} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k_h \times k_w}$ the convolutional kernels, the operation can be written as

$$Y_{c,i,j} = \sum_{c'=1}^{C_{\text{in}}} \sum_{u=1}^{k_h} \sum_{v=1}^{k_w} \mathbf{K}_{c,c',u,v} \mathbf{X}_{c',i+u,j+v} \quad (2.12)$$

up to the chosen padding and stride convention. In contrast to fully connected layers, the same kernel parameters are reused across spatial positions, which greatly

reduces the number of free parameters and embeds the prior assumption that similar local patterns may appear at different locations in the image.

This design induces two key properties that have made convolutional architectures central to computer vision. First, convolutions naturally support local feature extraction, enabling the detection of edges, textures, and progressively more complex visual motifs through hierarchical composition across layers. Second, the repeated aggregation of local information across depth yields increasingly large effective receptive fields, allowing deep convolutional networks to capture both fine-grained and higher-level semantic structure. These properties make CNNs particularly suitable for image classification and, more generally, for visual recognition problems in which spatial regularity and translation-related invariances play an important role.

Modern convolutional architectures are also strongly shaped by the introduction of residual learning. A residual block can be described as

$$\mathbf{y} = \mathbf{x} + \mathcal{F}(\mathbf{x}, \boldsymbol{\theta}), \quad (2.13)$$

where $\mathcal{F}$ denotes a learnable residual function, typically implemented by a small sequence of convolutions, normalizations and nonlinearities. When the dimensions of input and output do not match, the skip connection can be replaced by a projection $P(\mathbf{x})$, yielding

$$\mathbf{y} = P(\mathbf{x}) + \mathcal{F}(\mathbf{x}, \boldsymbol{\theta}). \quad (2.14)$$

The shortcut introduced by this reformulation provides a direct path for both forward activations and backward gradients, mitigating degradation phenomena and facilitating the training of substantially deeper networks. Residual connections are a structural component that directly affects trainability and information propagation. By preserving a shortcut path across layers, they facilitate optimization, support feature reuse, and help stabilize deep architectures. For this reason, skip-based designs have become a common pattern in modern visual backbones, including several efficient and hybrid models.

These aspects are also relevant for Neural Architecture Search. In CNN-based search spaces, candidate architectures often differ not only in the choice of operators, but also in the presence and arrangement of their connections. Such choices influence both final performance and optimization behavior, and are especially important when architecture quality is estimated through training-free proxies that depend on signal propagation.

2.3.2 Vision Transformers

The success of Transformers in Natural Language Processing (NLP) [128] motivated the exploitation of attention-based architectures for visual tasks. In computer vision, convolutional networks had long remained dominant because their locality and translation-related inductive biases are well aligned with image data. However, the same inductive biases that make CNNs effective also impose a fixed, local interaction structure. This can make it difficult to capture long-range dependencies and global context. Vision Transformers (ViT) emerged from the idea that visual recognition could instead be addressed through a more general sequence-processing paradigm, in which image regions are represented as tokens and their interactions are learned through self-attention. This perspective introduced a markedly different architectural design, which has become increasingly relevant in modern vision models and in the neural architecture search settings considered in this thesis.

A Vision Transformer adapts the standard Transformer encoder to images by converting the input into a sequence of patch-level embeddings [29]. Given an image $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$, the image is partitioned into N non-overlapping patches of spatial size $P \times P$, where $N = \frac{HW}{P^2}$. Each patch is flattened to a vector $\mathbf{x}_p \in \mathbb{R}^{P^2 C}$ and linearly projected into a latent space of dimension D . In the original ViT formulation [29], a learnable classification token $\mathbf{x}_{\text{cls}}$ is prepended to the patch sequence, and a learnable positional embedding is added to retain information about the order of the tokens. The resulting input sequence can be written as

$$\mathbf{z}_0 = [\mathbf{x}_{\text{cls}}, \mathbf{x}_p^{(1)} \mathbf{E}, \mathbf{x}_p^{(2)} \mathbf{E}, \dots, \mathbf{x}_p^{(N)} \mathbf{E}] + \mathbf{E}_{\text{pos}}, \quad (2.15)$$

where $\mathbf{E} \in \mathbb{R}^{(P^2 C) \times D}$ is a patch projection matrix and $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$ is a learnable matrix containing one positional embedding for each token position, including the classification token. In the original model, these are standard learnable one-dimensional absolute positional embeddings associated with the rasterized order of the patches.

The core operator of the model is self-attention. Given a matrix of token embeddings $\mathbf{Z} \in \mathbb{R}^{(N+1) \times D}$ when the classification token is included, three learned projections are computed:

$$\mathbf{Q} = \mathbf{Z} \mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z} \mathbf{W}_K, \quad \mathbf{V} = \mathbf{Z} \mathbf{W}_V, \quad (2.16)$$

with $\mathbf{Q}$ containing the queries, $\mathbf{K}$ the keys, and $\mathbf{V}$ the values. Intuitively, the query associated with a token specifies what information that token is seeking from the rest of the sequence; the keys describe how each token can be matched by other tokens; the values contain the information that can be aggregated once such

matches have been established. The attention output is then computed as

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \mathbf{V}, \quad (2.17)$$

where d_k is the key dimensionality. The matrix $\mathbf{Q}\mathbf{K}^\top$ measures pairwise similarities between tokens, the softmax normalizes these similarities into attention weights, and the weighted combination of $\mathbf{V}$ yields a content-dependent aggregation of information across the sequence. In practice, this mechanism is implemented in multi-head form, so that several attention operations are performed in parallel over different learned subspaces.

A standard ViT block alternates multi-head self-attention and a position-wise feed-forward network, with layer normalization and residual connections around each sub-block. Stacking such blocks allows the model to progressively refine the token representations while maintaining direct interactions among all tokens at every layer. In the original formulation, the output associated with the classification token after the last encoder block is used as a global image representation for classification. The design follows the standard Transformer encoder closely, with only a limited number of image-specific modifications, namely patch extraction and positional encoding.

Compared with convolutional architectures, Vision Transformers rely on a much weaker built-in inductive bias. Convolutions explicitly encode locality, spatial neighborhood structure, and weight sharing across positions. By contrast, self-attention performs global and content-dependent mixing among tokens, while spatial structure is injected only at the input stage through patch partitioning and positional encoding. This difference has important consequences. On the one hand, ViTs can model long-range interactions more directly than CNNs, since a token may attend to any other token already in a single layer. On the other hand, the reduced architectural bias generally makes pure transformers more data-hungry and more dependent on suitable training procedures, especially when compared with CNNs in moderate scale regimes. The original ViT study highlighted this aspect by showing that transformer-based vision models become particularly competitive when trained at sufficiently large scale.

From a practical viewpoint, vision transformers also introduce a different efficiency profile. Since self-attention scales quadratically with the sequence length, its computational and memory cost becomes increasingly demanding at higher input resolutions, which can limit direct deployment in resource constrained settings. In this thesis, transformer-based architectures are relevant mainly in connection with the hybrid CNN-ViT models studied in Section 4.4 and Chapter 5, where they enlarge the search space beyond classical CNN designs.

2.3.3 Efficient and hybrid vision architectures

Recent progress in computer vision has made architectural efficiency a central design requirement, especially in settings where models must operate under strict constraints on computation, memory, or latency, such as mobile and edge deployment. This has encouraged the development of visual backbones that seek a favorable trade-off between representational capacity and deployment cost through increasingly refined choices at the level of blocks, stages and feature transformations. In this context, efficiency is not determined only by the number of parameters or floating-point operations, but also by the structure of the adopted operators and by the way local and more global interactions are distributed across the network. These considerations have progressively brought convolutional and attention-based design principles closer, motivating architectures that combine elements from both families. This trend is particularly relevant for Neural Architecture Search, since it leads to richer search spaces in which efficiency constraints and architectural heterogeneity are tightly coupled.

Efficient visual backbones often rely on a small set of recurring architectural patterns. A first example is given by depthwise separable convolutions, which decompose spatial and channel mixing and can substantially lower the cost with respect to standard convolutions. A related and widely adopted pattern is the Inverted Bottleneck (IB), in which features are first expanded in the channel dimension, processed in a higher-dimensional space, and then projected back to a more compact representation. In this setting, the expansion ratio becomes an important design variable, since it directly controls the trade-off between intermediate representational capacity and computational cost. More recently, efficient design has also revisited the role of larger convolutional kernels, which can enlarge the effective receptive field and improve spatial mixing without abandoning the convolutional paradigm. This evolution is well illustrated by ConvNeXt [73], which shows that several architecture choices often associated with transformer-era design, including patchified stems, depthwise convolutions, Inverted Bottlenecks, larger kernels and a simplified block structure, can be integrated into a modern pure CNN backbone with strong accuracy and robust scalability. Taken together, these patterns define a broader design space in which efficiency emerges from the balance between locality, capacity and computational cost.

For these reasons, many recent vision models adopt hybrid designs that combine convolutional and transformer-inspired components within the same backbone [17], [67], [84]. The motivation is straightforward: convolutions provide strong local inductive biases and efficient spatial processing, while attention-based modules enable more flexible interactions across distant regions of the image. Rather than

replacing one paradigm with the other, hybrid architectures exploit different combinations of these mechanisms across stages or blocks, for instance by retaining convolutional processing in the early and computationally heavier parts of the network while introducing attention-based components more selectively where broader context may be more beneficial. This makes hybrid models especially attractive in efficient settings, where the goal is to enrich feature interactions without incurring the cost of a fully transformer-based design. At the same time, it also increases architectural heterogeneity, since relevant design choices now include not only depth, width, kernels and expansion ratios, but also where and how different forms of token or feature mixing are introduced within the backbone.

From the perspective of Neural Architecture Search, this architectural evolution has two main implications. First, hybrid and efficiency-oriented backbones define search spaces that are richer than classical CNN-centric settings, since candidate architectures may differ not only in topology or scale, but also in the distribution of heterogeneous building blocks across the network. Second, evaluating such candidates becomes less straightforward, because different architectural choices may affect different properties of the model, including representational capacity, optimization behavior, efficiency, and robustness. This makes the design of informative evaluation signals particularly important, especially when search is guided by training-free proxies rather than by full training. It also motivates the adoption of more realistic benchmarks, in which hybrid architectures can be assessed beyond small-scale classification accuracy and under a broader set of criteria. These aspects are especially relevant for Chapter 5.

Chapter 3

NAS for Unsupervised Domain Adaptation

The content, images, and tables included in this chapter were originally presented in the following publication:

- *L. Robbiano, M. R. Ur Rahman, F. Galasso, B. Caputo, and F. M. Carlucci, Adversarial branch architecture search for unsupervised domain adaptation, in Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2022, pp. 1008–1018*

3.1 Introduction

Unsupervised Domain Adaptation (UDA) aims to transfer knowledge from a labeled source domain to an unlabeled target domain, where data may be scarce and annotations are unavailable. Although UDA performance has steadily improved in recent years, effective domain transfer is still often tied to specific experimental configurations. Consequently, when a new backbone architecture becomes available, adapting an existing UDA pipeline to fully exploit it typically requires a slow, manual trial-and-error process. Although Neural Architecture Search (NAS) has recently reduced the reliance on hand-crafted heuristics in network design, it cannot be straightforwardly transferred to the UDA setting for two key reasons. First, standard NAS pipelines depend on labeled target data to validate candidate models during the search, whereas UDA assumes no target annotations. Second, state-of-the-art UDA for visual recognition, spanning classification, detection and segmentation, typically hinges on leveraging pretrained backbone networks, which clashes

with the common NAS practice of searching entire architectures from scratch. To the best of our knowledge, these two constraints have not been jointly addressed in prior work.

In this work, we introduce Adversarial Branch Architecture Search for UDA, a novel NAS framework designed to overcome both of the aforementioned obstacles. Our approach is grounded in robust UDA frameworks that rely on auxiliary adversarial branches [15], [33], where a secondary network (the adversarial branch) is attached to a primary, pretrained backbone. The backbone is trained in a supervised manner on labeled source data, whereas the auxiliary branch promotes alignment between source and target by adversarially encouraging domain-invariant representations, making the two domains indistinguishable. Despite its substantial impact on final performance, the design of this auxiliary branch has so far been largely hand-crafted.

Our first contribution tackles the absence of target annotations through a data-driven Ensemble approach for Model Selection (EMS), designed to operate directly on target-domain data. Model selection in UDA is inherently difficult and is often treated as a secondary issue: only a small number of prior studies address it, and they typically rely exclusively on source-domain features. In contrast, we show that explicitly leveraging target-domain representations leads to more effective domain transfer.

As a second core contribution, we propose a multi-fidelity Bayesian optimization strategy based on BOHB [31] to jointly explore the architecture and hyperparameter configuration of the auxiliary adversarial branch. Crucially, ABAS preserves, and explicitly leverages, the backbone design and its pre-training, which is essential to achieve and surpass state-of-the-art results in practical UDA settings.

We perform a detailed performance evaluation by applying ABAS to a widely used UDA baseline, DANN [33], and to a more recent state-of-the-art approach, ALDA [15]. Their ABAS-enhanced variants are evaluated across three standard computer vision benchmarks: PACS [60], Office31 [106] and Office-Home [129]. The results show that the default adversarial branches in DANN and ALDA can be effective in certain combinations of backbone and source-target pairs, particularly Office31 and Office-Home, where they perform substantially better than randomly sampled architectures. In contrast, these hand-crafted branches are less competitive on PACS, where ABAS is able to identify the strongest-performing UDA design.

3.2 Related works

3.2.1 Unsupervised Domain Adaptation (UDA)

Previous works on UDA have been divided into four broad families of approaches [149]: discrepancy-based, adversarial discriminative, adversarial generative and self-supervised methods. Among these, adversarial discriminative techniques often achieve the strongest results; however, their effectiveness is sensitive to the design of the auxiliary branch. When this component is not properly optimized, training can become unstable, potentially leading to divergence, and performance can vary substantially across architectural choices. We find this provides a strong motivation for investigating architecture search within adversarial discriminative UDA frameworks.

Following the approach proposed in [33], UDA can be tackled by learning domain-invariant representations through the combination of an auxiliary branch and adversarial training. In this paradigm, an additional branch is connected to the backbone and trained to predict the domain of each sample. Its gradients are then reversed during backpropagation, encouraging the backbone to learn the ability to extract features that are indistinguishable across source and target domains. Several extensions are worth highlighting. The approach proposed in [48] leverages cross-covariance statistics across samples, [75] incorporates the classifier predictions, and [126] carries a related idea over to the semi-supervised regime. In addition, [10] pushes discriminative gradient reversal down to the pixel level, while [15] integrates adversarial alignment with self-training. Despite proposing concrete instantiations of the auxiliary adversarial branch, these methods provide limited guidance on how to tailor the branch to different backbone architectures. To the best of our knowledge, our work is the first to introduce a data-driven pipeline for learning this component.

3.2.2 Model Selection for UDA

Hyper-parameter tuning and model selection are particularly difficult in UDA because target annotations are unavailable. As a result, many studies adopt a pragmatic strategy and keep hyperparameters fixed [144] across multiple datasets. A more principled alternative, Importance-Weighted Cross-Validation (IWCV), was introduced in [117]: source samples that are more similar to the target distribution receive higher weights. While IWCV yields an unbiased estimate, it can suffer from unbounded variance. Deep Embedded Validation [144] was proposed to mitigate this issue and to make IWCV applicable to deep feature representations. Similarly,

[88] develops a reduced-variance IWCV estimator for hyperparameter optimization in multi-source domain adaptation. Overall, existing approaches remain largely source-centric, only indirectly accounting for target characteristics. In contrast, we introduce a performance estimator that relies primarily on target-domain features.

3.3 Method

We introduce a data-driven pipeline for identifying the optimal auxiliary-branch architecture in unsupervised domain adaptation. As discussed in Section 3.2.1, several UDA methods rely on an auxiliary branch; while our analysis focuses on approaches rooted in domain-adversarial learning, the proposed strategy can be readily extended to other families of UDA techniques. In this section, we instantiate ABAS on (i) DANN, the seminal adversarial UDA method, and (ii) ALDA, which represented the state of the art at the time this work was first proposed. In both cases, the auxiliary branch is trained adversarially to narrow the source-target discrepancy at the feature level.

3.3.1 Background

For conciseness, we provide a mainly intuitive overview here, and refer the reader to the cited works for a formal and more complete presentation of the underlying methods.

Adversarial branches for UDA. Let $\mathbf{X}_s = \{\mathbf{x}_s^i, y_s^i\}_{i=0}^{N_s}$ denote a dataset sampled from a labeled source domain S , and $\mathbf{X}_t = \{\mathbf{x}_t^i\}_{i=0}^{N_t}$ a dataset drawn from a different, unlabeled target domain T , where the two domains share the same set of classes. The objective is to maximize classification accuracy on $\mathbf{X}_t$, while training on $\mathbf{X}_s$. As illustrated in Figure 3.1, the overall model comprises three components: a feature extractor G , a label predictor C , and an adversarial branch D . The key idea behind discriminative adversarial domain adaptation is to train a domain discriminator to distinguish between source and target samples; simultaneously, the feature extractor G is encouraged through gradient reversal to produce representations that make the two domains indistinguishable. Training proceeds by minimizing the following objective function:

$$\min_{G, C, D} \mathcal{L}_{CE}(C(G(x_s)), y_s) + \lambda \mathcal{L}_{adv}(D(G(x_s \cup x_t))), \quad (3.1)$$

where $\mathcal{L}_{CE}$ denotes the standard cross-entropy loss, evaluated only on labeled source samples, and $\mathcal{L}_{adv}$ is the adversarial loss that encourages alignment between source

and target distributions by reducing the domain discrepancy. In the case of DANN, the adversarial loss is

$$\mathcal{L}_{\text{adv}}(D(G(\mathbf{x}_i)), d_i) = d_i \log \frac{1}{D(G(\mathbf{x}_i))} + (1 - d_i) \log \frac{1}{1 - D(G(\mathbf{x}_i))}, \quad (3.2)$$

where $d_i \in \{0, 1\}$ is a binary variable indicating whether the sample $\mathbf{x}_i$ belongs to the source or target domain. The more recent method ALDA integrates domain-

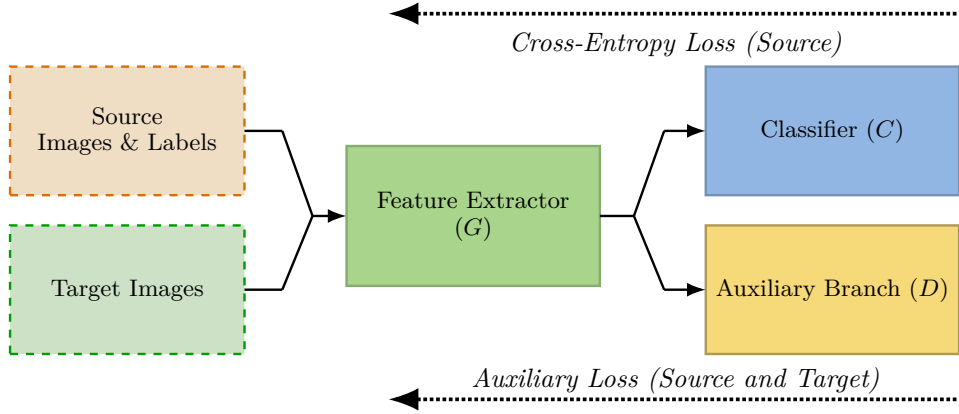


Figure 3.1: General framework shared by several UDA methods. The auxiliary branch D is attached to the feature extractor and can be used either to implement a domain discriminator [15], [33] or to compute a self-supervised auxiliary loss [9], [119].

adversarial training with self-training. Although its loss function comprises several components, in the following we focus on the adversarial terms only:

$$\mathcal{L}_{\text{adv}}(\mathbf{x}_s, y_s, \mathbf{x}_t) = \mathcal{L}_{\text{BCE}}(c^{(\mathbf{x}_s)}, y_s) + \mathcal{L}_{\text{BCE}}(c^{(\mathbf{x}_t)}, u^{(\hat{y}_t)}), \quad (3.3)$$

with $\mathcal{L}_{\text{BCE}}$ denoting the binary cross-entropy loss, $c^{(\mathbf{x})}$ represents the corrected pseudo-label vector obtained by multiplying the confusion matrix with the pseudo-label vector $\hat{y}$, and $u^{(\hat{y}_t)}$ corresponds to the opposite distribution of $\hat{y}_t$. Intuitively, the key idea is to replace the standard domain discriminator used in DANN, that outputs a domain probability, with a discriminator branch that predicts a confusion matrix. This matrix is subsequently used to refine the pseudo-label vector obtained by forwarding target samples through the network. The corrected pseudo-labels are then used as supervisory signals during training. The discriminator is optimized to yield domain-specific corrected vectors, whereas the feature extractor is trained adversarially to prevent it from distinguishing between domains. In this work, ABAS is employed to identify the highest-performing architecture for D in both DANN and ALDA.

Bayesian optimization for NAS. Several optimization strategies could be employed to search for the best adversarial branch. However, our search space is relatively compact and low-dimensional (see Section 3.3.2), making it well suited to Bayesian Optimization (BO) (Section 2.2.2). To improve efficiency, we specifically rely on BOHB [31], a multi-fidelity approach that combines BO with Hyperband [64]. Unlike standard single-fidelity BO, which evaluates every candidate using the full training budget, BOHB performs a sequence of partial evaluations with reduced budgets, pruning poorly performing configurations early and allocating more computation only to the most promising ones. As a consequence, under the same wall-clock constraints, BOHB can explore substantially more configurations and typically converges faster than alternative search strategies.

3.3.2 Search space

Competitive UDA pipelines for visual object categorization typically rely on pretrained models, often trained on ImageNet. As a result, the architectures of G and C are generally not searched from scratch, but are instead fixed as portions of a given backbone. In practice, the main design decision reduces to selecting the split point along the backbone, i.e. determining where the feature extractor G ends and the classifier C begins (Figure 3.1). Our search space (Figure 3.2) jointly

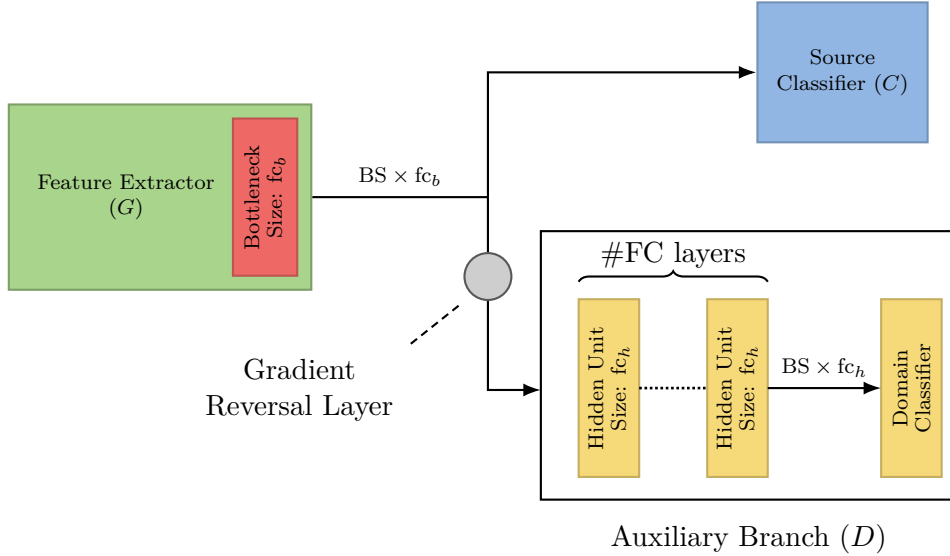


Figure 3.2: Integration of the ABAS search space into the main UDA architecture. The searchable auxiliary branch is attached after the feature extractor bottleneck and consists of a variable number of fully connected layers followed by the final domain classifier.

includes hyperparameters and architectural choices. Specifically, the domain-loss weight λ in (3.1) and the dropout rate applied to the fully connected layers of D are treated as hyperparameters. In contrast, the number of fully connected layers, the hidden dimension and the bottleneck size determine the depth and width of the auxiliary branch D . Assigning an independent width to the bottleneck layer provides an explicit handle on the branch effective capacity, by constraining the amount of information passed through it.

3.3.3 Ensemble Model Selection (EMS)

Selecting the best-performing model in the context of UDA is challenging because target labels are unavailable, thus preventing a direct performance measurement on the dataset of interest. To address this issue, we propose an ensemble-based strategy that combines multiple predictors with weak mutual correlation. Empirically, we demonstrate that a linear regressor trained on these metrics using one dataset can still recover a meaningful ranking of candidate models on a different, previously unseen dataset, underscoring the strong generalization capability of the proposed approach.

Although EMS is not tied to any particular choice of predefined metrics, in our implementation we instantiate it as an ensemble of six complementary estimators.

Entropy. In the context of domain adaptation, entropy minimization is widely adopted as a regularization term for unlabeled target samples [38], [105]. Intuitively, lower entropy reflects more confident predictions and consequently better separated classes. We compute target entropy as follows:

$$H(C(G(X_T))) = -\frac{1}{N_t} \sum_{\mathbf{x}_i \in X_T} C(G(\mathbf{x}_i)) \log(C(G(\mathbf{x}_i))). \quad (3.4)$$

Diversity. A batch-level diversity metric was proposed by [136] as an objective to be maximized to perform UDA. The measure is defined as the entropy of the predicted class distribution aggregated over the batch.

Silhouette and Caliński-Harabasz score. Target samples are first assigned their most confident predicted labels, and then clustered via K-Means using class centroids computed from these predictions. The resulting partition can be evaluated with the Silhouette [104] and Caliński-Harabasz [8] scores to quantify how well the classes are separated.

Source (weighted) loss. Classifier quality can be estimated by computing the cross-entropy loss on labeled source samples. Source samples can be weighted

according to their similarity to the target domain, following Importance Weighted Validation [117], [144]. We initially evaluated a source-weighted accuracy term within EMS. However, due to practical drawbacks, we ultimately opted for the standard unweighted source accuracy instead. More specifically, importance-weighting approaches are highly sensitive to the quality of the importance estimation, which is non-trivial to implement reliably. Moreover, they typically require a dedicated source validation split, which is not always available in small-scale settings such as Office31 [106].

Time consistent pseudo-labels. Building on the idea of [152], we define target pseudo-labels by taking the mode of the model predictions observed through training (one prediction per epoch). The underlying intuition is that the class predicted most consistently across epochs is more likely to be correct. After training concludes, we compute these pseudo-labels and use them to retroactively approximate model performance on the target domain at each epoch.

With the exception of source loss, all these metrics are computed on the target domain. They are computationally inexpensive to evaluate and, taken individually, exhibit a reasonable correlation with target accuracy; however, as illustrated in Figure 3.3, each metric can fail in specific degenerate scenarios. To improve robustness, we build an ensemble of weak predictors by fitting a simple least-squares linear regressor to predict target performance from their combination. For each dataset, we randomly sample 200 configurations and train each of them while collecting 100 snapshots, resulting in 20,000 observations. Here, the term dataset refers to a complete multi-domain benchmark, such as Office31 or Office-Home, rather than to a single source or target domain within one adaptation task. At each snapshot, we compute the metrics described above and use them as input features for the regressor, after standardizing each feature to zero mean and unit variance. All metrics are used when EMS selects the best snapshot within a single run, while the pseudo-label score is excluded when EMS is used to compare different runs. Since target labels cannot be used to train this predictor on the UDA task being optimized, we learn the regressor on one experimental setting and then apply it to a different one. Specifically, for Office31 we use the regressor trained on Office-Home, and vice versa, while the regressor used for PACS is trained on Office31. Therefore, the target labels used to fit the regressor belong only to an auxiliary multi-domain dataset, distinct from the one on which the predictor is later applied. This does not violate the UDA constraint for the evaluated adaptation task, since labels from its target domain are never used for model selection. Importantly, while the regressor output prediction should not be interpreted as an absolute accuracy estimate on an unseen dataset, it remains strongly correlated with the true model ranking and outperforms the individual metrics taken alone, as shown in Table 3.1. This

result suggests that even a simple linear model trained on these unsupervised metrics can capture ranking information that transfers across multi-domain datasets. We exploit this predictor both to select the best configuration and to identify the best-performing training snapshot for each run.

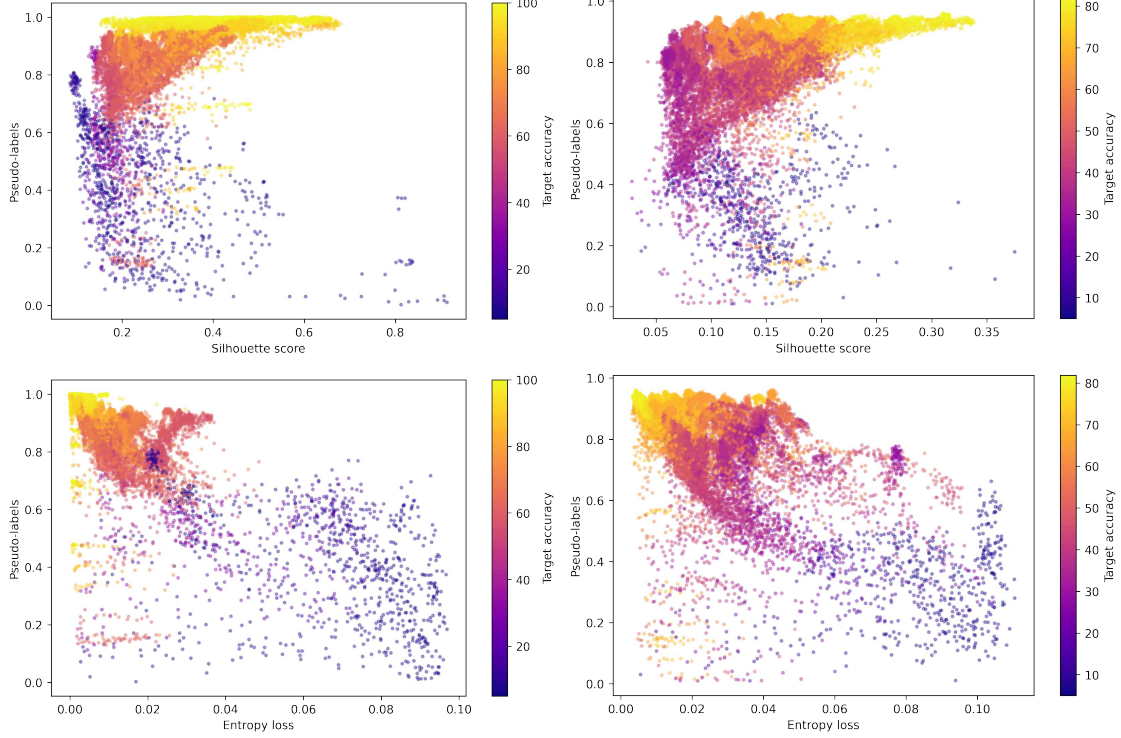


Figure 3.3: Model configurations trained on Office31 (left), and Office-Home (right). The two axes correspond to different unsupervised validation metrics used by EMS, while color encodes target-domain accuracy. A single metric alone does not provide a reliable prediction of model performance, while combining multiple metrics helps separate well-performing and poorly performing configurations. This highlights the complementarity of the validation metrics and their usefulness for estimating model quality. The similarity between corresponding plot pairs (first and second columns) suggests that an estimator learned on one dataset can transfer to another.

Table 3.1: Rank correlation between different metrics and target accuracy. EMS is evaluated in a cross-dataset setting, where the linear regressor is trained on a dataset different from the one reported in the table, and consistently outperforms the other individual performance estimators.

Metric	Average corr. with Target Acc.		
	Office31	Office-Home	PACS
Entropy	0.76	0.61	0.57
Diversity	0.76	0.77	0.76
Pseudo-Labels	0.71	0.61	0.68
Source Accuracy	0.42	0.20	0.48
Silhouette	0.70	0.72	0.74
Calinski-Harabasz	0.29	0.03	-0.13
EMS (ours)	0.87	0.81	0.80

3.3.4 Adversarial branch optimization

Given a fixed computational budget, ABAS executes multiple rounds that alternate between sampling candidate configurations and evaluating them (Algorithm 1). At each iteration, we draw B candidate configurations $\{\Theta_t^j\}_{j=1}^B$ by sampling the Bayesian acquisition function $\alpha(\Theta | D)$. Each configuration is then instantiated as an auxiliary adversarial branch for method Q , and the resulting network is trained under the target adaptation setting. Once training completes, we collect supervised information from the source and unsupervised signals from the target, and feed them to the model-selection module described in Section 3.3.3. The ensemble-based predictor then provides a performance estimate that is fed back to the BO routine to guide subsequent sampling. This loop is repeated for a predefined number of rounds, after which the best-performing model is returned. Despite its conceptual simplicity, the pipeline proves to be effective and, as shown in the next section, can yield substantial improvements over prior state-of-the-art baselines. Finally, although we adopt BOHB in our implementation, the overall framework is compatible with alternative search strategies.

Algorithm 1 Adversarial branch optimization

- 1: **Input:** Domain adaptation method Q , Performance estimator E , BO surrogate model $p(f|\Theta, D)$ and acquisition function $\alpha(\Theta|D)$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: Recommend $\{\Theta_t^j\}_{j=1}^B = \arg \max \alpha_{t-1}(\Theta|D)$
 - 4: **for** $j = 1$ **to** B **in parallel do**
 - 5: Build the auxiliary branch and evaluate $E(Q(\Theta_t^j))$ to obtain its corresponding performance metric f_t^j
 - 6: **end for**
 - 7: Update D and thus $p(f|\Theta, D)$ with $\{\Theta_t^j, f_t^j\}_{j=1}^B$
 - 8: **end for**
 - 9: **return** The best model according to $E(Q(\Theta^*))$
-

3.4 Experiments

3.4.1 Datasets

We evaluate our approach on three publicly available multi-domain benchmarks.

Office31 [106], a widely used dataset for domain adaptation, consists of 4,652 images covering 31 object categories across three domains: *Amazon* (A), *DSLR* (D) and *Webcam* (W). The DSLR and Webcam domains include photographs acquired in the same office setting using, respectively, a consumer reflex camera and a standard webcam, whereas the Amazon domain is composed of product images scraped from amazon.com.

Office-Home [129] contains approximately 15,500 images from 65 categories spanning four domains: *Art*, *Clipart*, *Product* and *Real-World*. Similarly to the Amazon domain from Office31, the Product domain contains product photos collected from online sources, while Real-World images were captured with a conventional digital camera.

PACS [60] comprises 9,991 images distributed across four domains, each including objects from seven classes. Although PACS is most often used for domain generalization, we adopt it here in a single-source setting to provide a diverse testbed.

3.4.2 Training protocol

ABAS operates through an outer BOHB loop, coupled with an inner training loop in which candidate networks are instantiated and trained. For BOHB, we use $\eta = 3$, with a minimum budget of 2000 iterations and a maximum budget of 6000 iterations, and we run the procedure for 24 iterations. Configurations that appear most promising under a smaller budget are then re-evaluated using larger budgets. For each sampled configuration, BOHB specifies both a computational budget and an architectural setting, which we use to construct the corresponding auxiliary branch for the selected adversarial UDA method.

Across all experiments, we adopt a ResNet-50 [43] pretrained on ImageNet-1k as the backbone. The adversarial branch, instantiated according to each sampled configuration, is attached immediately after the bottleneck layer that precedes the final classification layer (Figure 3.2). Models are trained end-to-end using the same implementation and protocol as [15]. More specifically, we employ SGD with batch size 36, momentum 0.9 and weight decay 5×10^{-4} . We adjust the learning rate

during training as

$$\mu_p = \frac{\mu_0}{(1 + \alpha \cdot p)^\beta}, \quad (3.5)$$

where $p \in [0; 1]$ is a real number linearly increasing during training, $\mu_0 = 10^{-3}$ is the initial learning rate, $\alpha = 10$ and $\beta = 0.75$. Following [33], we also adjust the weight ρ of the gradient reversal layer as

$$\rho_p = \frac{2}{1 + e^{-\gamma \cdot p}} - 1, \quad (3.6)$$

with $\gamma = 10$ in all of our experiments.

Because both the bottleneck and the auxiliary branch are initialized from scratch, we optimize these components with a larger learning rate ($10 \cdot \mu_p$) and a stronger weight decay (0.001) than those used for the pretrained backbone. At each iteration, we forward one source batch and one target batch through the backbone. The cross-entropy loss is evaluated on the source batch only, whereas the adversarial loss (DANN or ALDA loss depending on the setting) is computed using both source and target samples. After the model has been trained, we feed both source and target features to the performance estimator (Section 3.3.3), which produces the feedback signal used to guide the outer Bayesian optimization loop.

To train the regressor, we randomly sample 200 configurations per dataset and train each run while storing 100 intermediate snapshots, yielding 20,000 data points in total. For each snapshot, we compute the metrics described in Section 3.3.3, thereby constructing the training set for the regressor. Note that the pseudo-label-based metric is used only to select the best snapshot within a single training run, and not to rank configurations across different runs. All input features were normalized to have zero mean and unit variance.

We adopt a linear least-squares regressor, using the Scikit-Learn [94] implementation. In our cross-dataset setup, the regressor we apply to Office31 is trained on metrics produced with Office-Home and vice versa, while the predictor used for PACS is trained on Office31.

Unless otherwise specified, the baseline methods and their EMS-enhanced variants were evaluated over four independent runs, and the corresponding results are reported as mean accuracy and standard deviation. These repeated experiments provide an estimate of the variability associated with model training and show that the effect of EMS is observed across different runs. In contrast, due to the computational cost of repeating the complete nested optimization procedure, which requires training multiple candidate architectures within each BOHB search, the full ABAS pipeline was executed once per adaptation setting. Consequently, the reported ABAS results do not quantify the variability across independent search

runs and should be interpreted as the outcome of the corresponding single search execution. Similar limitations have been documented [63] as common practice in the NAS literature, where the computational cost of architecture search has often led to reporting the best architecture obtained from a single search run [6], [81]. Accordingly, variability across independent ABAS executions is not explicitly characterized in the present evaluation.

3.4.3 Search space sensitivity

Given the additional cost of automatically searching for an optimal design, it is essential to determine whether the auxiliary branch meaningfully affects final accuracy and whether the notion of a “good” branch is inherently data-dependent. To investigate these questions, we follow common NAS evaluation practice [141] and uniformly sample 40 configurations for each experimental setting to probe the search space. Our observations indicate that the branch design can critically influence optimization: it can effectively determine whether training succeeds or fails. Even after discarding a non-negligible fraction (10%) of runs that did not converge, the choice of adversarial-branch architecture explains, on average, at least a 20 percentage-point spread in absolute performance (Figure 3.4 and Table 3.2). Note that in these experiments λ was kept fixed to isolate architectural effects from hyperparameter variation. Moreover, Figure 3.5 shows that the optimal configuration is strongly setting-dependent. Specifically, the adversarial branch design introduced in ALDA performs well on Office31 and Office-Home, the benchmarks on which it was originally evaluated, but it falls markedly behind on several PACS adaptation scenarios (Table 3.5 and Figure 3.4).

3.4.4 Performance estimation

As noted in Section 3.2 and emphasized in [39], reliable model selection is a central component of transfer learning, and it becomes even more critical when a UDA framework explicitly aims to tune hyperparameters. In the following experiments, we investigate how strongly different proxy metrics correlate with true target accuracy. For this purpose, we sample 40 configurations per setting and compute the metrics introduced in Section 3.3.3 at every training epoch. Representative trends are reported in Table 3.1 and Figure 3.3. Individually, each metric acts as a weak proxy: it is generally correlated with target accuracy, yet it can break down in particular cases. In contrast, combining them through a linear model yields a substantially more reliable performance estimate. To evaluate our model-selection procedure independently of the NAS component, we re-trained the original ALDA

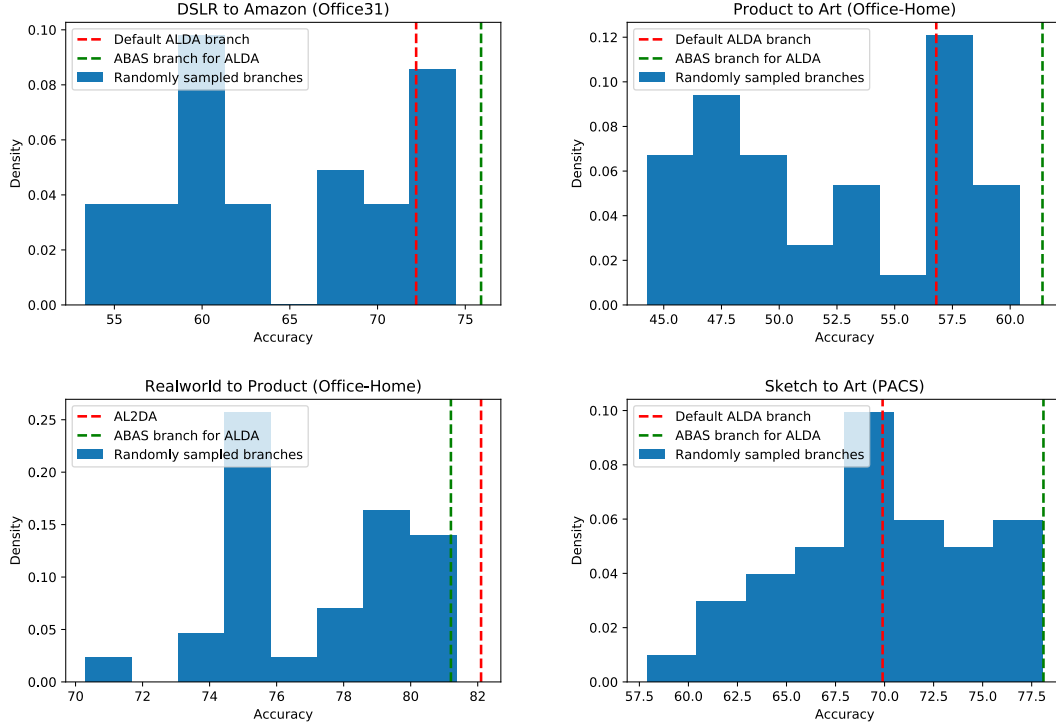


Figure 3.4: Accuracy distributions obtained by training 40 randomly sampled architectures for the ALDA auxiliary branch on three representative settings, excluding the 10% of runs that did not converge. Compared with many search spaces [141], the proposed one is sufficiently broad for architectural choices to produce a substantial variation in final performance, around 20% across all settings. The default ALDA branch performs well on Office31 and Office-Home, where it was originally evaluated, but degrades on PACS, while ABAS adapts to the different settings and consistently outperforms random sampling.

baseline and compared two reporting strategies: using the final epoch (as in the original paper) versus selecting the model via our estimator (Tables 3.3, 3.4 and 3.5). It is worth noting that a) despite using the official implementation we were unable to reproduce the exact numbers reported in [15] and b) on both Office-Home and PACS our model selection strategy improves the average performance by up to two percentage points.

3.4.5 Comparison with baseline methods

A comparison against the baselines is reported in Tables 3.3, 3.4 and 3.5. ABAS consistently improves over the underlying UDA methods, DANN and ALDA, even

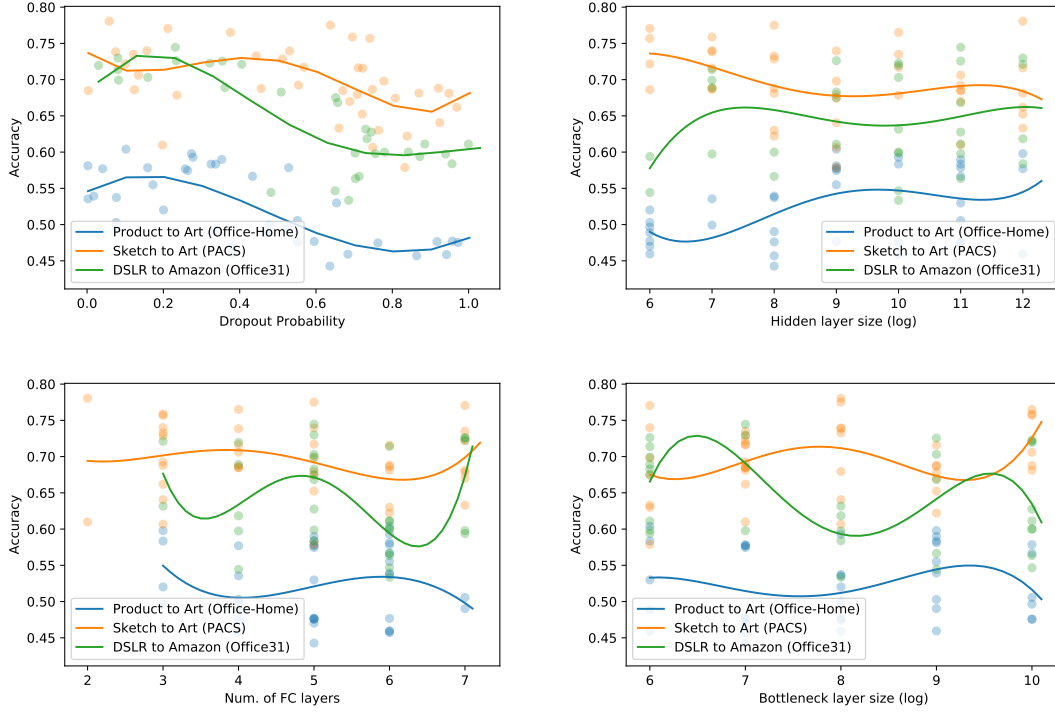


Figure 3.5: Sensitivity analysis of the ABAS search space. For each dataset, 40 configurations are randomly sampled and trained on one representative setting. The results show that the best hyperparameter values vary substantially across tasks. In each plot, the x-axis corresponds to the search-space hyperparameter under analysis.

when those baselines are paired with our model selection strategy, and this holds across all three datasets. The smallest average gain is 1.9% (Office31: ALDA 87.5% vs. ABAS-ALDA 89.4%), while the largest reaches 3.1% (Office-Home: DANN 56.0% vs. ABAS-DANN 59.1%). With the exception of Office-Home, where we were unable to reproduce the official ALDA results, ABAS-ALDA achieves the best performance among compared methods in all settings. One caveat emerges on the relatively easy Office31 $D \rightarrow W$ transfer, where ABAS-ALDA is surpassed by a few methods. We conjecture that, since DSLR and Webcam domains are highly similar, the adversarial branch may have limited meaningful discrepancy to exploit; notably, ALDA itself also performs weakly on this particular setting.

Table 3.2: A set of configurations was randomly sampled from the ABAS search space and trained with ALDA on Office31 [106] (reported in the first six rows), and Office-Home [129] settings. The results highlight the strong impact of the auxiliary branch configuration on final performance. Div. denotes the fraction of runs that did not converge, Min and Max report the minimum and maximum accuracy, Mean gives the average accuracy with standard deviation, and Range is computed as Max - Min.

Setting	Div.	Min	Max	Mean $\pm$ Std.	Range
A-W	0.23	64.59	95.79	83.4 ± 9.8	31.19
A-D	0.28	72.49	91.57	84.4 ± 6.2	19.08
D-W	0.15	17.11	99.06	89.0 ± 21.7	81.95
D-A	0.28	46.59	70.15	59.3 ± 6.4	23.55
W-D	0.22	16.47	100.00	89.3 ± 24.2	83.53
W-A	0.17	34.97	74.55	63.8 ± 8.3	39.58
Ar-Cl	0.08	26.60	50.71	41.7 ± 6.4	24.11
Ar-Pr	0.14	59.12	65.14	63.2 ± 2.1	6.01
Ar-Rw	0.22	43.30	75.89	70.0 ± 6.3	32.59
Cl-Ar	0.14	36.55	57.87	51.6 ± 7.3	21.32
Cl-Pr	0.14	55.58	66.41	60.9 ± 3.9	10.84
Cl-Rw	0.15	46.12	70.25	63.2 ± 5.6	24.13
Pr-Ar	0.14	40.15	57.21	49.8 ± 4.9	17.06
Pr-Cl	0.11	26.27	49.52	39.7 ± 6.5	23.25
Pr-Rw	0.07	41.47	75.64	66.5 ± 10.7	34.16
Rw-Ar	0.07	59.81	67.86	64.7 ± 2.3	8.06
Rw-Cl	0.07	15.40	54.80	46.2 ± 7.6	39.40
Rw-Pr	0.14	70.30	81.36	77.1 ± 2.6	11.06
AVG	0.2	42.9	72.4	64.6 ± 7.9	29.49

Table 3.3: ABAS results on Office31 using a ResNet-50 backbone. “+ EMS” denotes experiments evaluated with the proposed model selection strategy. For the baselines and their EMS variants, we report the mean and standard deviation over four runs.

	A-W	A-D	D-W	D-A	W-D	W-A	AVG
ResNet-50 [43]	68.4 $\pm$ 0.2	68.9 $\pm$ 0.2	96.7 $\pm$ 0.1	62.5 $\pm$ 0.3	99.3 $\pm$ 0.1	60.7 $\pm$ 0.3	76.1
DANN [33]	82.0 $\pm$ 0.4	79.7 $\pm$ 0.4	96.9 $\pm$ 0.2	68.2 $\pm$ 0.4	99.1 $\pm$ 0.1	67.4 $\pm$ 0.5	82.2
ADDA [127]	86.2 $\pm$ 0.5	77.8 $\pm$ 0.3	96.2 $\pm$ 0.3	69.5 $\pm$ 0.4	98.4 $\pm$ 0.3	68.9 $\pm$ 0.5	82.9
JAN [76]	85.4 $\pm$ 0.3	84.7 $\pm$ 0.3	97.4 $\pm$ 0.2	68.6 $\pm$ 0.3	99.8 $\pm$ 0.2	70.0 $\pm$ 0.4	84.3
MADA [95]	90.0 $\pm$ 0.1	87.8 $\pm$ 0.2	97.4 $\pm$ 0.1	70.3 $\pm$ 0.3	99.6 $\pm$ 0.1	66.4 $\pm$ 0.3	85.2
CBST [155]	87.8 $\pm$ 0.8	86.5 $\pm$ 1.0	98.5 $\pm$ 0.1	71.2 $\pm$ 0.4	100.0 $\pm$ 0.0	70.9 $\pm$ 0.7	85.8
CAN [148]	92.5	90.1	98.8	72.1	100.0	69.9	87.2
CDAN+E [75]	94.1 $\pm$ 0.1	92.9 $\pm$ 0.2	98.6 $\pm$ 0.1	71.0 $\pm$ 0.3	100.0 $\pm$ 0.0	69.3 $\pm$ 0.3	87.7
MCS [68]	75.1	71.9	96.7	58.8	99.4	57.2	76.5
DMRL [137]	90.8 $\pm$ 0.3	93.4 $\pm$ 0.5	99.0 $\pm$ 0.2	73.0 $\pm$ 0.3	100.0 $\pm$ 0.0	71.2 $\pm$ 0.3	87.9
DM-ADA [140]	83.9 $\pm$ 0.4	77.5 $\pm$ 0.2	99.8 $\pm$ 0.1	64.6 $\pm$ 0.4	99.9 $\pm$ 0.1	64.0 $\pm$ 0.5	81.6
3CATN [62]	95.3 $\pm$ 0.2	94.1 $\pm$ 0.3	99.3 $\pm$ 0.5	73.1 $\pm$ 0.2	100 $\pm$ 0.0	71.5 $\pm$ 0.6	88.9
ALDA [15]	95.6 $\pm$ 0.5	94.0 $\pm$ 0.4	97.7 $\pm$ 0.1	72.2 $\pm$ 0.4	100.0 $\pm$ 0.0	72.5 $\pm$ 0.2	88.7
ResNet-50	67.0 $\pm$ 1.9	76.4 $\pm$ 1.4	94.8 $\pm$ 1.3	56.2 $\pm$ 1.0	98.7 $\pm$ 0.1	58.3 $\pm$ 0.9	75.2
ResNet-50 + EMS	70.4 $\pm$ 2.0	77.1 $\pm$ 2.0	95.6 $\pm$ 0.6	58.2 $\pm$ 2.6	99.1 $\pm$ 0.5	60.1 $\pm$ 1.1	76.8
DANN	85.0 $\pm$ 0.5	82.5 $\pm$ 0.5	96.7 $\pm$ 0.2	63.9 $\pm$ 1.1	99.2 $\pm$ 0.3	64.7 $\pm$ 0.7	82.0
DANN + EMS	85.5 $\pm$ 2.1	82.8 $\pm$ 2.2	97.1 $\pm$ 1.0	64.1 $\pm$ 0.5	99.8 $\pm$ 0.2	64.5 $\pm$ 2.0	82.3
ABAS-DANN	89.4	87.6	98.4	64.1	99.8	69.3	84.8
ALDA	94.8 $\pm$ 0.6	91.6 $\pm$ 0.9	98.3 $\pm$ 0.4	69.6 $\pm$ 0.9	99.9 $\pm$ 0.0	70.8 $\pm$ 0.7	87.5
ALDA + EMS	95.0 $\pm$ 1.4	91.9 $\pm$ 0.7	98.6 $\pm$ 0.4	69.3 $\pm$ 2.1	99.7 $\pm$ 0.3	71.1 $\pm$ 0.8	87.6
ABAS-ALDA	96.1	95.0	98.5	75.9	100.0	70.7	89.4

Table 3.4: ABAS results on Office-Home with a ResNet-50 backbone. Official ALDA results could not be reproduced with the released code. “+ EMS” denotes runs evaluated with the proposed model selection strategy.

	Ar-Cl	Ar-Pr	Ar-Rw	Cl-Ar	Cl-Pr	Cl-Rw	Pr-Ar	Pr-Cl	Pr-Rw	Rw-Ar	Rw-Cl	Rw-Pr	AVG
ResNet-50	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DANN	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN+E	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
TAT	51.6	69.5	75.4	59.4	69.5	68.6	59.5	50.5	76.8	70.9	56.6	81.6	65.8
ALDA	53.7	70.1	76.4	60.2	72.6	71.5	56.8	51.9	77.1	70.2	56.3	82.1	66.6
ResNet-50	35.6	61.5	70.4	44.0	56.1	57.4	46.5	31.9	69.1	62.4	38.0	75.6	54.1
ResNet-50 + EMS	36.5	63.1	71.7	45.8	57.5	59.3	47.9	32.3	69.9	63.5	37.3	76.0	55.1
DANN	39.8	58.0	68.1	48.6	57.0	59.9	46.9	38.4	68.8	63.2	47.7	75.9	56.0
DANN + EMS	40.3	60.4	70.1	49.6	57.5	60.5	47.9	38.3	69.1	63.7	47.7	76.8	56.8
ABAS-DANN	44.9	61.1	71.2	52.7	60.4	62.5	50.1	43.1	70.0	65.4	50.9	77.1	59.1
ALDA	46.4	68.6	74.6	57.6	67.0	69.4	57.2	46.3	75.6	69.2	53.3	80.8	63.8
ALDA + EMS	47.5	70.1	75.2	58.8	67.5	69.7	58.1	46.6	76.2	69.9	54.0	81.0	64.6
ABAS-ALDA	51.5	71.7	75.5	59.8	69.4	69.5	59.8	47.1	77.7	70.6	55.2	80.2	65.7

Table 3.5: ABAS results on PACS with a ResNet-50 backbone, “+ EMS” denotes runs evaluated with the proposed model selection strategy.

	P-A	C-A	S-A	A-P	C-P	S-P	A-C	S-C	P-C	A-S	C-S	P-S	AVG
ResNet-50	62.3	71.2	30.8	96.6	89.7	38.0	57.0	53.5	27.6	43.0	59.1	30.7	55.0
ResNet-50 + EMS	62.0	72.0	25.6	97.7	90.9	38.8	57.5	53.2	23.0	41.3	58.7	28.2	54.1
DANN [33]	79.9	88.4	70.3	97.3	95.1	57.1	82.4	70.8	64.7	62.5	68.7	61.4	74.9
DANN + EMS	80.3	89.0	72.6	98.1	95.6	56.2	82.2	73.3	64.0	63.4	69.7	59.6	75.3
ABAS-DANN	91.7	90.7	68.2	98.3	96.2	82.8	87.2	71.2	73.7	68.7	76.0	47.2	82.9
ALDA	89.3	91.9	69.9	98.3	97.3	63.4	85.1	75.2	74.3	79.2	70.6	60.7	79.6
ALDA + EMS	90.2	92.0	72.3	98.4	97.8	69.5	86.0	82.1	72.1	80.7	75.1	66.1	81.9
ABAS-ALDA	93.1	91.8	78.1	98.7	97.8	70.8	88.7	84.9	69.7	79.8	69.5	64.9	82.3

3.5 Conclusions

This work addresses two key prerequisites for applying NAS to UDA: it introduces a data-driven model-selection mechanism (EMS) that bypasses the absence of target labels, and it concentrates the search on auxiliary branches attached to a pretrained backbone, a practice that is crucial for achieving strong performance in realistic UDA pipelines. In previous work, the community explicitly underscored how central robust model selection is for UDA [39]. Because no validation set is identically distributed to the target test data, it has been common to tune hyperparameters directly on the test set. We hope that the solutions explored here, most notably EMS, represent early steps toward more methodologically sound model-selection practices.

It should be noted that EMS assumes the availability of an auxiliary labeled multi-domain dataset to train the performance estimator. Although no target labels from the adaptation task under evaluation are used, its applicability depends on the availability of a sufficiently related external benchmark. Moreover, the present evaluation considers a relatively constrained search space, in which a single auxiliary branch is attached at a fixed point of the backbone. This design preserves the pretrained backbone and keeps the optimization problem tractable, but broader search spaces could also be explored, including multiple auxiliary branches and the joint optimization of their attachment points.

Nevertheless, ABAS provides a set of reusable components: search space design, EMS, and branch optimization. These components are compatible with many NAS strategies and, more broadly, with UDA approaches that rely on an auxiliary branch. A particularly relevant case is represented by self-supervision based UDA methods that incorporate an auxiliary branch: since they adopt the same overall macro-architecture shown in Figure 3.2, ABAS could be applied to them with minimal modifications.

Chapter 4

Training-free NAS for tiny and hybrid architectures

The content, images, and tables included in this chapter were originally presented in the following publications:

- N. Cavagnero, L. Robbiano, B. Caputo, and G. Averta, *FreeREA: Training-free evolution-based architecture search*, in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 1493–1502
- N. Cavagnero, L. Robbiano, F. Pistilli, B. Caputo, and G. Averta, *Entropic Score metric: Decoupling topology and size in training-free nas*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2023, pp. 1459–1468

4.1 Introduction and motivation

In recent years, machine learning has rapidly expanded its impact across a broad range of domains, including education [97], industry [57], and healthcare [98]. As highlighted in Chapter 1, however, this progress comes with a growing concern: the substantial energy demand of modern training pipelines and associated environmental footprint [22], [80], [93], [114]. For instance, recent analyses [86] have shown that the carbon emission produced by training a single contemporary Large Language Model (LLM) can easily exceed tens of times those associated with the power consumption of a typical household in one year. When Neural Architecture

Search (NAS) is employed to identify high-performance models, this cost can increase even further, often by one order of magnitude or more, due to the repeated evaluation of multiple candidate architectures.

These observations motivate the development of methods that reduce the computational and energy burden of model design, particularly as research increasingly targets deployment on resource-constrained devices. In such settings, the architectural choices must explicitly account for limited compute and memory budgets, rather than prioritizing accuracy alone. Within this perspective, NAS [5], [7], [122] offers a principled alternative to manual trial-and-error, enabling the automatic discovery of models that make efficient use of available resources while limiting the need for extensive human expertise.

Conventional NAS methods, however, typically rely on training and evaluating a large number of candidate architectures during the search in order to estimate their performance. As a result, NAS pipelines often entail substantial computational time and resource requirements, which makes them difficult to adopt for many practitioners who lack access to large-scale or specialized computing infrastructure. Moreover, the energy cost of standard training-based NAS can be so high that it may effectively undermine the very motivation for designing tiny, efficient models, offsetting the expected sustainability benefits. For these reasons, enabling the large-scale development of lightweight neural networks calls for NAS methods that can identify promising architectures without fully training every candidate. In response, recent research has increasingly explored training-free proxies that estimate quality of architectures directly at initialization [1], [85], thereby bypassing the costly training phase and delivering substantial speedups even on modest hardware. Nevertheless, existing training-free NAS approaches remain, in general, less competitive than training-based alternatives, and they often overlook hardware-oriented constraints, such as model size and FLOPs, that are crucial when targeting deployment under strict resource budgets.

In this chapter, we investigate training-free NAS as a practical strategy to reduce the cost of architecture optimization while preserving its ability to identify accurate and resource-efficient models. The chapter is organized around two closely connected contributions. First, we introduce FreeREA, an evolution-based NAS method that replaces training-based candidate evaluation with a combination of training-free proxy metrics. Rather than relying on a single score, FreeREA combines signals related to network expressivity and trainability, which are embedded within a regularized evolution strategy. This allows the search to identify competitive architectures in only a few minutes, both in standard unconstrained benchmarks and in scenarios where explicit limits on model size and computational cost are imposed.

The second part of the chapter builds on the limitations of the aggregate scoring strategy used by FreeREA. While combining multiple proxies can improve robustness, different metrics do not necessarily evaluate the same architectural properties. In particular, topology and model size represent distinct design dimensions, and treating them as components of a single global fitness function may obscure the role played by each proxy. We therefore introduce Entropic Score, a data-agnostic training-free metric designed to capture topology-related properties through the entropy of network activations at initialization. Entropic Score is then paired with LogSynflow in a decoupled search strategy, where the former guides topological choices and the latter supervises dimensional ones. This formulation extends training-free NAS from small cell-based benchmarks to the design of tiny hybrid CNN-ViT architectures for ImageNet-scale classification, showing that training-free estimates can be used more effectively when they are assigned to the architectural dimensions for which they are most informative.

Overall, the chapter presents a progression from metric aggregation to metric specialization. Section 4.3 shows that training-free proxies can provide a viable replacement for repeated model training in evolutionary NAS, while Section 4.4 demonstrates that a more structured use of such proxies can further improve the search when architectures become more heterogeneous and deployment-oriented.

4.2 Related works

Following the earliest NAS studies, which were notoriously computationally demanding (e.g., the seminal work introducing NAS [153] reportedly [154] required more than 530,000 GPU hours), the research community progressively shifted its attention toward the development of more efficient search strategies.

Early efforts toward improving NAS efficiency largely revolved around parameter sharing [28], [69], [96], where newly sampled architectures reuse weights learned by previously explored candidates. While [96] still relied on reinforcement learning, the literature soon consolidated around two additional search paradigms: differentiable search and evolutionary methods. Differentiable approaches [28], [69] reformulate the search process so that it becomes end-to-end differentiable and can be optimized via gradient-based techniques, yielding substantial speedups compared to the original NAS formulations. In contrast, evolution-based strategies are typically straightforward to implement and naturally support weight inheritance from parent architectures; however, early variants often reported inferior performance relative to competing approaches [72], [101]. In REA [100], the authors introduced

a regularized tournament selection strategy, which was among the first evolutionary NAS methods shown to surpass manually designed architectures. Nevertheless, this line of work did not fully address the fundamental bottleneck of NAS, as it still required training and evaluating thousands of candidate models during the search.

Motivated by these limitations, recent work has increasingly investigated training-free scoring metrics that evaluate candidate architectures directly at initialization, thereby bypassing the costly training and validation steps that represent the primary bottleneck of most NAS pipelines. An early example is NASWOT [85], which aims to quantify the expressive capacity of the architecture and can identify promising candidates within seconds. Despite this efficiency, the architectures selected solely through this metric typically remained less accurate than those obtained via conventional training-based NAS. Consequently, this line of work was often adopted mainly as a warm-start or initialization strategy for subsequent, more traditional search procedures.

Subsequent works built on this idea in different ways. GA-NINASWOT [135] combined the NASWOT score with a genetic algorithm, while EPE-NAS [77] introduced refinements over the original metric formulation. Nonetheless, these approaches generally remained unable to match the effectiveness of training-based NAS. Building on the intuition behind NASWOT, other studies [16] proposed to complement a variant of the NASWOT criterion with the Neural Tangent Kernel (NTK) metric [53], in order to account not only for expressivity but also for the trainability of candidate architectures during the search.

4.3 FreeREA: combining training-free proxies for evolutionary NAS

In this first work, we introduce FreeREA, a constrained evolutionary method that leverages an optimized combination of training-free metrics as proxies for model accuracy. Our results show that these metrics can effectively replace full training during the search phase, enabling the automatic identification of high-performing architectures in only a few minutes. The method is evaluated on two widely adopted NAS benchmarks, where it surpasses state-of-the-art approaches in the unconstrained setting. In addition, we extend the analysis to constrained search, providing, to the best of our knowledge, the first evaluation of a training-free NAS method in this scenario and establishing a strong baseline for future work.

In summary, this work advances the state of the art through three main contributions. First, it introduces an optimized combination of training-free metrics for

architecture ranking, which can fully replace the training phase in NAS algorithms. Second, it proposes an improved evolutionary search algorithm that effectively exploits this ranking strategy to identify highly accurate models within a few minutes. Third, it provides an extensive experimental evaluation showing that training-free methods can serve as a viable alternative to training-based approaches, including in hardware-constrained settings.

4.3.1 Metrics

Table 4.1: Kendall- τ and Spearman- ρ correlations between training-free metrics and test accuracy on the three datasets considered in NATS-Bench [26]. Each metric is evaluated over three different initializations, and the resulting scores are averaged before computing the correlations.

	CIFAR-10		CIFAR-100		ImageNet-16-120	
Metric	τ	ρ	τ	ρ	τ	ρ
NTK [53]	-0.33	-0.49	-0.30	-0.45	-0.39	-0.56
SNIP [58]	0.45	0.61	0.47	0.62	0.41	0.55
Fisher [125]	0.39	0.54	0.40	0.55	0.36	0.48
Grasp [131]	0.28	0.41	0.35	0.50	0.35	0.49
PathNorm [54]	0.41	0.59	0.42	0.60	0.45	0.63
NASWOT [85]	0.61	0.79	0.62	0.81	0.60	0.78
Synflow [123]	0.57	0.77	0.56	0.76	0.56	0.75
LogSynflow (ours)	0.63	0.83	0.61	0.81	0.60	0.79

Training-free NAS methods depend on scoring functions that estimate expected performance of architectures without performing full training. Because no single metric perfectly aligns with final test accuracy, robust model ranking can benefit from combining several complementary signals, for instance those reflecting both trainability and expressivity. Following this rationale, we ran an initial study on NATS-Bench to quantify how well a set of candidate metrics can serve as surrogates for test accuracy. The resulting correlations, summarized in Table 4.1, indicate that the most informative indicators are LogSynflow (our logarithmic variant of Synflow [123]) and NASWOT [85]. Finally, we augment these scores with the number of skipped layers, which biases the search toward architectures that are easier to optimize.

To aggregate the different metrics, we compute the fitness of a candidate architecture as the sum of the normalized scores of each metric. This choice is motivated by the observation that architectures that are simultaneously easy to optimize and

sufficiently expressive are more likely to achieve strong accuracy [28]. Accordingly, in our implementation, the fitness value J for an architecture Θ is defined as:

$$J(\Theta) = \frac{\mathcal{S}(\Theta)}{\max_{\Theta' \in H} \mathcal{S}(\Theta')} + \frac{s(\Theta)}{\max_{\Theta' \in H} s(\Theta')} + \frac{\text{Skip}(\Theta)}{\max_{\Theta' \in H} \text{Skip}(\Theta')}, \quad (4.1)$$

where $\mathcal{S}(\cdot)$, $s(\cdot)$ and $\text{skip}(\cdot)$ are the LogSynflow, NASWOT and Skipped Layers metrics, respectively, and H is the set of all the explored candidate architectures. An ablation study assessing the contribution of the three terms in the fitness function is reported in Table 4.2. To further improve robustness, we compute both NASWOT and LogSynflow three times under different random initializations, and use the average across the three runs as the final score. Implementation details for the selected metrics are discussed below.

Table 4.2: Ablation study of the metrics included in the fitness function. The table reports the test accuracy of the final selected model. Each configuration is evaluated over 30 runs, with a search budget of 45 seconds on NATS-Bench and 12 minutes on NAS-Bench-101. The two metrics introduced in this work, Skipped Layers and LogSynflow, appear to provide the strongest contribution, while NASWOT has a noticeable effect mainly on NAS-Bench-101, likely because its discriminative ability is more effective in larger search spaces.

	NATS-Bench			NAS-Bench-101
	CIFAR-10	CIFAR-100	ImageNet-16-120	CIFAR-10
Baseline	94.36 $\pm$ 0.00	73.51 $\pm$ 0.00	46.34 $\pm$ 0.00	93.80 $\pm$ 0.02
w/o NASWOT	94.36 $\pm$ 0.00	73.51 $\pm$ 0.00	46.34 $\pm$ 0.00	93.55 $\pm$ 0.49
w/o Skipped Layers	93.76 $\pm$ 0.00	71.11 $\pm$ 0.00	41.44 $\pm$ 0.00	91.90 $\pm$ 1.69
w/o LogSynflow	94.30 $\pm$ 0.00	71.13 $\pm$ 0.00	44.48 $\pm$ 0.00	92.62 $\pm$ 0.08

NASWOT

NASWOT was proposed in [85] as a training-free indicator of architectural expressivity at initialization. In a ReLU network, forwarding an input sample produces activation patterns that split each layer’s outputs into active units (positive activations) and inactive units (non-positive activations), yielding corresponding binary masks. For an input space of fixed dimensionality, the number of distinct activation patterns, and thus the number of piecewise-linear regions induced by the network, serves as a proxy for the model’s ability to separate and discriminate different inputs. When two distinct inputs produce highly similar activation masks, the network has limited capacity to tell them apart. Conversely, a more expressive architecture tends to map even nearby points in the input space to different

activation patterns. To quantify this behavior, we follow the procedure in [85]: we construct a mini-batch of 64 samples, forward it through the network, and record the resulting binary masks. We then quantify the dissimilarity between activation patterns using the Hamming distance. These pairwise distances are used to construct a kernel matrix K_H , from which the final score is computed as:

$$s = \log|\det(K_H)|. \quad (4.2)$$

Larger values of s indicate higher network expressivity and, consequently, a stronger ability to separate the input samples. As shown in Figure 4.1, architectures achieving higher NASWOT scores generally attain better test accuracy. This trend is also reflected by the positive correlation reported in Table 4.1. Accordingly, optimizing the NASWOT metric biases the search toward more expressive architectures and, in turn, models that are likely to achieve higher test accuracy.

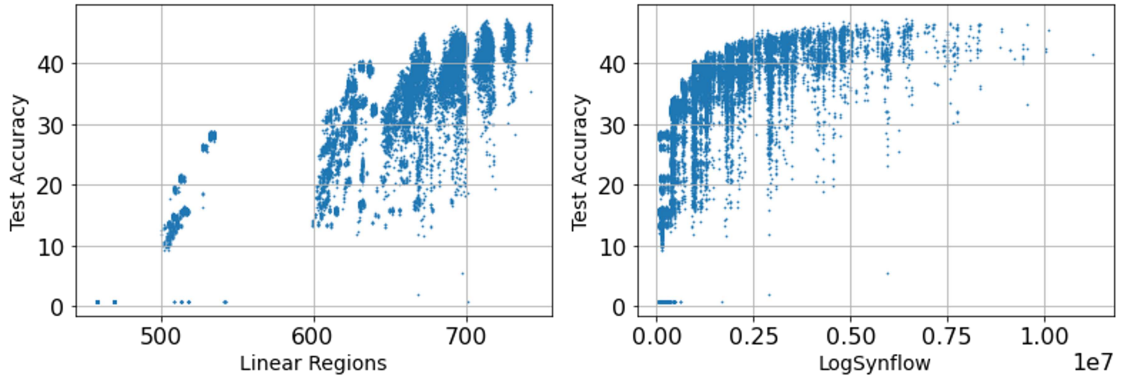


Figure 4.1: Relationship between test accuracy and the two training-free metrics used in the method, computed over all architectures in NATS-Bench: NASWOT on the left panel and LogSynflow on the right. Accuracy values refer to the ImageNet-16-120 dataset.

LogSynflow

Synflow was originally proposed in [123] as a pruning criterion, aimed at identifying which weights can be removed to compress a network while preserving its performance as much as possible. In its original formulation, Synflow provides a local measure of the importance of individual parameters. The idea was later extended by [1] to obtain an architecture-level score by summing the contributions of all weights. In its global form, the metric can be expressed as the inner product

Table 4.3: Kendall- τ and Spearman- ρ correlations between Synflow, LogSynflow, and test accuracy on NAS-Bench-101 [143]. The values show that LogSynflow provides a more reliable proxy for test accuracy than the original Synflow metric.

Metric	Kendall	Spearman
Synflow [123]	0.25	0.37
LogSynflow (ours)	0.31	0.45

between the parameter vector $\boldsymbol{\theta}$ and the gradient vector $\frac{\partial \mathcal{R}}{\partial \boldsymbol{\theta}}$:

$$\mathcal{S}(\boldsymbol{\Theta}) = \boldsymbol{\theta} \cdot \frac{\partial \mathcal{R}}{\partial \boldsymbol{\theta}}. \quad (4.3)$$

To evaluate the metric, the model is first re-parameterized by taking the absolute value of its weights, ensuring $\theta_i \geq 0$. An all-ones input tensor is then propagated through the network, and the resulting scalar output is backpropagated to compute the gradients. It must be noted that normalization layers must be disabled when computing Synflow [123]. In practice, the removal of batch normalization often leads to gradient explosion even for comparatively small models. As a result, the contribution of the weight magnitudes becomes effectively negligible, since the gradient-related term can dominate the product by several orders of magnitude. To mitigate this effect, we introduce LogSynflow, which applies a logarithmic transformation to the gradients to reduce their dynamic range before aggregating the per-weight contribution across the network:

$$\mathcal{S}(\boldsymbol{\Theta}) = \boldsymbol{\theta} \cdot \log \left(\frac{\partial \mathcal{R}}{\partial \boldsymbol{\theta}} + 1 \right). \quad (4.4)$$

To assess whether this modification yields a more informative indicator, we evaluated both metrics across both benchmarks considered in this work [26], [143] and measured their correlation with test accuracy (Tables 4.1, 4.3). The results show that LogSynflow consistently achieves a stronger correlation with test accuracy than the original Synflow formulation. Moreover, the positive correlation between LogSynflow and test accuracy, also qualitatively shown in Figure 4.1, indicates that favoring architectures with higher LogSynflow scores tends to select models that perform better at test time.

Skipped layers

Several training-free NAS approaches [16], [25], [53], [111] rely on the Neural Tangent Kernel (NTK) [53] as a base for indicators of model trainability. In our

setting, however, NTK is substantially more expensive to compute than alternative training-free metrics [58], [85], [131] and exhibits only a weak correlation with test accuracy (Table 4.1). To bias the search toward architectures that are easier to optimize, we therefore replace NTK with a simpler surrogate, defined as the number of skipped layers normalized by the total number of skip connections in a cell

$$s = \frac{\text{\#Skipped layers}}{\text{\#Skip connections}}. \quad (4.5)$$

Since the introduction of residual connections in [43], which has profoundly influenced modern deep learning, skip connections have become a standard design choice in state-of-the-art architectures, including EfficientNet [121], ResNeXt [139], ViT [29] and ConvNeXt [73]. These connections are widely recognized as a key enabler for training very deep networks: by providing shorter gradient paths across computational blocks, even a single bypass around a cell helps alleviate vanishing gradients and facilitates the backward propagation of the gradient through the model. The intuition behind this additional metric is to prioritize configurations that use a small number of skip connections while spanning many layers, thus promoting long-range shortcuts within a cell. Under this definition, the score is maximized in the extreme case where a single skip connection directly links the cell input to its output.

4.3.2 Search strategy

To identify the best-performing architecture, we adopt an evolutionary strategy based on an enhanced version of Tournament Selection with Ageing (REA) [100]. The procedure starts from a randomly generated population of N individuals, each represented by a genotype encoding the network structure. At each iteration, n candidates are uniformly sampled from the current population, and the highest-scoring individual in this subset is chosen as the parent. The selected genotype is then mutated to produce an offspring, which is inserted back into the population. Classic Tournament Selection maintains a fixed population size by discarding the least fit individuals, while REA instead eliminates the oldest candidate, thereby favoring younger solutions and promoting broader exploration of the search space. To further strengthen search dynamics, we adopt a variant in which two parents are selected at each iteration. Each parent is independently mutated to produce two offspring, and an additional child is generated via crossover. The crossover operation is implemented by uniformly selecting each gene from one of the two parents. This design improves exploration, since mutation typically induces local changes, whereas recombining two distinct genotypes can yield more diverse offspring that diverge substantially from their parents [72]. The benefits of this strategy over the original algorithm are demonstrated empirically in our experimental evaluation. To

keep the population size fixed, after removing the oldest individual we retain only the best N candidates. In both mutation and crossover, all gene selections are drawn uniformly at random.

We perform constrained search by enforcing upper bounds on FLOPs and on the number of parameters for candidate architectures, since these quantities do not depend on a specific deployment platform. When constraints are active, only feasible individuals are admitted into the population; if an offspring violates the limits, new offspring are generated until a feasible child is obtained. Unless stated otherwise, all experiments in this work use $N = 25$ and $n = 5$. Additional results for alternative choices of N and n are provided in Table 4.5.

4.3.3 Experiments

Benchmarks and experimental protocol

To enable a fair and reproducible comparison with prior NAS methods while keeping the computational budget negligible, we follow the approach described in Section 2.2.4 and conduct our experiments on tabular NAS benchmarks that provide precomputed training and evaluation results. By querying these benchmarks rather than retraining candidates, we substantially speed up the experimental evaluation and remove variability due to different training pipelines, therefore improving comparability across methods. More specifically, we use NAS-Bench-101, which enumerates a large set of cell-based CNN architectures evaluated on CIFAR-10 [56], and NATS-Bench [26], a tabular benchmark derived from NAS-Bench-201 [27], that reports results for a compact cell search space on CIFAR-10, CIFAR-100 and ImageNet-16-120 [19]. NATS-Bench also provides auxiliary efficiency indicators (e.g. FLOPs and latency), which we exploit when considering constrained settings.

Given that constrained NAS methods have not been systematically benchmarked on these tabular search spaces, we first assess FreeREA in an unconstrained setting, comparing it with representative state-of-the-art training-based and training-free approaches on both NAS-Bench-101 and NATS-Bench. We then move to a constrained evaluation on NATS-Bench, where we measure how close the architectures found by FreeREA are to the corresponding optimum under the same resource limits. FreeREA is run with a wall-clock budget of 45 seconds on NATS-Bench and 12 minutes on NAS-Bench-101. All experiments are carried out on a single NVIDIA RTX 3080 GPU. Results for training-based baselines on NATS-Bench are taken from the original benchmark paper [26], while training-free baselines are reported as in their respective publications.

Unconstrained setting

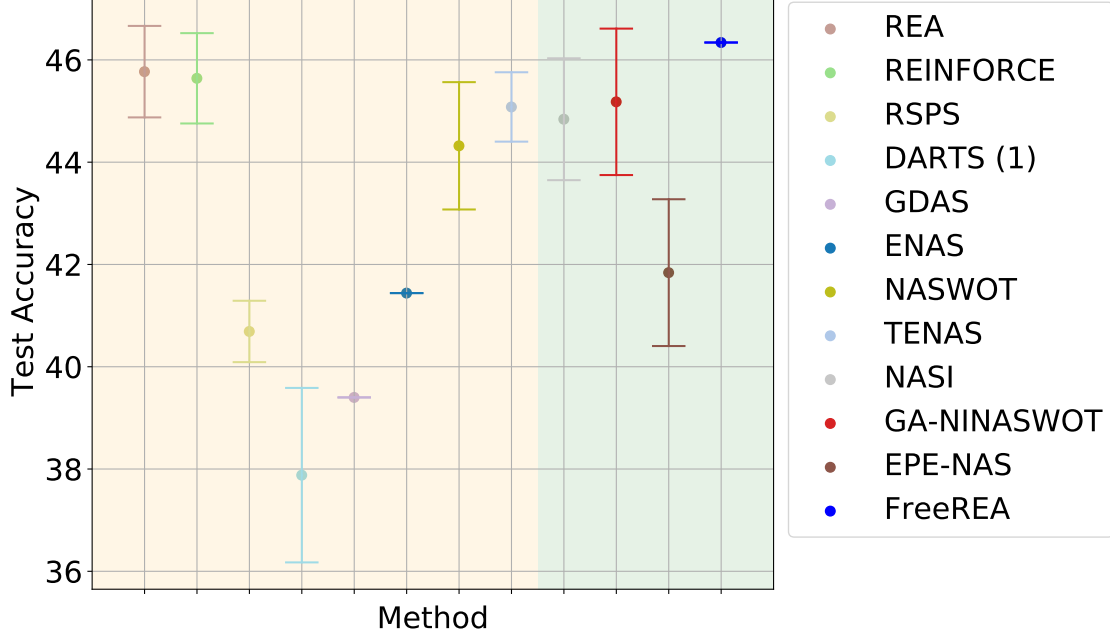


Figure 4.2: Mean test accuracy and corresponding standard deviation on the ImageNet-16-120 dataset of NATS-Bench. The shaded regions group training-based methods and training-free methods separately.

As an initial unconstrained study, we focus on NATS-Bench. The results in Table 4.4 and Figure 4.2 confirm that, among the considered state-of-the-art baselines, REA [100] is the strongest competitor in terms of the accuracy of the architectures it discovers. Nevertheless, FreeREA consistently improves upon this reference across all the datasets included in the benchmark, with the advantage becoming more pronounced as the task becomes more challenging. Concretely, FreeREA identifies models achieving 94.36% test accuracy on CIFAR-10, 73.51% on CIFAR-100, and 46.34% on ImageNet-16-120. These solutions are essentially optimal: they are within 0.01% of the optimum on CIFAR-10, match the optimum on CIFAR-100, and remain only 0.97% below the optimum on ImageNet-16-120.

Overall, FreeREA delivers markedly stronger results than all baselines considered here, including both training-free and training-based methods, an outcome that is particularly noteworthy given that training-based approaches are typically regarded as more reliable. This gap is even more significant when accounting for the search budget: while training-based competitors require on the order of 2×10^4 , 4×10^4 and 1.2×10^5 seconds under their respective caps, FreeREA operates for only 45 seconds. In addition, FreeREA exhibits substantially higher run-to-run stability

(Table 4.4 and Figure 4.2). The largest standard deviation we observe is 0.05 (on CIFAR-100), whereas REA displays much larger variability (0.31 on CIFAR-10, 0.84 on CIFAR-100 and 0.80 on ImageNet-16-120). Comparable trends hold for the other baselines. Taken together, these results indicate that FreeREA converges toward near-optimal solutions with lower uncertainty, potentially reducing the need for repeated searches to obtain a strong final architecture.

It is worth noting that, for training-free baselines, the savings brought by our approach typically amount to a few minutes, while for training-based methods they translate into hours, and in some cases even days, of computation. We attribute both the improvement in mean test accuracy and the reduced run-to-run variability to two factors: (i) the particular set of training-free metrics adopted to score candidates, and (ii) the changes introduced in the underlying evolutionary procedure. To disentangle these contributions, we run an additional ablation in which we keep the original REA search mechanism but replace the usual post-training validation ranking with our metric-based scoring. We refer to this variant as FreeREA⁻. As shown in Table 4.4, FreeREA⁻ achieves performance close to FreeREA, although it tends to exhibit higher variability in some settings. This result supports the effectiveness of the proposed metrics, while offering limited insight into the impact of the algorithmic modifications. A plausible explanation is that NATS-Bench is comparatively small, so differences in search dynamics are partially masked; we therefore expect the benefits of the modified search to become more apparent on more challenging benchmarks, such as NAS-Bench-101.

Figure 4.3 reports the average test accuracy of all methods as a function of wall-clock search time. The plot highlights that FreeREA achieves the most favorable balance between computational cost and performance. In particular, the training-based baselines, shown on the right-hand side, operate under comparable (capped) time budgets (see Table 4.4). To assess whether these conclusions hold in a more demanding setting, we evaluate FreeREA on NAS-Bench-101, which features a substantially larger and more challenging search space. Here we compare against the other training-free baselines and against the strongest training-based reference, i.e., REA. For REA, we set population size and tournament size to 50 and 5, respectively. This choice departs from the original implementation for NAS-Bench-101 [143] to fit our approximately 24-hour budget for the whole search, since generating the initial pool of 100 trained candidates alone would require roughly 18 hours. The same REA configuration is used for FreeREA⁻ to enable a controlled comparison, whereas FreeREA retains its default population size of 25. The results in Table 4.6 confirm that FreeREA remains the top-performing method, despite requiring only about twelve minutes of search.

Notably, on NAS-Bench-101 we observe a clearer gap between FreeREA and FreeREA⁻ in terms of average test accuracy, lending support to the hypothesis that

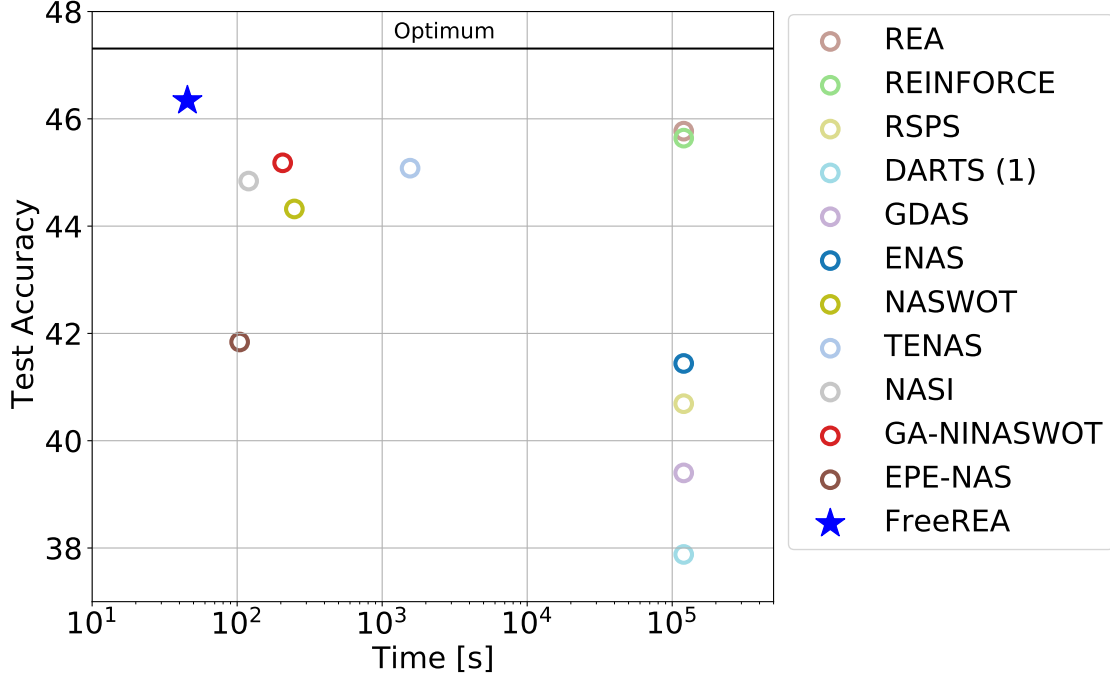


Figure 4.3: Average test accuracy versus time on NATS-Bench (ImageNet-16-120). The time axis is reported in logarithmic scale. For training-based methods, both accuracy and runtime values are taken from the original NATS-Bench paper [26].

our revised evolutionary procedure is more effective than the vanilla REA method. The changes introduced in our method are designed to strengthen exploration: generating offspring from two mutated parents enables a finer search in the proximity of strong candidates, while the crossover operator facilitates jumps to more distant regions of the space. Consistently with this intuition, we expect the advantage of FreeREA over FreeREA⁻ to become more pronounced as the search space grows in complexity and as the available search budget increases. To investigate this trend, we report in Figure 4.4 the evolution over time of mean test accuracy and variance for FreeREA, FreeREA⁻, and REA. An additional observation is that FreeREA exhibits higher variability at the very beginning of the search (between 10^1 and 10^2 seconds), but progressively stabilizes and ends with substantially lower uncertainty. This pattern is not observed for FreeREA⁻ or REA, and is consistent with the intended effect of our modifications: improved exploration early on, followed by more reliable convergence later in the run.

Table 4.4: Test accuracy and search time, measured in seconds, on NATS-Bench. Results for training-based baselines are reported from [26].

	CIFAR-10		CIFAR-100		ImageNet-16-120	
Algorithm	Accuracy	Time	Accuracy	Time	Accuracy	Time
Training-based						
REA [100]	94.02 ± 0.31	2e4	72.23 ± 0.84	4e4	45.77 ± 0.80	1.2e5
REINFORCE [153]	93.90 ± 0.26	2e4	71.86 ± 0.89	4e4	45.64 ± 0.78	1.2e5
RSPS [63]	91.05 ± 0.66	2e4	68.27 ± 0.72	4e4	40.69 ± 0.36	1.2e5
DARTS (1) [69]	59.84 ± 7.84	2e4	61.26 ± 4.43	4e4	37.88 ± 2.91	1.2e5
DARTS (2) [69]	65.38 ± 7.84	2e4	60.49 ± 4.95	4e4	36.79 ± 7.59	1.2e5
GDAS [28]	93.23 ± 0.58	2e4	68.17 ± 2.50	4e4	39.40 ± 0.00	1.2e5
ENAS [96]	93.76 ± 0.00	2e4	70.67 ± 0.62	4e4	41.44 ± 0.00	1.2e5
Training-free						
NASWOT [85]	93.15 ± 0.08	54	70.41 ± 0.35	54	44.32 ± 3.74	49
TENAS [16]	93.18 ± 0.39	327	70.24 ± 1.05	327	43.55 ± 8.24	278
TF-MOENAS [25]	93.96 ± 0.10	7753	71.94 ± 0.50	8361	45.68 ± 1.24	9091
FreeREA ⁻ (ours)	94.30 ± 0.02	45	73.30 ± 0.31	45	46.34 ± 0.00	45
FreeREA (ours)	94.36 ± 0.00	45	73.51 ± 0.05	45	46.34 ± 0.00	45
Optimum						
-	94.37	-	73.51	-	47.31	-

 Table 4.5: Ablation study on NATS-Bench [26] for different values of N and n . The table reports the test accuracy of the final selected model. Each configuration is evaluated over 30 runs with a search budget of 45 seconds.

N	n	CIFAR-10	CIFAR-100	ImageNet-16-120
25	5	94.36 ± 0.00	73.51 ± 0.00	46.34 ± 0.00
100	2	93.95 ± 0.14	71.80 ± 1.85	45.65 ± 2.57
100	50	94.36 ± 0.00	73.51 ± 0.00	46.34 ± 0.00
20	20	94.36 ± 0.00	73.51 ± 0.00	46.34 ± 0.00
100	25	94.34 ± 0.01	73.50 ± 0.00	46.34 ± 0.00
64	16	94.36 ± 0.00	73.51 ± 0.00	46.34 ± 0.00

Constrained search

AutoML research is increasingly expected to focus on models that can be deployed on resource-limited platforms. In this context, while optimization is crucial to make the most of the available compute and memory budget, we deliberately avoid embedding hardware-specific heuristics into the search procedure. Accordingly, we evaluate FreeREA in a constrained setting by restricting the admissible

Table 4.6: Test accuracy and search time, measured in seconds, for training-free methods and REA on NAS-Bench-101 [143].

Algorithm	Accuracy	Time
Training-based		
REA [100]	93.39 ± 0.09	86752
Training-free		
NASWOT [85]	92.23 ± 8.90	114
FreeREA ⁻ (ours)	93.13 ± 1.16	723
FreeREA (ours)	93.80 ± 0.02	724
Optimum	94.31	-

architectures through explicit budget bounds, without otherwise tailoring the algorithm to any particular target device.

Introducing explicit resource budgets is not entirely straightforward, and strong results in the unconstrained regime do not automatically translate to the constrained case. FreeREA ranks candidates using training-free scores that act as proxies for final test accuracy. However, the relationship between these scores and accuracy can change when attention is restricted to a feasible subset of the search space. As a consequence, a truly practical training-free NAS method should remain effective both with and without deployment constraints, ideally without requiring specific changes to either the scoring metrics or the search procedure.

Since the tabular benchmarks we consider do not provide established constrained evaluation baselines, we assess FreeREA under three progressively tighter budgets on FLOPs and on the number of parameters: $(10^8, 8 \times 10^5)$, $(7 \times 10^7, 5 \times 10^5)$, and $(4 \times 10^7, 3 \times 10^5)$. We run the method on the three NATS-Bench datasets under each constraint level. Given the lack of competitors, we report in Table 4.7 a comparison between the architectures found by FreeREA and the best feasible model available in the benchmark under the same constraints, which effectively provides an upper bound in this setting. On the two CIFAR tasks, the remaining gap to this oracle solution (i.e., the regret) is 0.32% on average, while on the more demanding ImageNet-16-120 the average gap increases to 0.56%. These figures mirror the unconstrained analysis, where the distance from the optimum is also larger on ImageNet-16-120 (0.97% on average). Overall, the results suggest that the proposed metric combination remains a reliable surrogate for test accuracy even under stringent resource budgets, and can therefore serve as a useful reference point for future work on constrained training-free NAS.

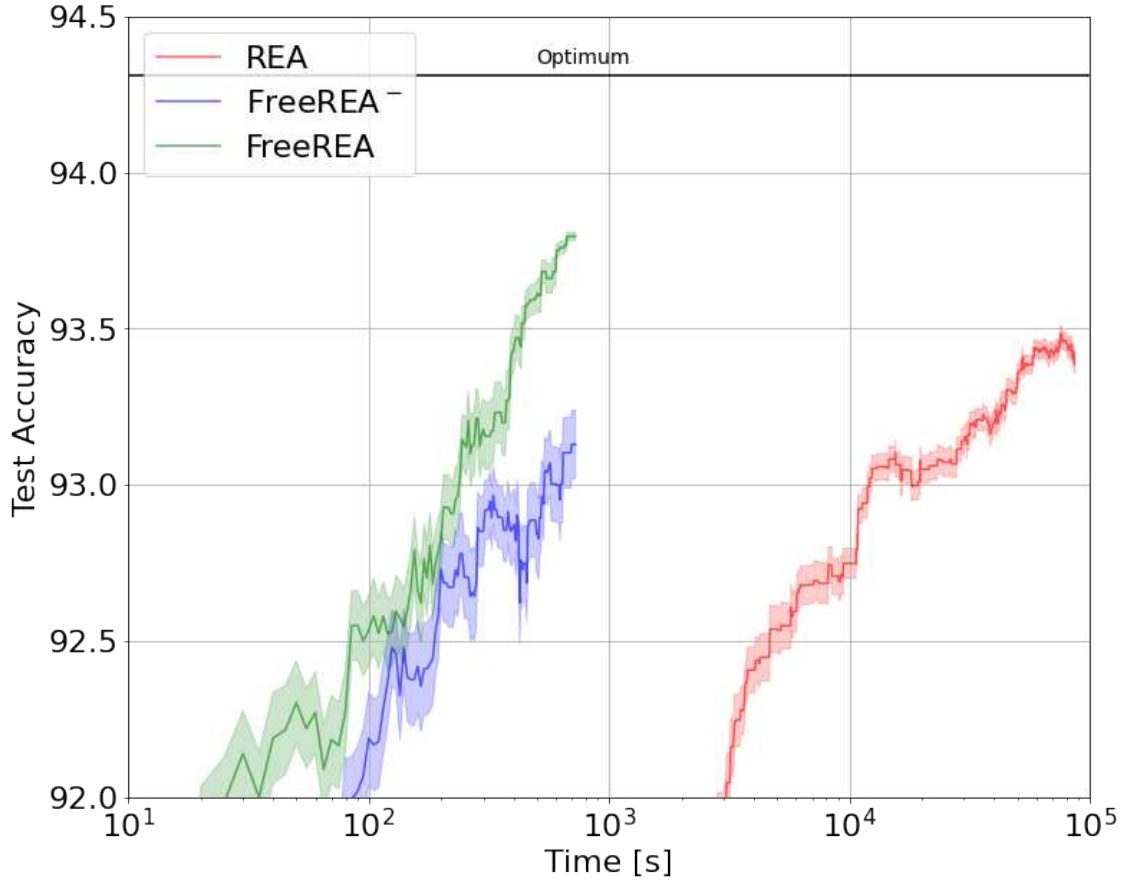


Figure 4.4: Evolution of test accuracy over search time for REA, FreeREA⁻, and FreeREA on NAS-Bench-101. The time axis is shown in logarithmic scale, and standard deviations are reduced by a factor of 10 for readability. FreeREA exhibits higher variability during the early search phase, between 10¹ and 10² seconds, but converges to lower uncertainty near the end of the search, while REA and FreeREA⁻ maintain comparable variance during the process.

4.4 Decoupling topology and size in training-free NAS

While FreeREA showed that a carefully designed combination of training-free metrics can effectively replace full training during the search, it also highlighted a more fundamental limitation of aggregated proxy-based evaluation. Different zero-cost metrics do not capture the same architectural properties with the same fidelity: some are more informative about trainability or model size, while others are more closely correlated to the expressivity of the network. Treating all these

Table 4.7: FreeREA test accuracy on NATS-Bench under constrained search settings. Regret represents the gap between the optimal and average test accuracy obtained during search. Each run is stopped after 45 seconds.

FLOPs	Params	Accuracy	Optimum	Regret
CIFAR-10				
1e8	8e5	94.12 ± 0.01	94.31	0.19
7e7	5e5	93.02 ± 0.13	93.75	0.73
4e7	3e5	91.30 ± 0.00	91.32	0.02
CIFAR-100				
1e8	8e5	72.08 ± 0.01	72.43	0.35
7e7	5e5	71.07 ± 0.00	71.63	0.54
4e7	3e5	68.39 ± 0.00	68.48	0.09
ImageNet-16-120				
1e8	8e5	46.08 ± 0.08	46.85	0.77
7e7	5e5	45.35 ± 0.04	45.95	0.65
4e7	3e5	40.73 ± 0.00	41.00	0.27

signals as components of a single global score may therefore lead to suboptimal search decisions, especially when the search space includes heterogeneous design dimensions.

Starting from this observation, the next step of our research was to move beyond metric aggregation and investigate whether distinct aspects of architecture design could be guided separately. In particular, we focused on the distinction between topology, which determines the structural organization of the network, and size, which controls its dimensionality and computational footprint. This led to the introduction of Entropic Score, a novel training-free metric designed to capture the expressive potential of a candidate architecture through the entropy of its activations at initialization.

Despite building on FreeREA, in this work Entropic Score was not used as an additional term in a cumulative ranking function. Instead, it was paired with LogSynflow within a decoupled search strategy, where the two metrics supervise complementary design choices: Entropic Score drives the search over topological characteristics, while LogSynflow guides dimensional decisions. This formulation makes it possible to exploit the strengths of each proxy more explicitly, rather than forcing them into a single undifferentiated fitness value.

Compared with the benchmark-oriented setting adopted for FreeREA, this work

also considers a more challenging and application-driven scenario, namely the design of tiny hybrid CNN-Transformer architectures for ImageNet classification under mobile-scale resource budgets. In this sense, Entropic Score represents both a methodological refinement and a step toward more realistic training-free NAS for edge-oriented vision models.

4.4.1 Method

Architectural design choices can be broadly grouped into two main categories: topological and dimensional. Topological aspects concern the structural organization of the network, including the types of layers involved, the connectivity patterns among them, and their arrangement within the overall architecture (Figure 4.5). Dimensional aspects, instead, are related to model size and computational complexity. These properties can be adjusted by acting, for instance, on the depth of the network, the number of channels in each layer, the expansion ratios of bottleneck modules, and similar design parameters. Since these two design dimensions reflect different properties of the architecture, in this work they are guided through distinct metrics: LogSynflow, already introduced in Section 4.3.1, is used to evaluate the dimensional aspects of the model, while the novel metric Entropic Score is introduced to guide topology decisions.

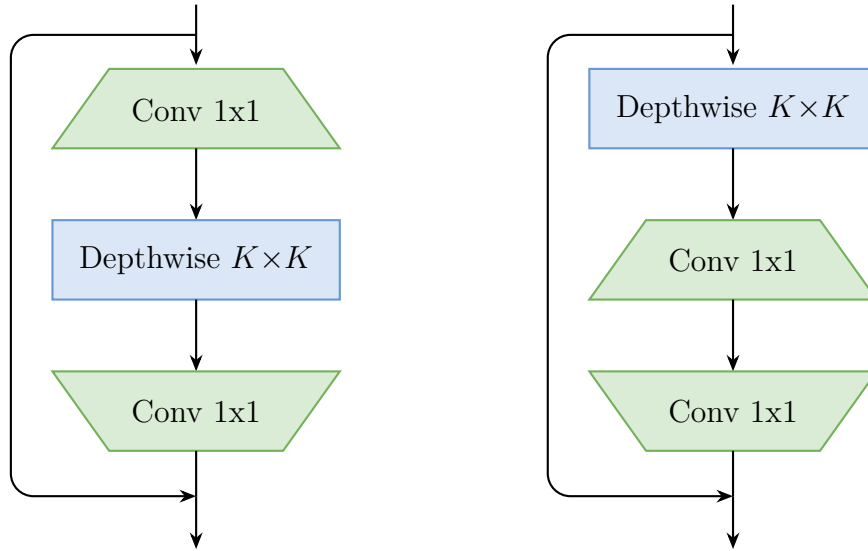


Figure 4.5: Alternative locally-enhanced FFN block designs. The left panel shows an Inverted Bottleneck [107] block, while the right panel illustrates a ConvNeXt [73] block.

Entropic Score

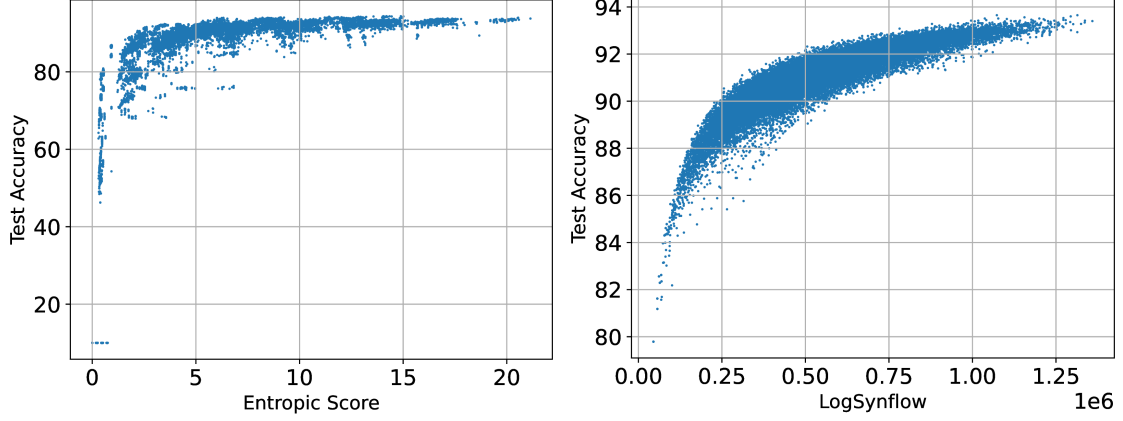


Figure 4.6: Relationship between training-free metrics and CIFAR-10 test accuracy. The left plot reports Entropic Score on the topological search space of NAS-Bench-101 [143], while the right plot shows LogSynflow on the dimensional search space of NATS-Bench [26]. Entropic Score is particularly effective for topology selection, while LogSynflow is better suited to architectural dimensioning.

The proposed metric is intended to quantify the ability of an architecture to represent and propagate informative signals. Its value is computed by feeding a random input tensor through the network and then aggregating the average element-wise entropy of the normalized activations across the model. From an intuitive perspective, a higher Entropic Score suggests a richer flow of information during the forward pass, which may in turn favor the fitting capacity of the architecture. Under this interpretation, Entropic Score can be regarded as a proxy for network expressivity. Like Synflow [123] and related measures, it is entirely data-agnostic, since its computation does not depend on training samples or labels. As a result, the search procedure is not tied to a specific dataset, but rather aims at identifying architectures with more general potential, which can then be employed in different application settings. Accordingly, we adopt a general search strategy that is not constrained to a specific dataset, with the goal of identifying models for a given task that remain applicable across different settings.

Given a network parameterized by θ , the proposed aggregated metric is defined as follows:

$$E(\theta) = - \sum_{i=1}^N \frac{1}{K} \sum_{j=1}^K a_{ij}(\theta) \cdot \log(a_{ij}(\theta) + \epsilon), \quad (4.6)$$

where N denotes the number of activations in the network, K is the number of elements in the normalized activation tensor $\mathbf{a}$, and ϵ is a small constant introduced

for numerical stability. A key component of the proposed method is the computation of a layer-wise quantity, which is subsequently aggregated to obtain a score for the entire architecture. Before evaluating the metric, the network undergoes a preparation procedure inspired by Synflow [123]. Specifically, all normalization operators are removed, the network weights are replaced by their absolute values, and nonlinear activation functions such as GELU [46] and Swish [99] are replaced with ReLU [87]. This preprocessing step ensures that only non-negative values are propagated through the network, so that the logarithm in (4.6) is well defined. Importantly, these transformations are applied only when computing the Entropic Score at initialization, while the architecture subsequently trained and evaluated retains its original normalization layers and activation functions. Following this preprocessing, a random input tensor $x \in [-0.5, 0.5]$ is fed through the model. For each layer, the resulting activations are normalized to the interval $(0, 1]$ by dividing them by the maximum value computed along the channel dimension. The corresponding layer-wise score is then obtained by computing the average element-wise entropy of the normalized activations. The final Entropic Score is then obtained by aggregating these values across all layers, thus yielding a scalar indicator of the expressive potential of the network topology. In practice, to reduce the effect of random fluctuations, the metric is evaluated three times using different initializations of both the network and the input tensor, and the final score is taken as the average of the three runs.

Within the NAS setting, Entropic Score is intended to capture the potential expressivity of a candidate architecture, under the assumption that higher values correspond to richer and more structured activation patterns. Empirically, this metric proves particularly effective when guiding topological design choices, as also highlighted by the results reported in Table 4.11a and Figure 4.6.

Decoupled search algorithm

The search procedure is built upon an evolutionary framework derived from REA [100], which is adopted as the core optimization strategy. In line with the refinements proposed in Section 4.3, the algorithm includes both multiple mutations at each iteration and crossover operations between parent architectures, in order to improve exploration, reduce locality effects and promote greater diversity within the population. All mutation and crossover operations are sampled with uniform probability.

The procedure is preceded by a multi-start phase designed to mitigate the dependence on the initial population. More specifically, several sub-populations are initialized and evolved independently for a limited amount of time, and the

best architectures emerging from each of them are then used to seed the main evolutionary stage. This initialization strategy improves robustness across runs and reduces the variance typically associated with purely random starting conditions.

A central aspect of the method is that the search is not driven by a single aggregated score. As previously discussed in Section 2.2.1, topology and dimensionality correspond to different families of architectural decisions, and training-free metrics do not provide equally informative signals for both. For this reason, the proposed approach combines Entropic Score and LogSynflow within a decoupled search strategy, rather than merging them into a single fitness value. LogSynflow provides information related to gradient propagation and network complexity, and proves to be particularly effective for dimensional choices. Entropic Score, on the other hand, is used to assess the expressive quality of the architectural topology. The two metrics therefore offer complementary cues.

This separation is motivated by the observation that a seamless aggregation of different zero-cost proxies, as commonly done in previous training-free NAS approaches [1], [16], may lead to suboptimal decisions when the individual metrics prioritize different properties of the model. A metric that is highly informative for sizing the network may contribute little to the selection of its structural configuration, while the opposite may hold for a metric tailored to topological assessment. The proposed decoupled formulation addresses this issue by assigning each metric to the design dimension for which it is most suitable.

Concretely, Entropic Score guides the selection of topological features, such as the type of block or the kernel size, whereas LogSynflow is used to determine dimensional attributes, including the number of output channels and the expansion ratios of bottleneck or MLP components. In this way, each phase of the search focuses on a restricted subset of architectural choices, while still contributing to the progressive refinement of the same candidate solutions.

The overall procedure is structured accordingly. During the initial multi-start stage, the search is driven exclusively by Entropic Score, in order to favor the emergence of promising topological configurations. The main search phase then alternates cyclically between topology optimization and size optimization, using the best architectures obtained at one stage as the starting point for the next. This alternating scheme allows the method to exploit the strengths of both metrics without forcing them into direct competition within the same ranking criterion.

In the experiments, the multi-start phase evolves five independent populations for three minutes each. Population size and tournament size are set to 25 and 5 during this stage, following the procedure described in Section 4.3, and are doubled in the subsequent main search phase. Topology and size optimization are then

alternated every five minutes for the smaller models and every six minutes for the larger ones, leading to a total search time of 45 and 55 minutes, respectively. Overall, the full procedure remains below one GPU hour, while providing a more accurate and flexible way to explore the joint design of topology and model size.

Search space

The search space considered in this work is derived from the architecture of EfficientFormerV2 [66], a modern hybrid transformer model that combines Inverted Bottleneck [107] with efficient Multi-Head Self-Attention (MHSA) layers [66]. In particular, the attention formulation adopted in this architecture family incorporates additional mechanisms such as downsampling, locality-enhancing operations [36], [112] and Talking Heads [109].

The use of training-free metrics makes the exploration process considerably more efficient from a computational standpoint, which in turn enables a more extensive investigation of the search space. For this reason, the original space of [66] is extended and refined to provide greater flexibility in the definition of the network components. More specifically, each Feed-Forward Network (FFN) block is allowed to vary independently in terms of output dimensionality and kernel size, rather than relying on a stage-wise fixed output dimension and a uniform 3×3 kernel across the entire network, as in [66]. In addition, the FFN block can be instantiated either as an Inverted Bottleneck [107] or as a ConvNeXt-style block [73] by modifying the position of the depthwise convolution (Figure 4.5). Greater flexibility is also introduced in the MHSA layers, for which the number of heads and the head dimension are treated as searchable parameters. By contrast, the kernel size of FFN blocks following attention layers is fixed to 3×3 , since larger kernels are unnecessary in the presence of the global receptive fields introduced by MHSA.

Overall, the searchable attributes of each FFN block include the FFN configuration (IBN or ConvNeXt), the output dimension, the kernel size, and the expansion ratio. For Transformer layers, the search additionally involves the number of attention heads and the head dimension, alongside the FFN configuration, output size, and expansion ratio. The distinction between topological and dimensional variables is reported in Table 4.8. To preserve the feasibility of residual connections, output dimensions are constrained to be non-decreasing, so that skip connections can always be implemented through zero-padded residuals [41]. The number of blocks in each stage, as well as the design of the downsampling layers, is instead kept consistent with [66]. As a result of these extensions, the proposed search space is substantially larger and more fine-grained than the one originally introduced in

Table 4.8: Division of searchable dimensions between topology and size. Following the convention adopted in NATS-Bench [26], kernel size is considered a topological dimension rather than a size-related one, as it characterizes the operator choice rather than its dimensionality.

Topology	Size
FFN Type	Output Channels
Kernel Size	Expansion Ratio
Number of Heads	Head Dimension

EfficientFormerV2 [66], allowing a more detailed exploration of both topological and dimensional design choices.

4.4.2 Experiments

To assess the effectiveness of the proposed method against both state-of-the-art architectures and existing NAS approaches, we performed a set of experiments within the search space described in the previous paragraphs. The goal of the search was to identify the best-performing architecture under different model-size constraints, following the footprint settings defined in [66]. Specifically, three target configurations were considered, namely S0, S1 and S2, corresponding to maximum parameter budgets of 3.5M, 6M, and 12.5M respectively. The resulting architecture family is referred to as ESFormer, after the proposed Entropic Score metric.

Table 4.10 reports the ImageNet-1k performance of several state-of-the-art families of mobile architectures across different model scales. Unless otherwise stated, the reported results are taken from the corresponding original publications. The only exception is represented by the EfficientFormerV2 models, which were re-trained using their original training setup, but without distillation.

Comparison with hand-designed networks

The architectures discovered by the proposed search consistently achieve higher accuracy than most manually designed competitors. In the smallest regime, ESFormer-S0 achieves a top-1 accuracy of 75.5%, outperforming all the alternative architectures with comparable, or even slightly larger, computational budgets. In the medium-scale setting, ESFormer-S1 obtains 78.8% top-1 accuracy, matching the best results among the considered hand-designed models, namely EdgeNeXt-S [82]. For the largest budget, EdgeViT-S [18] achieves a marginally higher accuracy than

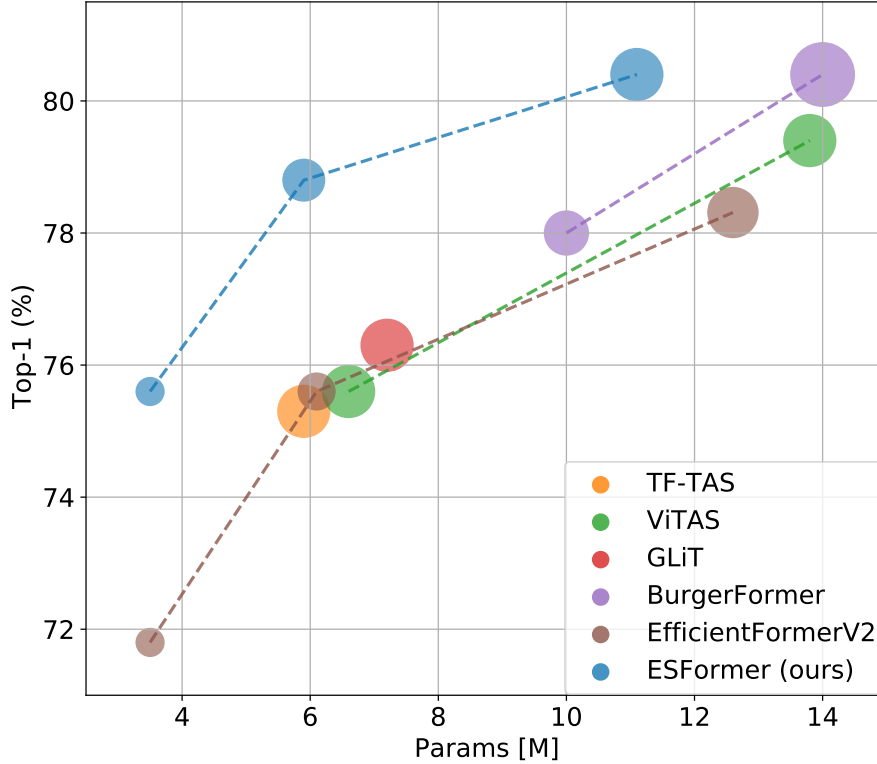


Figure 4.7: Comparison between model size and ImageNet-1k top-1 accuracy for state-of-the-art NAS approaches. Marker size encodes the computational cost of each model in terms of MACs.

ESFormer-S2. However, this improvement comes at the cost of substantially higher complexity, requiring approximately 35% more MACs. A similar observation holds when comparing architectures at intermediate scale where ESFormer-S1 surpasses EdgeViT-XS [18] by more than one percentage point under comparable computational budgets. Overall, when considering models with a similar number of parameters and operations, the proposed architectures achieve the best top-1 accuracy across all examined scenarios.

Comparison with automatically designed networks

When compared with architectures obtained through other NAS methods, the proposed approach is particularly notable for its search efficiency (Figure 4.7). The search time required to obtain the ESFormer models is always below one GPU hour, while most competing methods require several GPU days to identify a final

architecture. In particular, the proposed method reduces the search cost by approximately a factor of 12 with respect to TF-TAS [151], which is the fastest among the compared alternatives, while also improving top-1 accuracy by more than 3% with lower requirements in terms of MACs.

Among the NAS-designed baselines, BurgerFormer-Small [142] is the only model that achieves final accuracy competitive with ESFormer-S2. Nevertheless, this result is obtained with a search cost that is orders of magnitude higher, as well as with a significantly more complex final architecture, involving approximately 26% more parameters and 50% more MACs. It is also worth emphasizing that the proposed architectures substantially outperform EfficientFormerV2 across all model scales, with improvements exceeding 3 percentage points in top-1 accuracy in every considered setting.

Training details

The performance of the discovered architectures is assessed on ImageNet-1k at the standard input resolution of 224×224 . Unless otherwise specified, all models are trained for 300 epochs using Binary Cross-Entropy loss and the AdamW optimizer [78], with learning rate, batch size and weight decay set to 2×10^{-3} , 1024, and 2×10^{-2} respectively. The optimization process follows a cosine learning-rate schedule [79], preceded by a warm-up stage of 20 epochs to enhance training stability. Our training pipeline relies on widely adopted augmentation and regularization strategies, namely mixup/Cutmix [145], [147], RandAugment [21], Random Erasing [150], Stochastic Depth [51], Gradient Clipping [91] and Label Smoothing [120]. More aggressive augmentation and regularization settings are used for the largest models. A detailed summary of the adopted hyperparameters is provided in Table 4.9.

4.4.3 Ablation study

Correlation in NAS benchmarks

An aspect worth analyzing is the rank correlation between training-free metrics and the final test accuracy on standard NAS benchmarks. In this respect, Table 4.11 compares the correlation between CIFAR-10 top-1 accuracy and the rankings produced by state-of-the-art training-free metrics on two widely adopted benchmark datasets, namely NAS-Bench-101 [143] and NATS-Bench [26]. More specifically, Table 4.11a refers to NAS-Bench-101, which includes approximately

Table 4.9: Hyper-parameters used to train the models on ImageNet-1k.

	S0-S1	S2
Optimizer	AdamW	AdamW
Batch Size	1024	1024
LR	0.002	0.002
LR schedule	Cosine	Cosine
Training Epochs	300	300
Warmup Epochs	20	20
Weight Decay	0.02	0.02
Gradient Clipping	2.0	0.01
Mixup/Cutmix	0.8/1.0	0.8/1.0
RandAugment	m7-n1	m9-n1
Random Erasing	0.0	0.25
Stochastic Depth	0.05	0.1
Label Smoothing	0.1	0.1

423,000 convolutional architectures differing in topology, while Table 4.11b considers NATS-Bench, whose search space contains more than 30,000 architectures sharing the same topology but differing in size. The results highlight the strong capability of Entropic Score to identify promising topological configurations. In particular, its correlation with the final accuracy is markedly higher than that of the other considered training-free metrics, and is nearly twice the value obtained by NASWOT [85]. At the same time, the analysis on NATS-Bench shows that Entropic Score is considerably less informative when the main source of variation lies in the dimensional properties of the network. In this setting, LogSynflow is substantially more effective.

The same trend can also be observed in Figure 4.6, which reports the CIFAR-10 top-1 accuracy as a function of the rank assigned by Entropic Score and LogSynflow in topological and dimensional search spaces, respectively. Taken together, these results further support the view that the two metrics provide complementary information and therefore motivate their joint adoption within the proposed search framework.

Search ablation

To further investigate the benefits of Entropic Score as a search signal, the search space is enriched with an additional topological option, namely a standard

Table 4.10: ImageNet-1k classification results. Models are evaluated at the standard 224×224 resolution, except for MobileViTs, which use 256×256 inputs. $\diamond$ denotes training with distillation, $\dagger$ indicates training with the original configuration without distillation, and $\ddagger$ marks conservatively estimated search times when the original papers do not report them. Search time is reported in GPU hours/days. The symbols $\uparrow$ and $\downarrow$ indicate whether higher or lower values are better.

Model	Type	Design	Search Time	Params [M] $\downarrow$	MACs [G] $\downarrow$	Epochs $\downarrow$	Top-1 (%) $\uparrow$
XCiT-N12 [2]	Hybrid	Manual	-	3.0	0.5	400	69.9
MobileViT-XS [84]	Hybrid	Manual	-	2.3	0.7	300	74.8
EdgeViT-XXS [18]	Hybrid	Manual	-	4.1	0.6	300	74.4
MobileFormer-96M [17]	Hybrid	Manual	-	4.6	0.1	450	72.8
EfficientFormerV2-S0 [66]	Hybrid	Auto	> 8 days	3.5	0.4	300	71.8 $\dagger$
ESFormer-S0 (ours)	Hybrid	Auto	0.75 hours	3.5	0.4	300	75.5
DeiT-T [124]	Transformer	Manual	-	5.9	1.2	300	72.2
XCiT-T12 [2]	Hybrid	Manual	-	7.0	1.2	400	77.1
MobileViT-S [84]	Hybrid	Manual	-	5.6	2.0	300	78.4
EdgeViT-XS [18]	Hybrid	Manual	-	6.7	1.1	300	77.5
LeViT-128S [37]	Hybrid	Manual	-	7.8	0.3	1000	76.6 $\diamond$
MobileFormer-151M [17]	Hybrid	Manual	-	7.6	0.2	450	75.2
Edge-NeXt-S [82]	Hybrid	Manual	-	5.6	1.0	300	78.8
TF-TAS-Ti [23]	Transformer	Auto	0.5 days	5.9	1.4	300	75.3 $\diamond$
ViTAS-DeiT-A [115]	Transformer	Auto	$\sim$ 8 days $\ddagger$	6.6	1.4	300	75.6
GLiT-Tiny [13]	Hybrid	Auto	> 10 days $\ddagger$	7.2	1.4	1000	76.3 $\diamond$
BurgerFormer-Tiny [142]	Hybrid	Auto	11 days	10	1.0	300	78.0
EfficientFormerV2-S1 [66]	Hybrid	Auto	> 8 days $\ddagger$	6.1	0.7	300	75.6 $\dagger$
ESFormer-S1 (ours)	Hybrid	Auto	0.75 hours	5.9	0.9	300	78.8
LeViT-192 [37]	Hybrid	Manual	-	10.9	0.7	1000	80.0 $\diamond$
MobileFormer-508M [17]	Hybrid	Manual	-	14.0	0.5	450	79.3
XCiT-T24 [2]	Hybrid	Manual	-	12.0	2.3	400	79.4
EdgeViT-S [18]	Hybrid	Manual	-	11.1	1.9	300	81.0
ViTAS-Twins-T [115]	Hybrid	Auto	$\sim$ 8 days $\ddagger$	13.8	1.4	300	79.4
BurgerFormer-Small [142]	Hybrid	Auto	11 days	14.0	2.1	300	80.4
EfficientFormerV2-S2 [66]	Hybrid	Auto	> 8 days $\ddagger$	12.6	1.3	300	78.31 $\dagger$
ESFormer-S2 (ours)	Hybrid	Auto	0.9 hours	11.1	1.4	300	80.4

Residual Bottleneck block [43]. This design choice is generally ill-suited to mobile-scale architectures and would normally be discarded by conventional training-based NAS methods, which rely directly on validation accuracy to guide the optimization process. By contrast, a training-free search procedure based on suitable proxy metrics may repeatedly favor this block type, ultimately leading to suboptimal architectures. This setting therefore provides a useful stress test for assessing whether Entropic Score can effectively avoid this limitation. The proposed decoupled strategy is then evaluated through an ablation study in which several search configurations are compared, as reported in Table 4.12. In particular, three variants are considered: a search guided exclusively by LogSynflow, a search based on a direct combination of LogSynflow and Entropic Score, and the proposed decoupled search algorithm. In all cases, the search is performed under a constraint of at most 6

Table 4.11: Kendall and Spearman rank-correlation values between training-free scores and CIFAR-10 top-1 accuracy, computed a) on the topological search space of NAS-Bench-101 and b) on the size search space of NATS-Bench. Entropic Score is mainly effective for topology-related choices, while LogSynflow better captures variations in model dimension.

(a) Correlation w.r.t. NAS-Bench-101 *topological* search space.

Metric	Kendall τ	Spearman ρ
NASWOT	0.26	0.37
LogSynflow	0.31	0.45
Entropic Score (ours)	0.50	0.68

(b) Correlation w.r.t. NATS-Bench *size* search space.

Metric	Kendall τ	Spearman ρ
NASWOT	0.45	0.63
LogSynflow	0.76	0.92
Entropic Score (ours)	0.03	0.04

Table 4.12: Ablation study of the proposed search strategy on the extended search space, where Residual Bottleneck blocks are included as additional architectural choices. ImageNet-1k top-1 accuracy is reported. The symbols $\downarrow$ and $\uparrow$ indicate whether lower or higher values are better, respectively.

LogSynflow	Entropic Score	Decoupling	Params [M] $\downarrow$	MACs [G] $\downarrow$	Top-1 (%) $\uparrow$
✓	✗	✗	6.00	0.86	72.4
✓	✓	✗	5.92	0.94	75.7
✓	✓	✓	5.97	0.96	77.8

million parameters, therefore focusing on medium-scale architectures.

As shown in Table 4.12, the direct combination of LogSynflow and Entropic Score already yields a substantial improvement over the baseline configuration based solely on LogSynflow, leading to an accuracy gain of more than 3 percentage points. The proposed decoupled strategy further improves upon this result. By assigning the two metrics to the design dimension for which they are most informative, it produces architectures with consistently higher performance, providing an additional improvement of about 2 percentage points. Overall, this analysis confirms that the benefit of Entropic Score does not merely stem from another proxy added to the evaluation process. Rather, the results suggest that explicitly

separating topological and dimensional decisions allows the search to exploit the strengths of the two metrics effectively.

4.5 Conclusions

This chapter investigated how training-free performance proxies can substantially reduce the computational burden of Neural Architecture Search (NAS) while still enabling the discovery of competitive, and often state-of-the-art, models in resource-constrained scenarios. The two contributions presented in this chapter follow a natural development: from demonstrating that carefully combining existing zero-cost proxies can make training-free NAS practically effective, to showing that designing specialized proxies and using them within a structured search strategy can further improve both reliability and scalability. First, FreeREA established that training-free NAS can be both a competitive and fast alternative to training-based search on widely adopted benchmark search spaces. By selecting and calibrating a complementary set of training-free metrics as a surrogate of test accuracy and by employing an evolution-based search policy, FreeREA achieved strong results while requiring only minutes of search time even in the largest benchmark setting. Beyond the unconstrained case, the constrained experiments further highlighted that the same proxy-driven evolutionary procedure remains effective when explicit limits (e.g., on FLOPs and number of parameters) restrict the feasible region of the search space, making the approach relevant to realistic deployment scenarios.

Building on these insights, the second part of the chapter introduced Entropic Score, which pushes the training-free paradigm forward by emphasizing specialization and decoupling. While FreeREA relies on a single aggregated ranking signal derived from multiple metrics, Entropic Score motivates a more structured view: different architectural attributes (notably topology and size) are better assessed by different proxies. Entropic Score was proposed as a topology-oriented indicator based on the entropy of activations, and it was paired with LogSynflow to guide dimensionality. This decoupled, cyclic optimization strategy allows each proxy to operate in the regime where it is most informative, leading to a more targeted search and enabling the efficient discovery of high-performing tiny hybrid Transformer architectures. The resulting ESFormer family shows that training-free NAS can extend beyond classic CNN-centric benchmark spaces to more flexible hybrid design spaces, while still maintaining a very small search budget. The empirical evidence confirms that Entropic Score is particularly effective in ranking topologies, while it is less informative for determining network dimensions on its own, and therefore needs complementary metrics and explicit decoupling. More generally,

these observations reinforce a central message of the chapter: the practical effectiveness of training-free NAS depends on how proxies are combined, specialized and embedded into the search strategy.

Overall, the contributions presented here support a unified takeaway: training-free NAS can be a viable and highly efficient approach for designing tiny and hybrid architectures, provided that proxies are treated as structured signals rather than monolithic substitutes for training. However, it must be noted that the empirical evidence presented in this chapter is limited to image classification and to a relatively small number of search spaces, ranging from cell-based CNN benchmarks to a hybrid space derived from EfficientFormerV2. Therefore, further validation would be needed to establish whether the observed behavior of the considered proxies, and in particular their specialization with respect to topological and dimensional design choices, generalizes to substantially different architecture families and search spaces. Moreover, the efficiency constraints considered here are mainly expressed through hardware-independent quantities such as parameter count or FLOPs, rather than direct measurements of latency, memory usage or energy consumption on real devices. These limitations motivate several possible directions for future work, including evaluating training-free proxies across broader architectural spaces and tasks such as detection and segmentation, incorporating hardware-specific objectives into the search, developing more robust proxies across heterogeneous settings, and exploring hybrid strategies that integrate lightweight training signals when proxies are uncertain.

Chapter 5

A hybrid CNN-ViT architecture search benchmark

The content, images, and tables included in this chapter were originally presented in the following publication:

- *L. Robbiano, F. Pistilli, and G. Averta, A CNN-ViT hybrid architecture search benchmark on a large-scale dataset, IEEE Access, vol. 13, pp. 209 965–209 979, 2025*

5.1 Introduction

As discussed in Section 2.2.4, the high computational cost of NAS poses a significant barrier to the systematic evaluation of novel NAS methodologies. Consequently, assessing new algorithms often requires large-scale computational resources, limiting reproducibility and fair comparisons across approaches. To mitigate these issues, a number of NAS benchmarks have been proposed, with the aim of enabling reproducible research and standardized evaluation. By defining fixed search spaces and providing precomputed performance indicators for candidate architectures, these benchmarks allow NAS strategies to be compared under consistent experimental conditions, without the need for repeated and expensive model training.

However, making NAS evaluation inexpensive and reproducible is only part of the problem. A benchmark can provide a fixed search space and queryable

performance values while still encouraging conclusions that are specific to a narrow architectural family, a simplified dataset, or a limited evaluation objective. This issue becomes particularly relevant when benchmarks are used to validate training-free metrics, or evaluation dimensions that go beyond the simple validation accuracy, as detailed in Section 2.1.4. Such settings reveal a first limitation related to the architecture space. Many influential NAS benchmarks were designed around purely convolutional architectures, which have been extremely useful for controlled experimentation but only partially reflect the structure of recent vision backbones. Modern efficient architectures increasingly combine convolutional blocks, transformer-based modules, large-kernel spatial processing, inverted bottleneck structures, and stage-wise capacity allocation. As a result, a NAS method or a performance proxy that performs well on compact CNN-centric spaces may not behave in the same way when applied to hybrid architectures, where local and global feature processing mechanisms coexist. Evaluating NAS in this regime therefore requires a benchmark based on a search space that explicitly represents this architectural heterogeneity.

A second limitation concerns the scale and representativeness of the data used to train and evaluate candidate models. Small datasets like CIFAR and heavily reduced ImageNet variants make exhaustive benchmarking more affordable, but they may distort the relative ranking of architectures. For NAS evaluation, the relevant question is not whether a model can achieve reasonable performance on a small proxy dataset, but whether the ranking induced by that dataset is informative for larger-scale visual recognition. This is especially important near the top of a search space, where candidate architectures often differ by small margins and where noisy or poorly representative proxies can change which design would be selected by a search algorithm.

A third limitation is that architecture quality cannot be reduced to in-distribution accuracy alone. In realistic settings, a model may also be required to maintain performance under distribution shift, exhibit stable behavior across different training seeds, and satisfy deployment constraints on heterogeneous hardware. These dimensions are not interchangeable: the architecture with the highest validation accuracy may not be the most robust under distributional shift. Similarly, parameter count and theoretical computational cost do not necessarily translate into the same inference latency ranking across different devices. A benchmark for practical NAS should therefore expose these trade-offs rather than hiding them behind a single scalar objective.

The benchmark introduced in this chapter, HyViTas-Bench, is designed to address these requirements. It provides a controlled yet realistic evaluation environment for hybrid CNN-ViT architectures, using a reduced ImageNet-based dataset

that preserves a substantially larger scale than conventional small-image benchmarks. Each architecture is trained multiple times to account for variability due to random initialization and stochastic optimization. In addition to in-distribution accuracy, the benchmark includes out-of-distribution evaluations, latency measurements across heterogeneous hardware platforms, and precomputed training-free proxy scores. In this way, HyViTas-Bench supports the analysis of NAS methods from multiple perspectives.

The remainder of the chapter first briefly reviews the NAS benchmarks most closely related to this work, then describes the HyViTas-Bench search space, datasets, and training protocol. Finally, it analyzes the benchmark through standard NAS baselines and training-free proxies.

5.2 Related works

Building on the general discussion of NAS benchmarks introduced in Section 2.2.4, this section focuses on the benchmarks more closely related to the proposed contribution. Early tabular benchmarks such as NAS-Bench-101 [143], NAS-Bench-201 [27], and NATS-Bench [26] have played a central role in making NAS evaluation more reproducible and computationally accessible.

NAS-Bench-101 [143] provides exhaustive evaluations for approximately 423,000 cell-based architectures trained on CIFAR-10. The benchmark relies on a constrained search space defined as a Directed Acyclic Graph (DAG) with a fixed number of nodes and edges, enabling reproducible and systematic comparisons among NAS algorithms. While highly impactful, NAS-Bench-101 is inherently limited by its exclusive focus on a single small-scale dataset. To alleviate this limitation, NAS-Bench-201 [27] introduced a more compact search space of 15,625 candidate architectures, evaluated across multiple datasets, including CIFAR-10 [56], CIFAR-100 [56], and ImageNet-16-120 [19]. Due to the presence of isomorphic architectures, however, the effective number of distinct network topologies is reduced to 6,466. Building upon NAS-Bench-201, NATS-Bench [26] further enriched the benchmark by extending the search space along the width dimension, varying the number of channels assigned to each architecture. This design choice substantially increases the total number of candidate models, resulting in a benchmark comprising more than 32,000 architectures.

In parallel with these efforts, the growing interest in hardware-aware NAS has motivated the introduction of benchmarks explicitly accounting for efficiency constraints. HW-NAS-Bench [59] extends NAS-Bench-201 with hardware-related

performance indicators, including inference latency and energy consumption, measured across different computing platforms. Another significant contribution is Lat-Bench [30], which focuses specifically on runtime latency and evaluates candidate architectures on a heterogeneous set of devices, ranging from embedded systems to mobile GPUs. More recently, NAS-Bench-MR [24] broadened the scope of NAS benchmarking by supporting multi-resolution search across several computer vision tasks, such as image segmentation, video understanding, and 3D object detection. However, in this setting each architecture is trained only once per task, preventing an explicit assessment of performance variability across repeated training runs.

Despite the substantial contribution of these benchmarks, most of them remain primarily focused on convolutional architectures, offering no support for the evaluation of more recent architectural paradigms. As Vision Transformers [29] and hybrid CNN-ViT models have gained increasing relevance, a clear gap has emerged in the systematic benchmarking of NAS methods targeting these architectures.

In this context, OoD-ViT-NAS [47] was recently introduced as a benchmark specifically tailored to Vision Transformers, with an emphasis on assessing generalization under out-of-distribution (OoD) conditions. The benchmark also investigates the behavior of training-free performance predictors, showing that such metrics exhibit weak correlation with OoD accuracy, while model size appears to be a more reliable indicator. However, similarly to NAS-Bench-MR, OoD-ViT-NAS evaluates each architecture through a single training instance, which limits the analysis of robustness with respect to different random initializations. In this case, the lack of repeated training runs stems from the adoption of one-shot supernet weight sharing, a strategy that significantly reduces computational cost but is known to suffer from weight entanglement, whereby shared parameters across sub-networks can bias performance estimates and lead to unreliable evaluations.

The benchmark we propose in this chapter is designed to overcome these limitations by providing a comprehensive evaluation framework for hybrid CNN-ViT architectures trained on a large-scale dataset derived from ImageNet-1k. In contrast to most existing NAS benchmarks, which are restricted to small-scale datasets, the proposed benchmark comprises 6,561 distinct architectures each trained independently three times on large-scale data, enabling statistically robust performance estimates. This experimental protocol supports consistent and reliable comparisons across architectures and more accurately reflects the challenges encountered in realistic large-scale visual recognition scenarios. The comparison with representative NAS benchmarks reported in Table 5.1 highlights how HyViTas-Bench differs from prior evaluation resources.

Table 5.1: Summary of representative NAS benchmarks. HyViTas-Bench is the first benchmark in this set to address hybrid CNN-ViT architectures on large-scale data, while also providing out-of-distribution accuracy, hardware-related measurements, and precomputed training-free proxies within a unified evaluation framework.

Benchmark	# Arch.	Dataset(s)	Family	ID Accuracy	OoD Accuracy	HW metrics	Proxy metrics	Multiple runs
NAS-Bench-101 [143]	423k	CIFAR-10	CNN	✓	✗	✗	✗	✓
NAS-Bench-201 [27]	6.5k	CIFAR	CNN	✓	✗	✗	✗	✓
NATS-Bench [26]	6.5k + 32k	ImageNet-16	CNN	✓	✗	✓	✗	✓
HW-NAS-Bench [59]	6.5k	CIFAR	CNN	✓	✗	✓	✗	✗
LatBench [30]	15k	ImageNet-1k	CNN	✗	✗	✓	✗	✗
NAS-Bench-MR [24]	2.5k	ImageNet-1k, KITTI [34], Cityscapes [20]	CNN	✓	✗	✓	✗	✗
OoD-ViT-NAS [47]	3k	ImageNet-1k	ViT	✓	✓	✓	✓	✗
HyViTas-Bench	6.6k	RedImageNet	CNN ViT	✓	✓	✓	✓	✓

5.3 HyViTas-Bench

5.3.1 Search space

HyViTas-Bench defines a four-stage hybrid search space that couples convolutional and transformer-based blocks in a sequential design (Figure 5.1). The overall macro-architecture includes a fixed stem, implemented as a non-overlapping 4×4 convolution inspired by EdgeNeXt [82], and a fixed fully convolutional classification head. The searchable part is restricted to the four intermediate stages: the first two stages are purely convolutional, while the last two stages contain transformer-based blocks, enabling controlled exploration of CNN-ViT hybrid designs within a structured backbone template. The first two searchable and purely convolutional stages each consist of three ConvNeXt [73] blocks. ConvNeXt was proposed as a refinement of the residual convolutional paradigm introduced by ResNet, enriched with architectural choices inspired by Vision Transformers. In particular, it adopts large-kernel depth-wise convolutions and replaces traditional batch-normalization pipelines with Layer Normalization [3], while using GELU [46] as the non-linear activation. In our implementation, we replace Layer Normalization with standard Batch Normalization to achieve more efficient inference. The final two searchable stages are based on transformer blocks, each using a fixed configuration of 8 attention heads. Following the design proposed in [67], each transformer block is paired with a ConvNeXt block (Figure 5.2), thereby blending convolutional inductive biases with the global receptive field enabled by attention mechanisms. This configuration yields a search space that is both structured and practically usable

for hybrid CNN-Transformer architectures. It is deliberately designed to strike a balance between expressiveness and tractability: constraining the block choices within each stage avoids a combinatorial growth that would make exhaustive evaluation infeasible, while still supporting systematic architectural variations within the hybrid design. As a result, the benchmark enables controlled exploration of different capacity allocations and effective receptive field sizes, while maintaining a consistent stage-wise organization across candidate networks. The values considered for kernel sizes, expansion factors and output channel depths are summarized in Table 5.2.

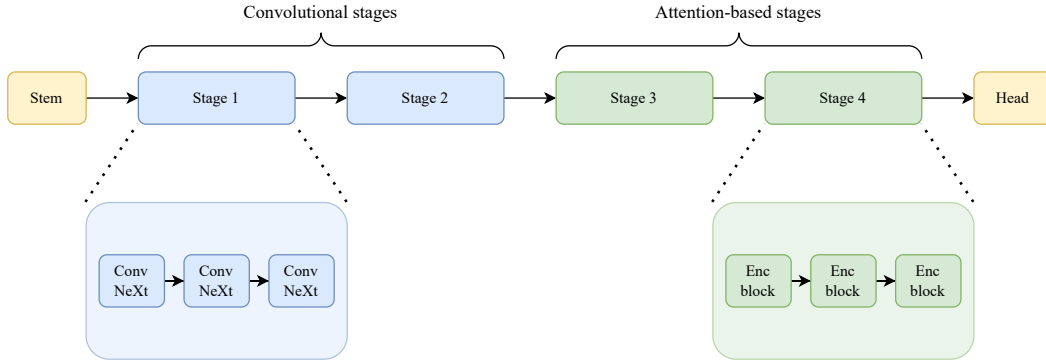


Figure 5.1: Schematic representation of the hybrid CNN-ViT backbone underlying the HyViTas-Bench search space. The network is organized into four stages and includes a fixed Stem module inspired by EdgeNeXt [82], two searchable convolutional stages built from ConvNeXt [73] blocks, two searchable transformer-based stages derived from LocalViT [67], and a fixed classification head. The search dimensions comprise kernel size, the expansion factor shared across stages of the same type, and the output depth, which is optimized independently for each stage. The structure of the encoder blocks is shown in Figure 5.2.

5.3.2 Datasets

The substantial computational cost associated with training large numbers of architectures on large-scale datasets has traditionally limited tabular NAS benchmarks to small-scale datasets, such as CIFAR-10 and CIFAR-100. To partially alleviate this burden, benchmarks such as NAS-Bench-201 and NATS-Bench adopt a heavily reduced variant of ImageNet-1k, namely ImageNet-16-120, obtained by downsampling images to a resolution of 16×16 pixels and restricting the dataset to 120 classes. While this design choice enables tractable large-scale benchmarking, the reliance on such simplified datasets constitutes a fundamental limitation, as

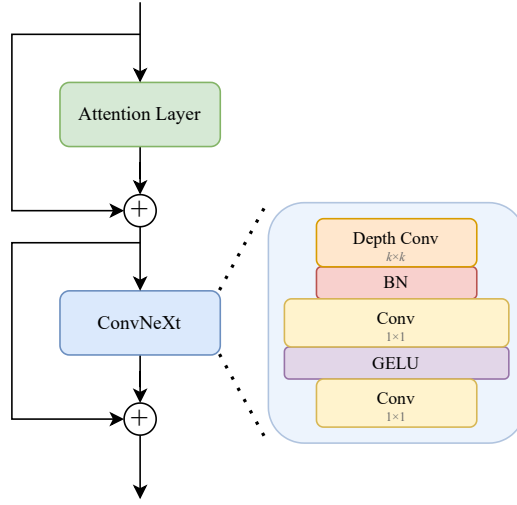


Figure 5.2: Structure of the transformer-based encoder block used in the attention-driven stages of the search space. The block consists of an attention module followed by a ConvNeXt [73] block, with residual connections around both components. The inset on the right details the ConvNeXt sub-block, which adopts an inverted bottleneck structure and includes a depthwise convolution.

Table 5.2: Searchable dimensions of the HyViTas-Bench search space. Kernel size and expansion factor are optimized independently for the first and last pairs of stages, yielding a search space containing 6,561 different architectures.

Dimension	Allowed values
Kernel size	[3, 5, 7]
Expansion factor	[1, 2, 4]
Output depth (stage 1)	[16, 32, 48]
Output depth (stage 2)	[64, 80, 96]
Output depth (stage 3)	[112, 128, 144]
Output depth (stage 4)	[192, 224, 256]

they often fail to capture the complexity required for reliable assessment of real-world generalization. In light of the increasing architectural complexity of modern neural networks, the evaluation of NAS methods on more realistic datasets that better approximate full-scale data distributions has therefore become increasingly important.

To fill this gap, the proposed benchmark evaluates candidate architectures on a large-scale dataset built following the best practices outlined in [90]. Specifically, the dataset is derived from ImageNet-1k by randomly selecting 500 classes and downscaling all images to a maximum resolution of 192×192 pixels. This design choice balances computational tractability and data representativeness, preserving a wide variety of visual statistics while remaining suitable for large-scale NAS experimentation. The resulting dataset comprises 523,901 training images and 25,000 validation images, with an overall storage footprint below 20 GB. While substantially smaller than the full ImageNet-1k dataset (approximately 146 GB), it remains considerably larger and more representative in both resolution and sample count than commonly used alternatives such as CIFAR or ImageNet-16-120.

By training all architectures on this dataset, the benchmark enables a more realistic evaluation of generalization performance, thereby reducing the gap between existing NAS benchmarks and practical deployment scenarios. This experimental setting facilitates more meaningful comparisons among NAS methods and supports the development of approaches tailored to large-scale, real-world applications.

To evaluate the robustness of each architecture under out-of-distribution conditions, all models are additionally tested on three auxiliary datasets derived from established OoD benchmarks. These datasets are constructed by applying the same reduction protocol adopted for the in-distribution training set, namely by selecting the same subset of 500 classes and downscaling all images to a maximum resolution of 192×192 pixels.

ImageNet-Sketch [132] is a domain generalization benchmark consisting of black-and-white, sketch-like depictions of classes drawn from ImageNet-1k. In contrast to natural photographs, these samples lack texture and color, therefore emphasizing a model’s ability to capture semantic information from shape and structural cues. Owing to its abstract and texture-free nature, the dataset poses a particularly challenging test for visual recognition models.

ImageNet-C [45] is a benchmark designed to assess robustness to input corruptions, in which 19 algorithmically generated perturbations, comprising 15 core and 4 additional types, are applied to the ImageNet-1k validation set. These corruptions span a wide range of visual degradations, including noise, blur, weather effects and digital artifacts, and are each defined at five increasing levels of severity. Within our benchmark, every architecture is evaluated across all severity levels, from level 1 (mild) to level 5 (severe), for each corruption type, enabling a comprehensive analysis of robustness under varying perturbation strengths. Unless otherwise stated, quantitative results reported for the reduced version of this dataset are averaged over the 15 core corruption types, while detailed per-corruption and per-severity outcomes are provided as part of the benchmark release.

Stylized-ImageNet [35] is constructed by altering the original textures of ImageNet-1k images through Adaptive Instance Normalization (AdaIN) [52] style transfer, using randomly sampled artistic styles while largely preserving object shapes and the overall spatial structure of the scenes. This transformation enforces a pronounced reliance on shape-based cues, making the dataset particularly challenging and the most demanding among those considered for robustness evaluation in our benchmark.

This additional evaluation protocol allows for a comprehensive analysis of architectural robustness and generalization beyond the training distribution, complementing in-distribution performance assessment and supporting the investigation of resilience to domain shift within NAS. In the remainder of this chapter, the reduced and computationally lighter versions of ImageNet-Sketch, ImageNet-C and Stylized-ImageNet validation sets are denoted by the prefix *Red-* (e.g. *RedImageNet-C*). For brevity in tables and figures, the out-of-distribution datasets are further abbreviated as *RedIN-Sketch*, *RedIN-C* and *RedStylized-IN*.

5.3.3 Training and evaluation protocol

All the architectures in HyViTas-Bench are trained on the RedImageNet dataset described in Section 5.3.2. Each model is trained for 200 epochs using the AdamW optimizer with a batch size of 1024. Low-fidelity training runs with a reduced number of epochs are not used as a substitute for full training, as such approximations may alter the relative ranking of candidate architectures; instead, the benchmark is intended to provide a reliable and stable ground-truth evaluation. The learning rate is scheduled according to a cosine annealing policy, with an initial value of 0.001, preceded by five warm-up epochs employing a linear increase from 5×10^{-5} . All experiments are conducted using mixed-precision training to enhance computational throughput and memory efficiency. Data augmentation follows the RandAugment [21] strategy, applied after a random resized crop to 160×160 pixels, with horizontal flipping performed with probability 0.5.

For every architecture, training is performed in three independent runs with different random seeds in order to capture variability arising from stochastic optimization and weight initialization, yielding a total of 19,683 trained models. Model performance is assessed on the validation set using both top-1 and top-5 accuracy metrics. All experiments are carried out with PyTorch 2.2 on a distributed computing cluster equipped with custom NVIDIA A100 GPUs.

Beyond accuracy-based metrics, the benchmark also reports inference latency for each architecture. As inference time is intrinsically dependent on the target

hardware, latency is measured across 19 heterogeneous platforms, including server-class and consumer GPUs, edge devices, and general-purpose CPUs based on both x86 and ARM architectures. For each device, latency is measured by executing 52 forward passes with a batch size of one image. The first two iterations are discarded to remove initialization effects such as kernel loading, clock ramp-up and memory allocation overhead. The final latency value is computed as the average over the remaining 50 runs. All measurements are performed using PyTorch 2.6 to ensure consistency across hardware platforms.

5.4 Benchmark evaluation

For a NAS benchmark to be meaningful, it must cover a broad set of architectures, spanning both high-performing and suboptimal designs. Although a general correlation between model capacity and accuracy is to be expected, the underlying search space should be constructed to prevent trivial optimization strategies, such as parameter count maximization, from consistently producing the best results. A well-designed search space encourages performance gains driven by substantive architectural decisions rather than by brute-force scaling alone.

To illustrate these aspects, Figure 5.3 depicts the relationship between model size and average validation accuracy across all architectures in the benchmark, with Pareto-optimal solutions highlighted in red. Although the largest model in the search space comprises approximately 4 M parameters, the highest accuracy is achieved by an architecture with only 2.9 M parameters, showing that superior performance does not necessarily require maximal model capacity. Figure 5.4 further complements this analysis by showing the distribution of validation accuracies together with their standard deviation, which is taken as an indicator of training stability.

Across the benchmark, the mean top-1 validation accuracy is 68.60%, with the majority of architectures achieving performance in the 60% – 74% range, and the best-performing model reaching 73.66%. A smaller subset of architectures exhibits markedly lower accuracy, clustering between 40% and 47%. These underperforming models share common structural characteristics, specifically an initial stage with an output dimension of 16, an expansion factor of 1 and a kernel size of 3, which collectively suggest limited representational capacity.

Training stability, quantified as the standard deviation across three independent runs, is generally high, with most architectures showing low variance ($\sigma \approx 0.2$), indicating consistent behavior across different random initializations. Nonetheless, a small fraction of models displays substantially higher variance, revealing sensitivity

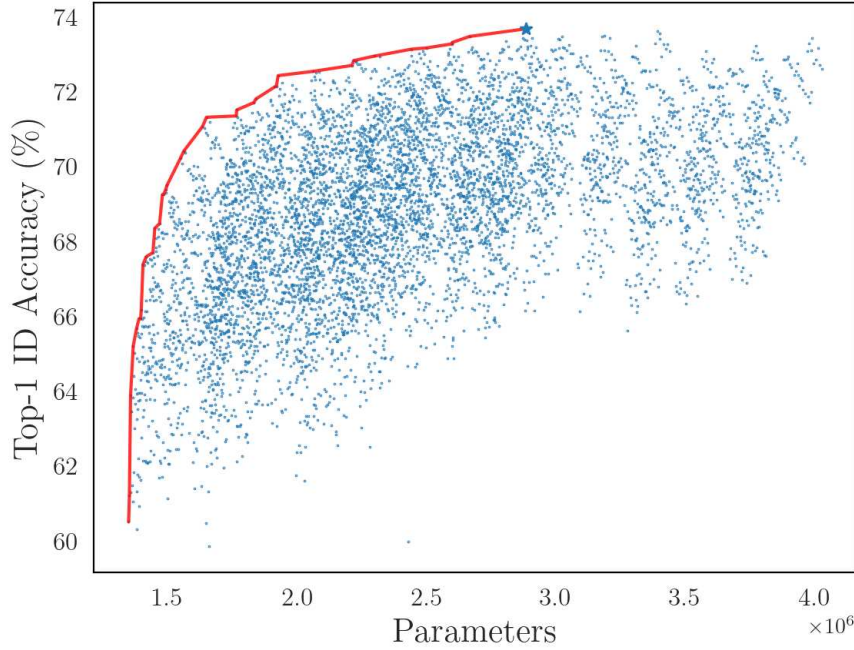


Figure 5.3: Relationship between model size and final validation accuracy for the architectures in the benchmark, with accuracy averaged over three training runs. Architectures with accuracy below 50% are omitted for visualization purposes, as they account for less than 1% of the search space. The highlighted curve denotes the Pareto front, while the star marker (★) identifies the best-performing architecture in the search space.

to initialization. In extreme cases, certain architectures are particularly difficult to optimize, frequently encountering divergence or NaN values during training, which points to poor trainability. These unstable architectures tend to exhibit the same structural patterns observed in low-performing models, where the combination of minimal output dimensionality, small kernel size, and low expansion factor in the initial stage likely acts as a training bottleneck.

To show that strong performance on small-scale datasets such as CIFAR-10 does not necessarily translate into optimal behavior on larger and more challenging datasets, we analyze the consistency of architectural ranking across datasets of increasing scale. To this end, we randomly select 20 architectures from the benchmark and train them on CIFAR-10, CIFAR-100, ImageNet-16-120, and the full ImageNet-1k dataset. All experiments follow the same protocol described in Section 5.3.3, with the number of training epochs adjusted to account for dataset size, namely 100 epochs for the three small-scale datasets and 300 epochs for ImageNet-1k.

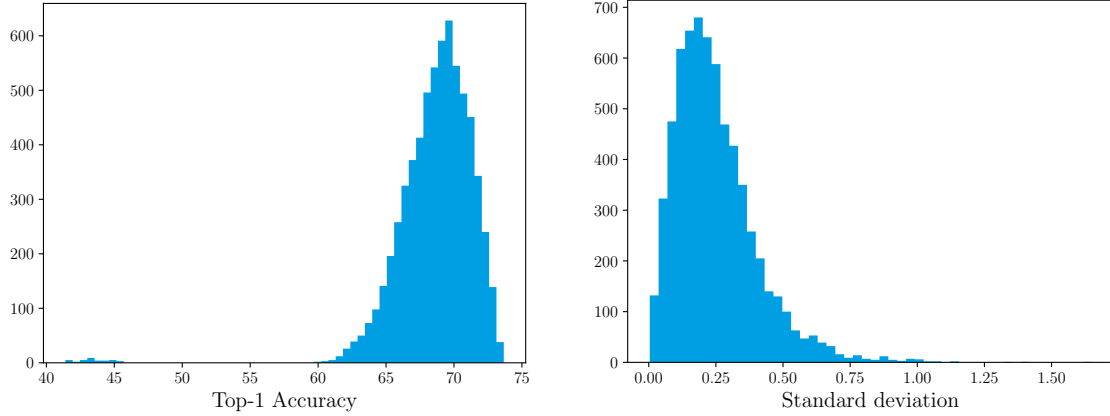


Figure 5.4: Distribution of the final validation top-1 accuracy across the architectures in the benchmark, reported as the average over three training runs, together with the corresponding standard deviation across runs. The left plot summarizes average performance, while the right plot shows sensitivity to random initialization, with lower standard deviations indicating more stable architectures. A limited number of outliers is omitted for visualization clarity.

The results reported in Table 5.3 show that the correlation between performance on ImageNet-1k and on the proxy dataset introduced in Section 5.3.2 is consistently higher than the correlations observed with smaller-scale datasets, including CIFAR-10, CIFAR-100 and ImageNet-16-120. To further evaluate the reliability of the proposed RedImageNet dataset as a proxy for large-scale performance, we consider a more demanding experimental scenario by randomly selecting 20 architectures from the top 15% of highest-performing models in the benchmark. This setting is especially demanding, since high-performing architectures often exhibit closely clustered accuracy values, making performance rankings more susceptible to noise. Notably, even under this challenging condition, the proposed proxy dataset preserves a statistically significant positive correlation with ImageNet-1k. In contrast, correlations obtained using other small-scale datasets degrade substantially, consistently yielding negative values. While previous works such as [90] have demonstrated that architectures discovered on small-scale datasets like CIFAR can achieve competitive absolute performance when transferred to large-scale benchmarks such as ImageNet-1k or even ImageNet-22k, our results highlight that these datasets do not reliably preserve relative performance rankings among architectures. This observation indicates that the proposed benchmark offers a more dependable and scalable proxy for ImageNet-1k performance, enabling meaningful comparative evaluation of architectures in large-scale scenarios while substantially reducing the computational cost of full ImageNet-1k training.

Table 5.3: Kendall- τ and Spearman- ρ correlations of architecture rankings between different datasets and ImageNet-1k, computed across 20 randomly sampled architectures.

Dataset	Random		Random (top 15%)	
	τ	ρ	τ	ρ
CIFAR-10	0.495	0.662	-0.221	-0.253
CIFAR-100	0.411	0.571	-0.147	-0.230
ImageNet-16-120	0.421	0.592	-0.105	-0.174
RedImageNet	0.758	0.898	0.332	0.503

5.4.1 Baseline experiments

To demonstrate the utility of the proposed benchmark, we evaluate it using six representative NAS baselines: Random Search, Regularized Evolution [100], REINFORCE [134], BANANAS [133], DARTS [69], and SPOS [40]. These methods are commonly employed in the NAS literature owing to their simplicity and strong performance on established benchmarks such as NAS-Bench-101 and NAS-Bench-201.

Random Search serves as a minimal reference baseline, in which candidate architectures are sampled uniformly from the search space. Despite its simplicity, this approach has been shown to be a surprisingly competitive strategy in constrained NAS scenarios. Considering the overall size of our search space, which comprises 6,561 architectures, we randomly sample 100 distinct architectures, evaluate their validation accuracy, and report the best observed performance.

Regularized Evolution with Aging (REA) employs a population-based search strategy in which new candidate architectures are obtained through mutations of existing ones, while the population is periodically updated by discarding the oldest exemplars. This approach has shown strong robustness and consistent performance across multiple NAS benchmarks, without relying on gradient information or surrogate models. In our experimental setting, we adopt a population size of 50 architectures and use tournaments of 10 candidates to select parent models for mutation. The total number of evaluated architectures is capped at 100.

REINFORCE is a policy-gradient method that models architectural decisions through a parametric probability distribution, which is iteratively updated to increase the likelihood of sampling high-performing architectures. At each step, an

architecture is drawn from the current policy, evaluated, and used to compute a gradient update driven by the observed reward. In our experiments, REINFORCE is executed with a fixed budget of 100 architecture evaluations, following the same evaluation protocol adopted for the other baseline methods.

BANANAS applies Bayesian optimization to the architecture search problem by training an ensemble of lightweight surrogate models on encoded representations of candidate architectures. These predictors are then used to identify promising designs. At each iteration, a pool of candidate architectures is generated by mutating the most promising models observed so far; all candidates are scored using the surrogate ensemble, and the top $k = 10$ architectures according to the acquisition function are selected for evaluation. The procedure is initialized with 10 randomly sampled architectures, and a candidate pool of 512 models is considered at each iteration. In our experiments, BANANAS is evaluated only for objectives that require training-based assessment, namely in-distribution or combined in-distribution and OoD accuracy. We do not apply BANANAS to training-free proxy objectives in zero-shot settings, as the surrogate models would be trained to approximate a quantity that is inexpensive to compute directly and already serves as a proxy for the true performance metric.

DARTS is a differentiable approach that replaces discrete architectural choices with continuous relaxations, thereby enabling optimization via gradient-based methods. It follows a bi-level optimization scheme in which the network weights θ are updated to minimize the training loss $\mathcal{L}_t(\theta, \alpha)$, while the architectural parameters α are optimized on a held-out validation split by minimizing $\mathcal{L}_v(\theta, \alpha)$. To apply DARTS within the HyViTas-Bench framework, we instantiate it in a layer-wise manner, where learnable logits α parameterize each of the eight architectural design choices listed in Table 5.2. Updates of θ and α are alternated during optimization, and a final discretization step is performed at the end of the search. The resulting architecture is then evaluated by querying its corresponding accuracy from the benchmark table.

SPOS operates by training a weight-sharing supernet in which a single candidate path, sampled uniformly from a search space, is activated per mini-batch. During training, only the parameters associated with the active path are updated. Once the supernet has been trained, individual architectures inherit the shared weights and are evaluated following a brief recalibration of the normalization layers. The final architecture is then identified using a lightweight evolutionary selection procedure.

We conduct experiments under two distinct evaluation settings:

- ID: candidate architectures are assessed exclusively on the RedImageNet validation set;
- ID+OoD: both in-distribution and out-of-distribution performance are considered. In this case, the search objective is defined as the sum of in-distribution accuracy and the average OoD accuracy, computed over RedImageNet-Sketch, RedStylized-ImageNet, and two severity levels of RedImageNet-C. Although the benchmark reports per-corruption results for all RedImageNet-C variants, in these experiments OoD performance is aggregated by averaging over the 15 core corruption types at each severity level.

Each experiment is repeated 500 times, and the mean performance together with the standard deviation is reported in the upper part of Table 5.4. An exception is made for the gradient-based methods DARTS and SPOS, which are executed only five times in the ID setting, due to the need to train a full supernet for each run. The results indicate that multi-shot approaches, namely Random Search, REA, REINFORCE and BANANAS, achieve competitive in-distribution accuracy, attaining performance levels close to those of the best architectures in the search space. It must be noted that without HyViTas-Bench, conducting these search experiments would entail a substantial burden, with an estimated cost of approximately 1,500 GPU-hours per run to train and evaluate up to 100 candidate architectures. Notably, REINFORCE does not demonstrate a clear advantage over Random Search, a finding that aligns with prior observations in the NAS literature [26], [63], where policy-gradient methods often fail to surpass random baselines under strict evaluation budgets due to noisy performance estimates and limited scope for effective policy updates.

In contrast, the disparity in out-of-distribution (OoD) performance remains substantially larger, highlighting a pronounced decoupling between in-distribution accuracy and robustness, as shown in Figure 5.5. Although it is well known that strong ID performance does not necessarily translate into improved OoD generalization, our findings underscore this mismatch in the specific setting of hybrid CNN-ViT architectures. Notably, despite being surpassed by multi-shot methods in terms of ID accuracy, one-shot gradient-based approaches achieve superior performance in most OoD evaluations, suggesting a greater capacity to identify architectures with enhanced robustness.

Due to the substantial computational burden imposed by conventional training-based NAS methods, recent work has increasingly explored alternative, computationally efficient strategies for estimating the potential performance of neural architectures without explicit training [1], [61], [71], [85]. These zero-shot approaches rely on intrinsic architectural characteristics to anticipate downstream performance, enabling fast exploration of large search spaces at a negligible computational cost.

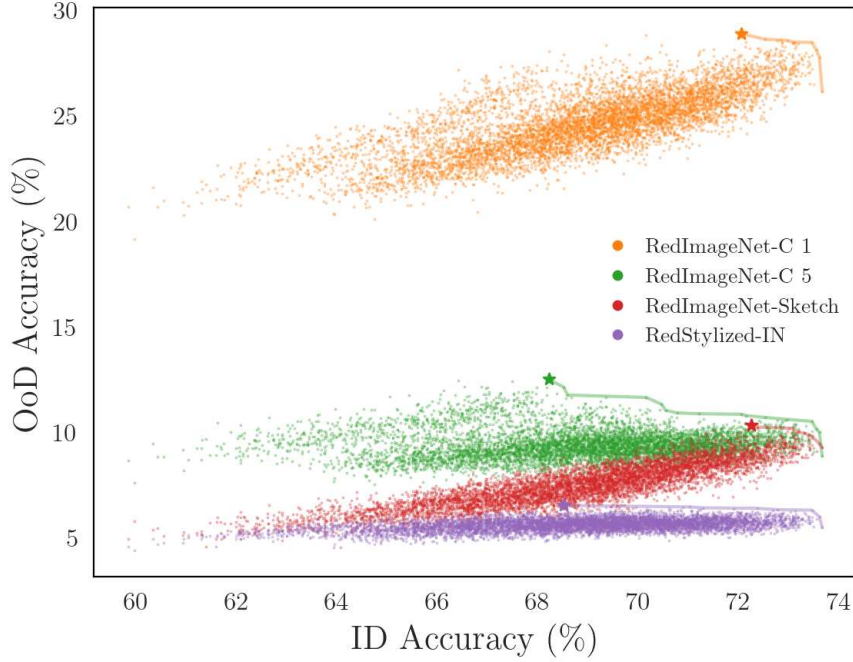


Figure 5.5: Relationship between in-distribution accuracy on RedImageNet and out-of-distribution accuracy on reduced versions of ImageNet-Sketch, Stylized-ImageNet and ImageNet-C at corruption severities 1 and 5. For RedImageNet-C, each accuracy value is averaged over the 15 corruption types. Star markers (★) denote the best-performing architectures for each OoD dataset, showing that the OoD optima generally differ from the architecture with the highest ID accuracy. The Pareto front is reported for each OoD setting. Kendall- τ correlations between ID and OoD accuracy are 0.68 for RedImageNet-Sketch, 0.25 for RedStylized-ImageNet, 0.57 for RedImageNet-C level 1, and 0.09 for RedImageNet-C level 5.

Motivated by these benefits, we incorporate several well-established training-free proxy metrics into our benchmark and precompute them for every architecture, thereby supporting efficient experimentation within hybrid CNN-ViT search spaces.

While several works on training-free proxies have focused on predicting neural network performance in in-distribution settings, far less attention has been paid to their ability to predict out-of-distribution generalization. More recent analyses, most notably [47], have begun to explicitly investigate this issue in the context of ViT-based architectures. Their results indicate that, within the considered ViT search space, widely used training-free metrics are surpassed by simple architectural attributes, such as parameter count or floating-point operations (FLOPs), when it comes to predicting generalization performance.

We build upon this line of investigation by conducting both correlation studies and full NAS experiments in the setting of hybrid CNN-ViT architectures included in the proposed HyViTas-Bench. Specifically, we report Kendall- τ and Spearman- ρ correlations between several training-free metrics and OoD accuracy (Table 5.5), and we execute three baseline search strategies 500 times for each metric, as summarized in the lower part of Table 5.4. As noted above, BANANAS is excluded from training-free experiments, since its surrogate would only approximate an already inexpensive proxy.

In line with the findings of [47], our results show that simple architectural attributes, such as model size, continue to display stronger correlations with accuracy on heavily corrupted datasets, including RedImageNet-C and RedStylized-ImageNet. Despite this general trend, we identify meaningful deviations when considering RedImageNet-Sketch, a dataset dominated by high-level semantic shifts rather than low-level image corruptions. In this setting, activation-based proxies such as NASWOT [85], as well as gradient-based metrics including SNIP [58] and Synflow [123], exhibit stronger correlations with OoD accuracy than simpler measures that rely exclusively on model size. These results indicate that training-free metrics which capture functional and optimization-related properties of network architectures can provide clear benefits under specific forms of domain shift, highlighting a promising direction for future research toward more context-aware and effective proxy metrics for robust NAS. It must be noted that, despite its simplicity, the number of parameters consistently emerges as a strong predictor of OoD accuracy across our experiments, outperforming more sophisticated proxies in several settings in terms of correlation. Interestingly, however, when training-free metrics are directly employed as optimization objectives in NAS experiments, this proxy is surpassed by NASWOT and SNIP, suggesting that a higher overall correlation with accuracy does not necessarily correspond to more effective search behavior.

Table 5.4: Top-1 in-distribution and out-of-distribution accuracies, reported as mean $\pm$ standard deviation, for Random Search, Regularized Evolution (REA), REINFORCE, and BANANAS over 500 independent trials and under different optimization objectives. Accuracy-driven searches optimize either the in-distribution validation accuracy alone or a joint “ID+OoD” criterion, computed as the average between the ID accuracy and the mean accuracy on four OoD evaluation sets: RedImageNet-C at corruption levels 1 and 5, RedImageNet-Sketch, and RedStylized-ImageNet. For zero-shot NAS, the optimization objective is instead given by standard training-free proxies. The table also includes the one-shot methods DARTS and SPOS, evaluated over five runs with ID accuracy as target. The “Optimum” row reports, for each dataset, the best architecture available in the benchmark, while “Search space mean” summarizes the average accuracy and standard deviation over the entire search space.

Method	Target	ID Accuracy	OoD Accuracy			
			RedIN-C 1	RedIN-C 5	RedIN-Sketch	RedStylized-IN
Accuracy-based						
Random Search	ID	73.03 $\pm$ 0.40	27.22 $\pm$ 0.70	9.70 $\pm$ 0.39	9.20 $\pm$ 0.42	5.80 $\pm$ 0.21
REA		73.36 $\pm$ 0.32	27.35 $\pm$ 0.67	9.69 $\pm$ 0.44	9.33 $\pm$ 0.32	5.82 $\pm$ 0.21
REINFORCE [134]		73.00 $\pm$ 0.39	27.18 $\pm$ 0.69	9.68 $\pm$ 0.41	9.21 $\pm$ 0.42	5.79 $\pm$ 0.22
BANANAS [133]		73.44 $\pm$ 0.29	27.29 $\pm$ 0.69	9.64 $\pm$ 0.46	9.32 $\pm$ 0.30	5.80 $\pm$ 0.21
DARTS [69]		72.89 $\pm$ 0.24	27.42 $\pm$ 0.34	9.94 $\pm$ 0.34	8.86 $\pm$ 0.27	5.85 $\pm$ 0.21
SPOS [40]		71.92 $\pm$ 1.00	27.48 $\pm$ 0.67	10.08 $\pm$ 0.67	9.03 $\pm$ 0.32	6.00 $\pm$ 0.21
Random Search	ID + OoD	72.97 $\pm$ 0.42	27.59 $\pm$ 0.59	9.93 $\pm$ 0.38	9.33 $\pm$ 0.39	5.90 $\pm$ 0.20
REA		73.26 $\pm$ 0.32	28.03 $\pm$ 0.42	10.15 $\pm$ 0.32	9.51 $\pm$ 0.36	6.02 $\pm$ 0.18
REINFORCE		72.97 $\pm$ 0.41	27.57 $\pm$ 0.59	9.92 $\pm$ 0.37	9.32 $\pm$ 0.38	5.90 $\pm$ 0.21
BANANAS		73.32 $\pm$ 0.31	28.07 $\pm$ 0.38	10.18 $\pm$ 0.29	9.55 $\pm$ 0.34	6.04 $\pm$ 0.18
Zero-shot						
Random Search	NASWOT [85]	72.98 $\pm$ 0.42	27.30 $\pm$ 0.70	9.72 $\pm$ 0.39	9.11 $\pm$ 0.36	5.74 $\pm$ 0.21
REA		73.18 $\pm$ 0.38	27.80 $\pm$ 0.59	9.88 $\pm$ 0.32	9.22 $\pm$ 0.29	5.81 $\pm$ 0.16
REINFORCE		72.96 $\pm$ 0.42	27.21 $\pm$ 0.72	9.68 $\pm$ 0.39	9.12 $\pm$ 0.41	5.71 $\pm$ 0.22
Random Search	Synflow [123]	71.78 $\pm$ 1.05	26.75 $\pm$ 0.83	9.75 $\pm$ 0.44	8.53 $\pm$ 0.54	5.70 $\pm$ 0.25
REA		72.46 $\pm$ 0.56	27.34 $\pm$ 0.64	9.90 $\pm$ 0.35	8.76 $\pm$ 0.30	5.62 $\pm$ 0.25
REINFORCE		71.84 $\pm$ 0.99	26.77 $\pm$ 0.81	9.75 $\pm$ 0.43	8.58 $\pm$ 0.48	5.70 $\pm$ 0.24
Random Search	LogSynflow [11]	72.02 $\pm$ 0.89	26.83 $\pm$ 0.82	9.75 $\pm$ 0.40	8.62 $\pm$ 0.44	5.68 $\pm$ 0.23
REA		72.46 $\pm$ 0.52	27.28 $\pm$ 0.71	9.85 $\pm$ 0.39	8.69 $\pm$ 0.31	5.59 $\pm$ 0.24
REINFORCE		72.04 $\pm$ 0.86	26.85 $\pm$ 0.84	9.76 $\pm$ 0.40	8.63 $\pm$ 0.44	5.69 $\pm$ 0.24
Random Search	DSS [23]	71.67 $\pm$ 1.02	26.69 $\pm$ 0.85	9.75 $\pm$ 0.43	8.52 $\pm$ 0.50	5.72 $\pm$ 0.25
REA		72.33 $\pm$ 0.66	27.40 $\pm$ 0.60	9.96 $\pm$ 0.35	8.73 $\pm$ 0.30	5.67 $\pm$ 0.25
REINFORCE		71.65 $\pm$ 1.05	26.71 $\pm$ 0.83	9.76 $\pm$ 0.43	8.48 $\pm$ 0.50	5.73 $\pm$ 0.24
Random Search	Jacobian [49]	70.01 $\pm$ 2.02	25.48 $\pm$ 1.16	9.66 $\pm$ 0.41	7.83 $\pm$ 0.93	5.71 $\pm$ 0.20
REA		70.34 $\pm$ 1.43	25.58 $\pm$ 0.92	9.61 $\pm$ 0.33	8.08 $\pm$ 0.63	5.78 $\pm$ 0.22
REINFORCE		69.91 $\pm$ 2.08	25.49 $\pm$ 1.13	9.68 $\pm$ 0.42	7.83 $\pm$ 0.93	5.71 $\pm$ 0.20
Random Search	SNIP [58]	72.33 $\pm$ 0.68	27.09 $\pm$ 0.79	9.83 $\pm$ 0.41	8.84 $\pm$ 0.44	5.76 $\pm$ 0.23
REA		72.64 $\pm$ 0.45	27.56 $\pm$ 0.53	9.97 $\pm$ 0.30	8.78 $\pm$ 0.28	5.67 $\pm$ 0.24
REINFORCE		72.40 $\pm$ 0.68	27.11 $\pm$ 0.79	9.83 $\pm$ 0.41	8.84 $\pm$ 0.42	5.75 $\pm$ 0.25
Random Search	# Parameters	71.35 $\pm$ 1.32	26.97 $\pm$ 0.78	10.03 $\pm$ 0.61	8.33 $\pm$ 0.70	5.83 $\pm$ 0.23
REA		72.40 $\pm$ 0.82	27.37 $\pm$ 0.58	9.88 $\pm$ 0.39	8.82 $\pm$ 0.47	5.67 $\pm$ 0.23
REINFORCE		71.30 $\pm$ 1.26	26.92 $\pm$ 0.87	10.01 $\pm$ 0.55	8.32 $\pm$ 0.71	5.81 $\pm$ 0.24
Search space mean	-	68.60 $\pm$ 3.26	24.52 $\pm$ 1.64	9.39 $\pm$ 0.72	7.48 $\pm$ 1.00	5.59 $\pm$ 0.32
Optimum	-	73.66	28.86	12.49	10.33	6.52

Table 5.5: Kendall- τ and Spearman- ρ rank correlations between several training-free proxies and the final top-1 accuracy, evaluated both in-distribution on RedImageNet validation and under out-of-distribution shifts, including RedImageNet-C across all corruption severities, RedImageNet-Sketch, and RedStylized-ImageNet. NASWOT shows the strongest correlation with in-distribution accuracy, but it decreases under high-severity corruptions (RedImageNet-C 5), while remaining comparatively informative for domain-shifted data such as RedImageNet-Sketch.

Setting		NASWOT	Synflow	LogSynflow	DSS	Jacobian	SNIP	Entropic	# Parameters
ID	τ	0.65	0.44	0.50	0.44	-0.01	0.54	-0.05	0.37
	ρ	0.83	0.62	0.69	0.63	-0.01	0.73	-0.07	0.54
RedIN-C 1	τ	0.41	0.41	0.47	0.37	0.06	0.43	-0.06	0.55
	ρ	0.56	0.57	0.65	0.53	0.09	0.60	-0.08	0.74
RedIN-C 2	τ	0.33	0.36	0.42	0.33	0.09	0.38	-0.06	0.52
	ρ	0.46	0.51	0.58	0.46	0.13	0.53	-0.09	0.71
RedIN-C 3	τ	0.27	0.32	0.38	0.28	0.09	0.38	-0.06	0.52
	ρ	0.37	0.45	0.52	0.40	0.14	0.46	-0.09	0.68
RedIN-C 4	τ	0.15	0.24	0.29	0.20	0.10	0.24	-0.06	0.44
	ρ	0.20	0.34	0.40	0.27	0.15	0.31	-0.09	0.62
RedIN-C 5	τ	0.01	0.14	0.18	0.09	0.11	0.11	-0.06	0.37
	ρ	-0.01	0.20	0.24	0.11	0.16	0.12	-0.08	0.53
RedIN-Sketch	τ	0.52	0.33	0.38	0.38	0.02	0.52	-0.03	0.22
	ρ	0.70	0.49	0.55	0.55	0.03	0.71	-0.05	0.32
RedStylized-IN	τ	0.18	0.21	0.25	0.21	0.12	0.25	-0.14	0.33
	ρ	0.26	0.30	0.37	0.31	0.19	0.36	-0.21	0.47

The Jacobian-based proxy [49] exhibits consistently weak rank correlation with the performance of hybrid CNN-ViT architectures. This behavior is consistent with prior evidence [47], showing that metrics originally introduced to assess adversarial robustness in CNNs do not generalize well to transformer-based models. Notably, Entropic Score, introduced in Section 4.4, was not included among the precomputed metrics released with the benchmark and was not used for the search experiments, since it was specifically designed to capture topological variations, while the HyViTas-Bench search space is mostly size-based. For completeness, however, we report its correlation values in Table 5.5. The weak overall correlation observed here is consistent with the poor performance of Entropic Score on the size-based search space reported in Table 4.11b. To further investigate this behavior, we analyzed Entropic Score separately with respect to different architectural decisions. As discussed in Section 4.4, Entropic Score was shown to be substantially more informative for topological rather than for dimensional variations. In HyViTas-Bench, most searchable variables affect model dimensionality, while kernel size represents the main topology-related choice. We therefore partitioned the search space by fixing all architectural variables except one family at a time and computed the rank correlation between Entropic Score and final accuracy within each resulting subspace. When varying kernel size while keeping the remaining choices fixed, this procedure produces 729 subspaces, each containing the nine possible kernel-size configurations. The resulting local correlations show an overall positive tendency. Conversely, when the same analysis is performed by varying the expansion factors or output dimensions, the correlation distributions shift toward negative values. The corresponding mean Kendall- τ and Spearman- ρ correlations, together with the proportion of positive values, are summarized in Table 5.6, while Figure 5.6 shows the distributions for three families of architectural choices. Although this analysis does not establish a causal relationship between the composition of the search space and the global performance of the proxy, it provides additional evidence consistent with the interpretation that Entropic Score primarily captures topology-related variations and is considerably less informative when architectural differences are predominantly dimensional.

These findings further suggest that the effectiveness of NAS strategies and training-free evaluation metrics cannot be taken for granted across heterogeneous architectural search spaces and evaluation settings. Overall, our results emphasize the importance of validating such approaches on diverse architectures and benchmarks in order to draw more reliable conclusions about their generalization properties.

Table 5.6: Mean Kendall- τ and Spearman- ρ correlations between Entropic Score and final accuracy across subspaces obtained by varying one family of architectural choices at a time while fixing the others. Standard deviations across subspaces are reported together with the mean values, along with the percentage of positive correlations.

Family		Rank correlation	Positive values (%)
Kernel size	τ	0.164 ± 0.242	70.65
	ρ	0.215 ± 0.319	73.94
Expansion factor	τ	-0.196 ± 0.318	24.69
	ρ	-0.262 ± 0.410	26.06
Output dimension	τ	-0.181 ± 0.094	6.17
	ρ	-0.265 ± 0.136	6.17

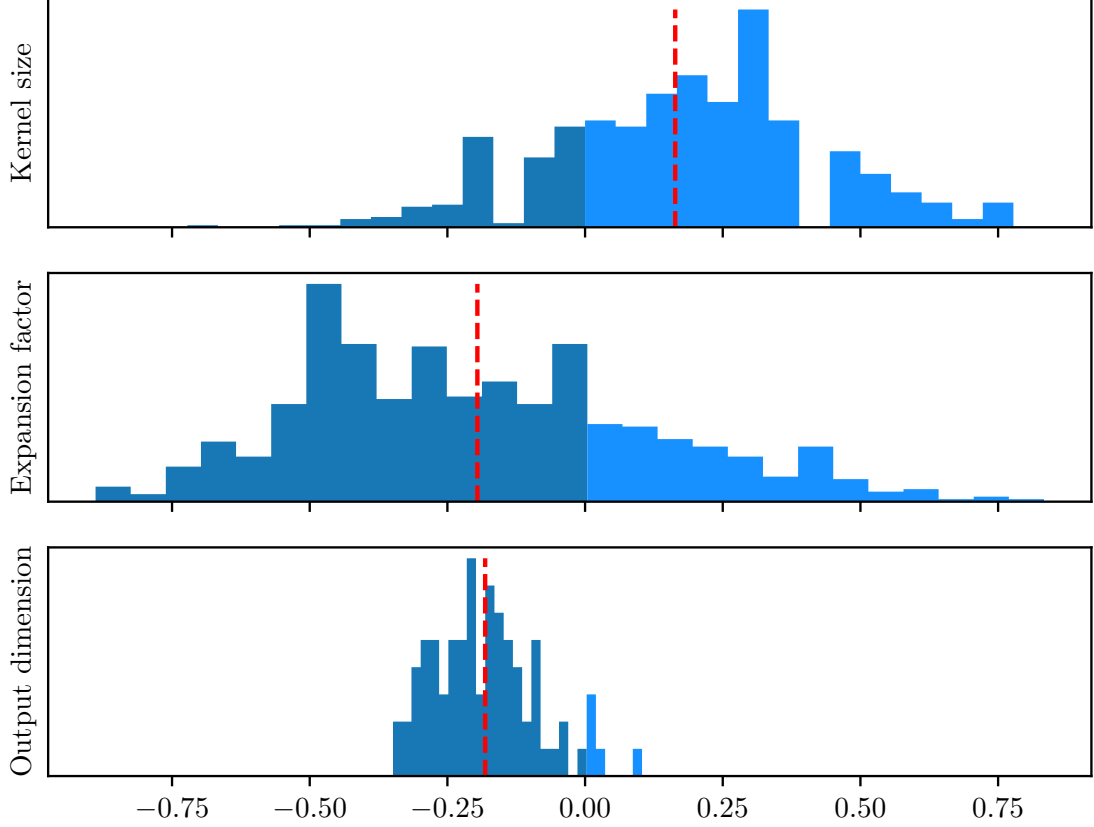


Figure 5.6: Distribution of Kendall- τ rank correlations between Entropic Score and final accuracy across the subspaces obtained by fixing all architectural choices except the considered family. Lighter bars highlight positive correlation values, while the red dashed line denotes the mean correlation for each distribution. Kernel-size variations mostly exhibit positive correlations, whereas the other architectural choices are shifted towards negative correlation values.

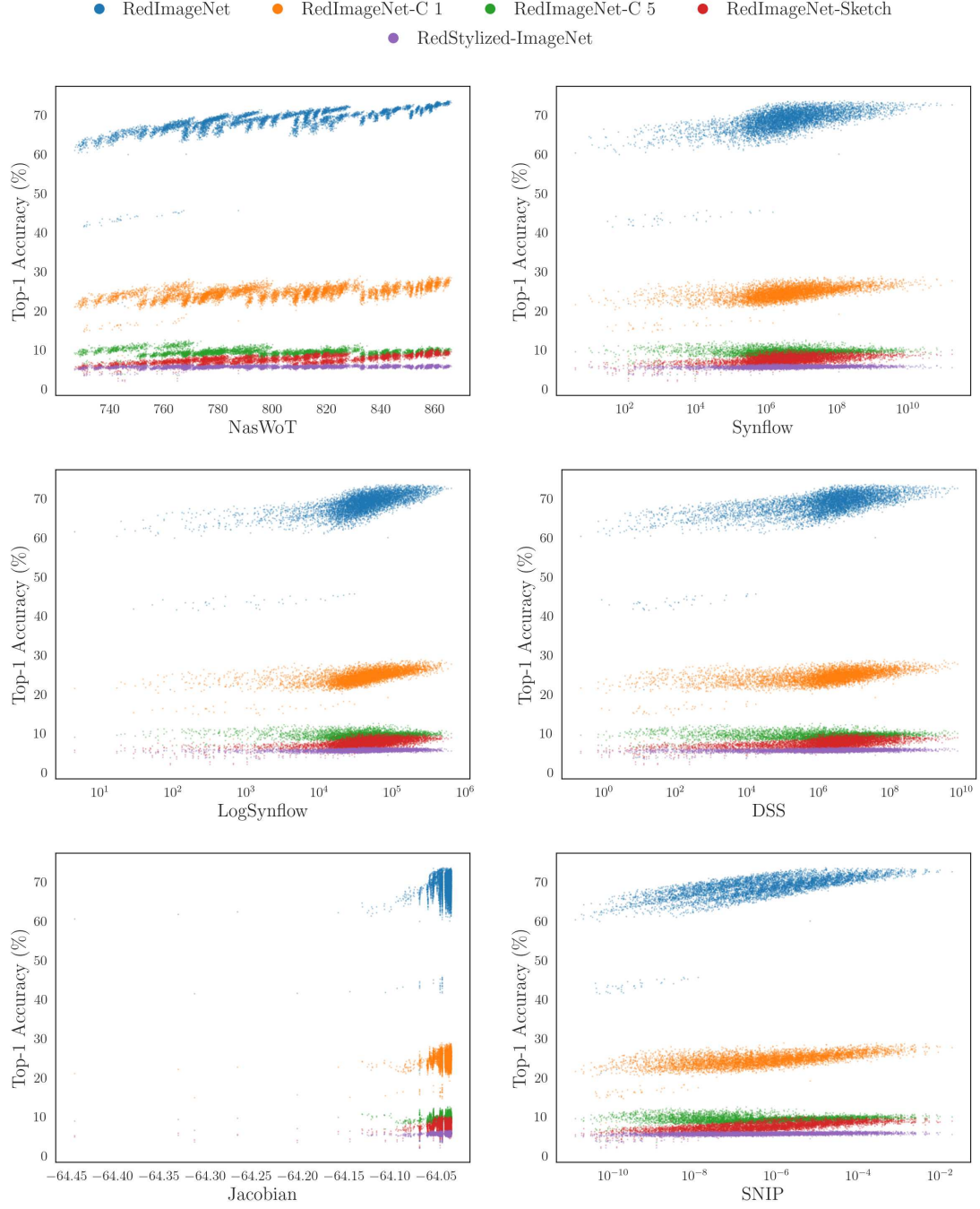


Figure 5.7: Scatter plots relating six training-free metrics to the corresponding top-1 accuracy of the architectures included in HyViTas-Bench, considering both in-distribution and out-of-distribution evaluations. Each point represents one architecture under a specific evaluation setting, while colors distinguish the in-distribution dataset from the considered OoD benchmarks.

5.4.2 Accuracy and latency across hardware platforms

The analyses presented in the previous section focus on accuracy and robustness. However, one of the goals of the benchmark is to support a broader characterization of candidate models, including deployment-related properties. This aspect is particularly relevant for NAS, since the architecture selected by a search algorithm is often expected not only to achieve high predictive performance, but also to satisfy practical constraints imposed by the target hardware platform. For this reason, we complement the previous evaluation with an analysis of inference latency measurements collected across several hardware devices. We include the latency measurements as part of the benchmark.

As described in Section 5.3.3, latency is measured for each architecture using single-image inference. This setting is especially relevant for edge and interactive applications, where the objective is often to minimize response time rather than maximize the throughput over large batches. Since inference time is not an intrinsic property of the architecture alone, but depends on the interaction between the network structure, the execution framework, and the hardware backend, the benchmark reports measurements on a heterogeneous set of platforms. These include low-power edge devices, embedded GPUs, general-purpose CPUs, consumer GPUs, and server-grade accelerators. A summary of the latency statistics is reported in Table 5.7.

Table 5.7: Inference latency measured on a representative subset of the hardware platforms included in HyViTas-Bench. For each device, latency is reported in milliseconds as the mean, standard deviation, minimum, and maximum across all architectures in the search space. The selected devices cover the main hardware classes considered in the benchmark and highlight that single-image inference does not always follow hardware tier, with the RTX 5090 achieving lower latency than the H100.

Type	Device	Inference time (ms)			
		Mean	Std.	Min	Max
Edge CPU	Raspberry Pi 4	125.037	15.992	89.899	218.476
Edge GPU	NVIDIA Jetson Orin Nano	18.806	0.716	17.349	20.342
Workstation CPU	Intel i9 10920X	9.590	0.723	7.524	12.366
Workstation CPU	AMD Threadripper 5995WX	9.895	0.894	8.054	12.943
Mobile GPU	NVIDIA GTX 1650 Max-Q	5.858	0.318	4.998	7.566
Consumer GPU	NVIDIA RTX 5090	1.508	0.079	1.335	1.677
Server GPU	NVIDIA H100	3.034	0.155	2.697	3.313

Figure 5.8 illustrates the relationship between top-1 in-distribution accuracy

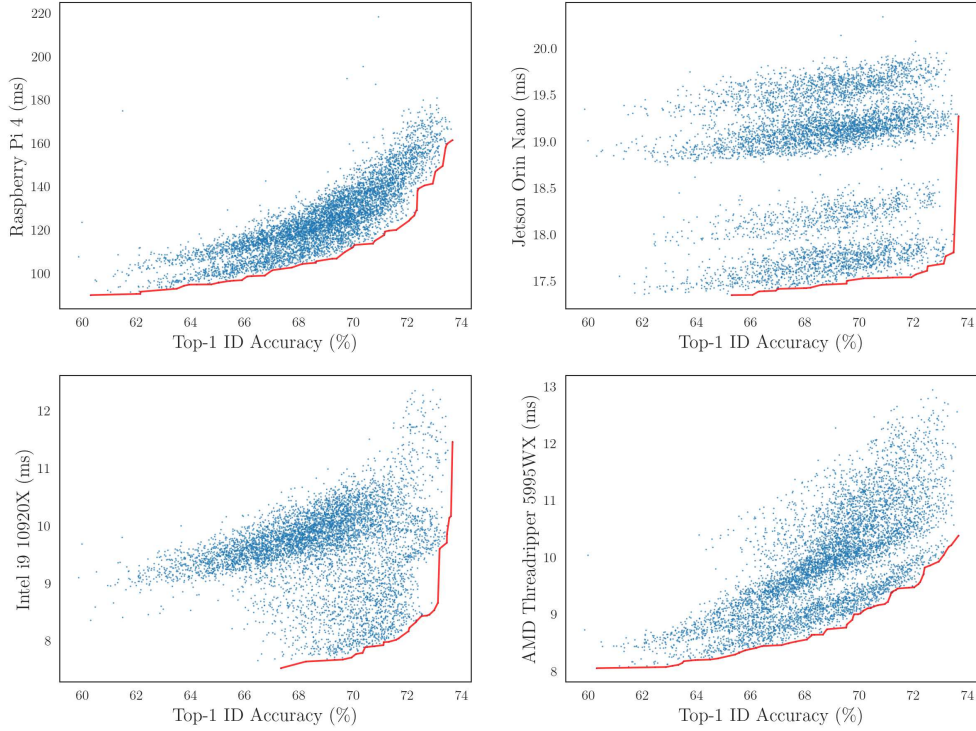


Figure 5.8: Relationship between in-distribution top-1 accuracy and inference latency across representative hardware platforms. Each point corresponds to one architecture in the search space, while the highlighted curve denotes the Pareto front. The plots show that different platforms induce different latency distributions and Pareto-optimal sets.

and inference latency for a representative subset of devices. The accuracy-latency scatter plots further show that the trade-off between predictive performance and inference speed is hardware-dependent. Although architectures achieving higher accuracy tend to require higher compute time, the relationship is not perfectly monotonic. Several architectures obtain similar accuracy while exhibiting different latency values, and the shape of the Pareto front changes across devices. This means that the most attractive architecture depends on the deployment target: a model that lies close to the Pareto front on one platform may be less competitive on another.

An interesting observation emerges when comparing consumer and server-grade GPUs. In the single-image inference setting considered here, high-end consumer GPUs can achieve lower latency than datacenter accelerators. This behavior is consistent with the fact that server GPUs are primarily designed for high-throughput workloads and large-scale parallel execution, while compact models evaluated on single images may not fully exploit their computational capacity. As a result, the

advantage of server-grade accelerators becomes less pronounced in this setting. This suggests that single-sample latency should be evaluated directly on the target platform, as hardware class alone is not always sufficient to predict response time in the intended execution setting.

These observations motivate the inclusion of measured hardware latency in HyViTas-Bench. Even when latency is not the primary objective of the analysis, it provides an additional view of the search space and allows accuracy-oriented results to be interpreted in relation to concrete deployment costs.

5.5 Conclusions

This chapter introduced HyViTas-Bench, a benchmark designed to support the evaluation of NAS methods under more realistic conditions than those typically considered in compact tabular benchmarks. Rather than focusing only on in-distribution accuracy, the benchmark exposes several complementary aspects of architecture quality, including robustness to distribution shift, stability across repeated training runs, training-free metrics, and inference latency on heterogeneous hardware platforms.

The experimental analysis highlights two main points. First, the choice of dataset used for benchmark evaluation has a substantial impact on the conclusions that can be drawn about architecture quality. The proposed RedImageNet dataset preserves the ranking of architectures with respect to full ImageNet-1k more reliably than smaller proxy datasets, especially near the top of the search space, where NAS decisions are most sensitive to ranking noise. This supports the need for benchmarks that remain computationally tractable while retaining sufficient scale and representativeness.

Second, the results show that architecture evaluation cannot be reduced to a single criterion. High in-distribution accuracy does not necessarily identify the most robust models, and the behavior of training-free proxies varies depending on the type of distribution shift. Similarly, the latency analysis confirms that deployment cost cannot be fully captured by architecture-level quantities alone. These observations reinforce the role of HyViTas-Bench as a testbed for studying NAS methods beyond standard accuracy-driven evaluation.

As with any NAS benchmark, the findings reported in this chapter are necessarily tied to the scope of the considered evaluation setting: HyViTas-Bench defines a structured search space based on a specific family of hybrid architectures and

focuses on image classification. Therefore, the behavior observed for search algorithms, training-free proxies, as well as the robustness patterns identified in the benchmark, should not be assumed to generalize to substantially different architecture families, search spaces, or tasks. This motivates the complementary use of benchmarks covering different architectural families and experimental settings when assessing the generality of NAS methods. Moreover, although the considered OoD datasets cover different forms of distribution shift, they do not provide an exhaustive characterization of model robustness. The adoption of a fixed training protocol for all candidate architectures enables controlled comparisons, but does not guarantee that each architecture is evaluated under its individually optimal training configuration.

A further consideration concerns the long-term evolution of the benchmark. Hardware latency measurements inherently depend on both the target platform and the associated software stack, and their relevance may change as new devices, runtimes, and architectural paradigms emerge. Future extensions could also include measurements under quantized inference, which is not considered in the current version of the benchmark and may provide a more representative characterization of deployment efficiency, particularly on edge platforms. Maintaining the benchmark over time may therefore require extending it with additional hardware platforms and architectural families, as its continued representativeness will be a factor determining its long-term relevance and adoption within the NAS community.

Overall, HyViTas-Bench complements the methodological contributions discussed in the previous chapters by addressing the evaluation side of practical NAS and by providing a controlled but more realistic environment in which search algorithms, training-free proxies, and robustness or deployment-aware objectives can be compared consistently.

Chapter 6

Conclusions

This thesis investigated Neural Architecture Search from a practical perspective. Rather than treating NAS only as a generic optimization problem over a predefined search space, the dissertation focused on the conditions that make architecture search meaningful in realistic scenarios. In particular, the work addressed three complementary aspects of practical NAS: the lack of a standard supervision signal, the computational cost of candidate evaluation, and the need for benchmarks that better reflect modern architectures and deployment-oriented criteria.

6.1 Summary of contributions

The first contribution of the thesis concerns Neural Architecture Search under limited target supervision. In standard NAS, candidate architectures are usually selected by measuring their validation performance on the target task. This assumption does not hold in Unsupervised Domain Adaptation (UDA), where target-domain labels are unavailable. At the same time, practical UDA pipelines often rely on pretrained backbones, making full-network search undesirable. To address these constraints, the thesis introduced Adversarial Branch Architecture Search (ABAS), which restricts the search to the auxiliary adversarial branch attached to a pretrained model, jointly optimizing its architecture and hyperparameters. A key component of this contribution is the Ensemble Model Selection (EMS) strategy, which replaces unavailable target accuracy with a combination of complementary unsupervised signals computed mostly on target-domain features. EMS also shows that a performance predictor learned from these signals in one experimental setting

can be transferred to a different dataset, where it still provides a meaningful ranking of candidate models despite the absence of target labels. Coupled with multi-fidelity Bayesian optimization, this estimator enables architecture search without target supervision. The resulting framework shows that the design of the adversarial branch is highly setting-dependent and that automatically adapting it can consistently improve existing UDA methods.

The second contribution of the thesis concerns NAS under limited computational budget. A central limitation of conventional NAS is that evaluating candidate architectures often requires repeated training, which makes search computationally expensive and, in many cases, incompatible with the goal of designing efficient models for constrained scenarios. This issue is especially relevant when the target architectures are intended for small-scale or resource-limited deployments, since the cost of the search itself may undermine the practical value of the final model.

Within this context, the thesis studied training-free NAS, where architectures are scored at initialization through zero-cost proxies rather than through full or partial training. A first contribution in this direction was to show that a carefully designed combination of training-free metrics, embedded into an appropriate evolutionary search strategy, can already make training-free NAS highly competitive. This result is important because it demonstrates that the use of proxy signals does not necessarily imply a severe loss in search quality. On the contrary, when the search procedure and the evaluation signals are properly aligned, training-free NAS can provide a strong compromise between efficiency and effectiveness, including in constrained settings.

A further contribution was to move beyond the idea of a single aggregated ranking score and to argue that different architectural properties should be guided by different training-free signals. In particular, the thesis emphasized the distinction between topological and dimensional design choices. These two aspects of architecture design affect the model in different ways and are not necessarily captured equally well by the same proxy. Starting from this observation, the thesis introduced Entropic Score as a topology-oriented metric and combined it with LogSynflow within a decoupled search strategy, where topology and size are optimized separately but cooperatively. This represents a more structured view of training-free NAS: zero-cost proxies should not be treated as interchangeable approximations of final accuracy, but as signals whose usefulness depends on the architectural dimension they are meant to supervise.

More broadly, the progression from the first training-free method to the later decoupled formulation supports a general methodological conclusion: practical training-free NAS does not emerge simply from replacing training with a cheaper

scalar estimator. Its effectiveness depends on understanding what each proxy captures, how it should be combined with other signals, and how it should be integrated into the search process. In this sense, the thesis contributes both concrete methods and a more refined interpretation of proxy-based architecture evaluation.

The third contribution of the thesis concerns realistic benchmarking and evaluation for NAS. Even when search methods become more efficient, their assessment remains strongly dependent on the benchmarks used to evaluate them. Much of the NAS literature in computer vision has historically relied on small-scale datasets and search spaces that are predominantly convolutional and only partially representative of modern visual architectures. This creates a mismatch between methodological progress and the setting in which NAS is expected to be useful. To address this issue, the thesis introduced a benchmark specifically designed for hybrid CNN-ViT architectures and built on a reduced yet still large-scale ImageNet-based dataset. This contribution broadens the evaluation setting along several dimensions. First, it considers hybrid architectures rather than only classical CNN search spaces. Second, it moves toward a more realistic data scenario than standard small-scale NAS benchmarks. Third, it includes multiple evaluation criteria beyond in-distribution accuracy, namely latency on heterogeneous hardware platforms and performance under out-of-distribution conditions. This makes it possible to study architectural quality as a multidimensional notion, rather than reducing it to a single benchmark score.

An important outcome of this part of the thesis is the observation that different evaluation criteria do not necessarily induce the same ranking over architectures. In-distribution accuracy, out-of-distribution robustness, model size, and latency only partially align. Similarly, the behavior of training-free proxies can change depending on the architectural space and on the target evaluation dimension. This reinforces one of the core messages of the dissertation: NAS should not be considered only as the optimization of validation accuracy on a simplified benchmark, but as the design of architectures under realistic constraints and according to evaluation criteria that are meaningful for deployment.

Taken together, the contributions of this thesis define a coherent perspective on practical NAS. The first part showed that architecture search can be adapted to settings where the standard supervision signal is unavailable. The second part showed that NAS can be made substantially more efficient through training-free evaluation, especially when proxy signals are used in a structured manner. The third part showed that these methodological advances require stronger and more realistic evaluation frameworks. Across all these directions, the common objective was to move NAS closer to the conditions under which architecture search is actually expected to operate in practice.

6.2 Limitations and future work

Despite these contributions, the work presented in this thesis has limitations which suggest natural directions for future research. A first limitation concerns the scope of the application domains considered in the dissertation. Although the proposed methods and benchmarks were motivated from a general NAS perspective, the experimental investigation focused mainly on image classification and closely related visual settings. In particular, the training-free methods were studied on classification-oriented search spaces and benchmarks. Extending these ideas to other tasks, such as object detection, segmentation, visual understanding or multi-modal architectures, remains an open direction. This is especially relevant because the behavior of performance proxies, search strategies and evaluation criteria may differ substantially across tasks.

A second limitation regards the restricted nature of the search spaces considered in the thesis. In the UDA setting, the search was intentionally limited to an auxiliary branch rather than the full network. In the training-free setting, the studied spaces, although progressively more realistic, still remain controlled subsets of possible architectural designs. This restriction is methodologically reasonable, since fully unconstrained search would be considerably harder to analyze and evaluate. However, it also means that the conclusions drawn here should not be interpreted as universally valid for arbitrary architecture spaces. A relevant future direction is therefore to investigate how the proposed ideas scale to broader and less structured search spaces, where interaction between topology, dimensionality and efficiency may be more complex.

A third limitation concerns the approximate nature of the evaluation signals used throughout the thesis. In different ways, the proposed methods rely on accessible proxies rather than on the ideal downstream objective. In the UDA setting, model selection is guided by surrogate criteria because target labels are unavailable. In particular, EMS does not use target labels from the adaptation task under evaluation, but it still requires labeled auxiliary multi-domain data to learn the cross-dataset performance estimator. This makes the approach suitable for settings where related labeled adaptation benchmarks are available, but also raises questions about its transferability to less related domains or to scenarios where such auxiliary supervision is missing. In training-free NAS, zero-cost metrics act as proxies for final trained performance. In benchmarking, even a large-scale dataset and a diverse set of evaluation criteria still provide only a partial approximation of real deployment conditions. These approximations are necessary to make the problem tractable, but they also define the limits of the proposed approaches. Future work should therefore continue investigating how to construct evaluation signals that are both efficient and more tightly aligned with the final quantities of interest.

In the specific case of training-free NAS, one promising direction is to move toward more specialized and adaptive proxy designs. The results of the works presented in this thesis suggest that it is unlikely that a single universal zero-cost metric will perform equally well across all architectural domains and evaluation settings. A more promising strategy is to develop families of complementary proxies, each tailored to a specific design dimension, architecture class, or target property. This could be extended further through adaptive search procedures that estimate when a proxy is reliable, or through hybrid methods that selectively incorporate lightweight training signals only in regions of the search space where training-free estimates become uncertain.

Another important direction concerns the integration of multiple practical constraints within a unified NAS framework. In this thesis, missing supervision, computational efficiency, and realistic evaluation were addressed in separate but related contributions. In many realistic applications, however, these aspects coexist. A future practical NAS framework should ideally be able to reason jointly about them: for example, deciding how much of the architecture should be searched, which evaluation signals are trustworthy under limited supervision, and how to trade off predictive performance, efficiency and robustness. Developing methods that address these dimensions simultaneously remains a challenging but highly relevant research direction.

The benchmarking contribution also opens several opportunities for future work. One of them is the development of broader and more realistic NAS benchmarks, including richer search spaces, additional architecture classes, more diverse deployment targets, and more detailed hardware-related measurements. Another is the need to better understand the relationship between in-distribution performance and more deployment-relevant criteria, such as robustness, calibration, latency or energy efficiency. The analysis in this thesis already suggests that strong in-distribution performance is not sufficient to characterize architectural quality. Future benchmarks and NAS methods should therefore increasingly reflect this multidimensional perspective.

Finally, the thesis points toward a broader research direction in which NAS is treated as a problem of decision-making under imperfect information and practical constraints. From this viewpoint, architecture search is meaningful only to the extent that the optimized objective, the accessible evaluation signal, and the target deployment scenario remain sufficiently aligned. Future work could build more explicitly on this perspective, developing NAS methods that reason about uncertainty in their own evaluation signals, adapt their search behavior to the reliability of the available feedback, and explicitly account for mismatches between benchmark objectives and real-world requirements.

Bibliography

- [1] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, “Zero-cost proxies for lightweight NAS”, in *International Conference on Learning Representations*, 2021.
- [2] A. Ali et al., “Xcit: Cross-covariance image transformers”, *Advances in neural information processing systems*, 2021.
- [3] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization”, *arXiv preprint arXiv:1607.06450*, 2016.
- [4] E. Barbierato and A. Gatti, “Toward green ai: A methodological survey of the scientific literature”, *Ieee Access*, vol. 12, pp. 23 989–24 013, 2024.
- [5] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, “Hardware-aware neural architecture search: Survey and taxonomy.”, in *IJCAI*, vol. 2021, 2021, pp. 4322–4329.
- [6] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, “Path-level network transformation for efficient architecture search”, in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 678–687.
- [7] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware”, in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://arxiv.org/pdf/1812.00332.pdf>
- [8] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis”, *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.
- [9] F. M. Carlucci, A. D’Innocente, S. Bucci, B. Caputo, and T. Tommasi, “Domain generalization by solving jigsaw puzzles”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2229–2238.

- [10] F. M. Carlucci, P. Russo, T. Tommasi, and B. Caputo, “Hallucinating agnostic images to generalize across domains”, in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, 2019, pp. 3227–3234.
- [11] N. Cavagnero, L. Robbiano, B. Caputo, and G. Averta, “FreeREA: Training-free evolution-based architecture search”, in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 1493–1502. DOI: [10.1109/WACV56688.2023.00154](https://doi.org/10.1109/WACV56688.2023.00154)
- [12] N. Cavagnero, L. Robbiano, F. Pistilli, B. Caputo, and G. Averta, “Entropic Score metric: Decoupling topology and size in training-free nas”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2023, pp. 1459–1468.
- [13] B. Chen et al., “Glit: Neural architecture search for global and local image transformer”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12–21.
- [14] J. Chen and X. Ran, “Deep learning with edge computing: A review”, *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019. DOI: [10.1109/JPROC.2019.2921977](https://doi.org/10.1109/JPROC.2019.2921977)
- [15] M. Chen, S. Zhao, H. Liu, and D. Cai, “Adversarial-learned loss for domain adaptation”, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 3521–3528.
- [16] W. Chen, X. Gong, and Z. Wang, “Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective”, in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=Cnon5ezMHtu>
- [17] Y. Chen et al., “Mobile-former: Bridging mobilenet and transformer”, in *CVPR*, 2022.
- [18] Z. Chen, F. Zhong, Q. Luo, X. Zhang, and Y. Zheng, “Edgevit: Efficient visual modeling for edge computing”, in *International conference on wireless algorithms, systems, and applications*, Springer, 2022, pp. 393–405.
- [19] P. Chrabaszcz, I. Loshchilov, and F. Hutter, “A downsampled variant of imagenet as an alternative to the CIFAR datasets”, *arXiv preprint arXiv:1707.08819*, 2017.
- [20] M. Cordts et al., “The cityscapes dataset for semantic urban scene understanding”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

- [21] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “Randaugment: Practical automated data augmentation with a reduced search space”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.
- [22] P. Dhar, “The carbon impact of artificial intelligence”, *Nature Machine Intelligence*, vol. 2, no. 8, pp. 423–425, 2020.
- [23] M. Ding et al., “Hr-nas: Searching efficient high-resolution neural architectures with lightweight transformers”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2982–2992.
- [24] M. Ding et al., “Learning versatile neural architectures by propagating network codes”, in *International Conference on Learning Representations*, 2022.
- [25] T. Do and N. H. Luong, “Training-free multi-objective evolutionary neural architecture search via neural tangent kernel and number of linear regions”, in *International Conference on Neural Information Processing*, Springer, 2021, pp. 335–347.
- [26] X. Dong, L. Liu, K. Musial, and B. Gabrys, “NATS-Bench: Benchmarking nas algorithms for architecture topology and size”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. DOI: [10.1109/TPAMI.2021.3054824](https://doi.org/10.1109/TPAMI.2021.3054824)
- [27] X. Dong and Y. Yang, “NAS-Bench-201: Extending the scope of reproducible neural architecture search”, in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: <https://openreview.net/forum?id=HJxyZkBKDr>
- [28] X. Dong and Y. Yang, “Searching for a robust neural architecture in four gpu hours”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1761–1770.
- [29] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale”, in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=YicbFdNTTy>
- [30] L. Dudziak, T. Chau, M. Abdelfattah, R. Lee, H. Kim, and N. Lane, “Brp-nas: Prediction-based nas using gcns”, *Advances in neural information processing systems*, vol. 33, pp. 10 480–10 490, 2020.
- [31] S. Falkner, A. Klein, and F. Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale”, in *International conference on machine learning*, PMLR, 2018, pp. 1437–1446.
- [32] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity”, *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.

- [33] Y. Ganin et al., “Domain-adversarial training of neural networks”, *Journal of machine learning research*, vol. 17, no. 59, pp. 1–35, 2016.
- [34] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite”, in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [35] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, “Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness”, in *International conference on learning representations*, 2018.
- [36] C. Gong et al., “Nasvit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training”, in *International Conference on Learning Representations*, 2021.
- [37] B. Graham et al., “Levit: A vision transformer in convnet’s clothing for faster inference”, in *ICCV*, 2021.
- [38] Y. Grandvalet and Y. Bengio, “Semi-supervised learning by entropy minimization”, *Advances in neural information processing systems*, vol. 17, 2004.
- [39] I. Gulrajani and D. Lopez-Paz, “In search of lost domain generalization”, *arXiv preprint arXiv:2007.01434*, 2020.
- [40] Z. Guo et al., “Single path one-shot neural architecture search with uniform sampling”, in *European conference on computer vision*, Springer, 2020, pp. 544–560.
- [41] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.
- [42] C. He, H. Tan, S. Huang, and R. Cheng, “Efficient evolutionary neural architecture search by modular inheritable crossover”, *Swarm and Evolutionary Computation*, vol. 64, p. 100 894, 2021, ISSN: 2210-6502. DOI: <https://doi.org/10.1016/j.swevo.2021.100894> [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2210650221000559>
- [43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [44] X. He, K. Zhao, and X. Chu, “Automl: A survey of the state-of-the-art”, *Knowledge-based systems*, vol. 212, p. 106 622, 2021.
- [45] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations”, in *7th International Conference on Learning Representations, ICLR*, 2019.

- [46] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus)”, *arXiv preprint arXiv:1606.08415*, 2016.
- [47] S.-T. Ho, T. Van Vo, S. Ebrahimekhani, and N.-m. Cheung, “Vision transformer neural architecture search for out-of-distribution generalization: Benchmark and insights”, in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- [48] W. Hong, Z. Wang, M. Yang, and J. Yuan, “Conditional generative adversarial network for structured domain adaptation”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1335–1344.
- [49] R. Hosseini, X. Yang, and P. Xie, “Dsrna: Differentiable search of robust neural architectures”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6196–6205.
- [50] Q. Hu et al., “Characterization of large language model development in the datacenter”, in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 709–729.
- [51] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth”, in *European conference on computer vision*, Springer, 2016, pp. 646–661.
- [52] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization”, in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1501–1510.
- [53] A. Jacot, F. Gabriel, and C. Hongler, “Neural tangent kernel: Convergence and generalization in neural networks”, in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf
- [54] Y. Jiang*, B. Neyshabur*, H. Mobahi, D. Krishnan, and S. Bengio, “Fantastic generalization measures and where to find them”, in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=SJgIPJBFvH>
- [55] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, “Neural architecture search with bayesian optimisation and optimal transport”, *Advances in neural information processing systems*, vol. 31, 2018.
- [56] A. Krizhevsky, G. Hinton, et al., “Learning multiple layers of features from tiny images”, 2009.
- [57] J. Lee, H. Davari, J. Singh, and V. Pandhare, “Industrial artificial intelligence for industry 4.0-based manufacturing systems”, *Manufacturing letters*, vol. 18, pp. 20–23, 2018.

- [58] N. Lee, T. Ajanthan, and P. Torr, “SNIP: Single-shot network pruning based on connection sensitivity”, in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=B1VZqjAcYX>
- [59] C. Li et al., “Hw-nas-bench: Hardware-aware neural architecture search benchmark”, *arXiv preprint arXiv:2103.10584*, 2021.
- [60] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, “Deeper, broader and artier domain generalization”, in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5542–5550.
- [61] G. Li, D. Hoang, K. Bhardwaj, M. Lin, Z. Wang, and R. Marculescu, “Zero-shot neural architecture search: Challenges, solutions, and opportunities”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 7618–7635, 2024. DOI: [10.1109/TPAMI.2024.3395423](https://doi.org/10.1109/TPAMI.2024.3395423)
- [62] J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and Z. Huang, “Cycle-consistent conditional adversarial transfer networks”, in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 747–755.
- [63] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search”, in *Uncertainty in artificial intelligence*, PMLR, 2020, pp. 367–377.
- [64] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: Bandit-based configuration evaluation for hyperparameter optimization”, in *International Conference on Learning Representations*, 2017.
- [65] P. Li, J. Yang, M. A. Islam, and S. Ren, “Making ai less ‘thirsty’”, *Commun. ACM*, vol. 68, no. 7, pp. 54–61, Jun. 2025, ISSN: 0001-0782. DOI: [10.1145/3724499](https://doi.org/10.1145/3724499) [Online]. Available: <https://doi.org/10.1145/3724499>
- [66] Y. Li et al., “Rethinking vision transformers for mobilenet size and speed”, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 16 889–16 900.
- [67] Y. Li et al., “Localvit: Analyzing locality in vision transformers”, in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 9598–9605. DOI: [10.1109/IROS55552.2023.10342025](https://doi.org/10.1109/IROS55552.2023.10342025)
- [68] J. Liang, R. He, Z. Sun, and T. Tan, “Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2975–2984.
- [69] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search”, in *International Conference on Learning Representations*, 2019.

- [70] H.-I. Liu et al., “Lightweight deep learning for resource-constrained environments: A survey”, *ACM Computing Surveys*, vol. 56, no. 10, pp. 1–42, 2024.
- [71] S. Liu, H. Zhang, and Y. Jin, “A survey on computationally efficient neural architecture search”, *Journal of Automation and Intelligence*, vol. 1, no. 1, p. 100 002, 2022, ISSN: 2949-8554. DOI: <https://doi.org/10.1016/j.jai.2022.100002>
- [72] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, “A survey on evolutionary neural architecture search”, *IEEE transactions on neural networks and learning systems*, vol. 34, no. 2, pp. 550–570, 2021.
- [73] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A convnet for the 2020s”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 976–11 986.
- [74] M. R. Loghmani, L. Robbiano, M. Planamente, K. Park, B. Caputo, and M. Vincze, “Unsupervised domain adaptation through inter-modal rotation for rgb-d object recognition”, *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6631–6638, 2020. DOI: [10.1109/LRA.2020.3007092](https://doi.org/10.1109/LRA.2020.3007092)
- [75] M. Long, Z. Cao, J. Wang, and M. I. Jordan, “Conditional adversarial domain adaptation”, in *Advances in Neural Information Processing Systems*, 2018, pp. 1640–1650.
- [76] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Deep transfer learning with joint adaptation networks”, in *International conference on machine learning*, PMLR, 2017, pp. 2208–2217.
- [77] V. Lopes, S. Alirezazadeh, and L. A. Alexandre, “Epe-nas: Efficient performance estimation without training for neural architecture search”, in *International conference on artificial neural networks*, Springer, 2021, pp. 552–563.
- [78] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization”, in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=Bkg6RiCqY7>
- [79] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts”, in *International Conference on Learning Representations*, 2017. [Online]. Available: <https://openreview.net/forum?id=Skq89Scxx>
- [80] A. S. Luccioni, S. Viguier, and A.-L. Ligozat, “Estimating the carbon footprint of bloom, a 176b parameter language model”, *Journal of Machine Learning Research*, vol. 24, no. 253, pp. 1–15, 2023.
- [81] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization”, *Advances in neural information processing systems*, vol. 31, 2018.

- [82] M. Maaz et al., “EdgeNext: Efficiently amalgamated cnn-transformer architecture for mobile vision applications”, in *European conference on computer vision*, Springer, 2022, pp. 3–20.
- [83] Y. Mao, X. Yu, K. Huang, Y.-J. Angela Zhang, and J. Zhang, “Green edge ai: A contemporary survey”, *Proceedings of the IEEE*, vol. 112, no. 7, pp. 880–911, 2024. DOI: [10.1109/JPROC.2024.3437365](https://doi.org/10.1109/JPROC.2024.3437365)
- [84] S. Mehta and M. Rastegari, “Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer”, in *International Conference on Learning Representations*, 2022. [Online]. Available: <https://openreview.net/forum?id=vh-0sUt8HlG>
- [85] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, “Neural architecture search without training”, in *International conference on machine learning*, PMLR, 2021, pp. 7588–7598.
- [86] J. Morrison, C. Na, J. Fernandez, T. Dettmers, E. Strubell, and J. Dodge, “Holistically evaluating the environmental impact of creating language models”, in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=04qx93Viwj>
- [87] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines”, in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [88] M. Nomura and Y. Saito, “Efficient hyperparameter optimization under multi-source covariate shift”, in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1376–1385.
- [89] OECD, “Competition in artificial intelligence infrastructure”, Organisation for Economic Co-operation and Development, Tech. Rep., 2025. DOI: [10.1787/623d1874-en](https://doi.org/10.1787/623d1874-en)
- [90] R. Panda et al., “Nastransfer: Analyzing architecture transferability in large scale neural architecture search”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 9294–9302.
- [91] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks”, in *International conference on machine learning*, Pmlr, 2013, pp. 1310–1318.
- [92] P. Patel et al., “Characterizing power management opportunities for llms in the cloud”, in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2024, pp. 207–222.
- [93] D. Patterson et al., “Carbon emissions and large neural network training”, *arXiv preprint arXiv:2104.10350*, 2021.

- [94] F. Pedregosa et al., “Scikit-learn: Machine learning in Python”, *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [95] Z. Pei, Z. Cao, M. Long, and J. Wang, “Multi-adversarial domain adaptation”, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S. A. McIlraith and K. Q. Weinberger, Eds., AAAI Press, 2018, pp. 3934–3941. [Online]. Available: <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17067>
- [96] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing”, in *International conference on machine learning*, PMLR, 2018, pp. 4095–4104.
- [97] S. A. Popenici and S. Kerr, “Exploring the impact of artificial intelligence on teaching and learning in higher education”, *Research and Practice in Technology Enhanced Learning*, vol. 12, no. 1, pp. 1–13, 2017.
- [98] A. Rajkomar, J. Dean, and I. Kohane, “Machine learning in medicine”, *New England Journal of Medicine*, vol. 380, no. 14, pp. 1347–1358, 2019.
- [99] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions”, *arXiv preprint arXiv:1710.05941*, 2017.
- [100] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search”, in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.
- [101] E. Real et al., “Large-scale evolution of image classifiers”, in *International conference on machine learning*, PMLR, 2017, pp. 2902–2911.
- [102] L. Robbiano, F. Pistilli, and G. Averta, “A CNN-ViT hybrid architecture search benchmark on a large-scale dataset”, *IEEE Access*, vol. 13, pp. 209 965–209 979, 2025. DOI: [10.1109/ACCESS.2025.3642734](https://doi.org/10.1109/ACCESS.2025.3642734)
- [103] L. Robbiano, M. R. Ur Rahman, F. Galasso, B. Caputo, and F. M. Carlucci, “Adversarial branch architecture search for unsupervised domain adaptation”, in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 1008–1018. DOI: [10.1109/WACV51458.2022.00108](https://doi.org/10.1109/WACV51458.2022.00108)
- [104] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”, *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

- [105] S. Roy, A. Siarohin, E. Sangineto, S. R. Buló, N. Sebe, and E. Ricci, “Un-supervised domain adaptation using feature-whitening and consensus loss”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9471–9480.
- [106] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual category models to new domains”, in *European conference on computer vision*, Springer, 2010, pp. 213–226.
- [107] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.
- [108] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green ai”, *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.
- [109] N. Shazeer, Z. Lan, Y. Cheng, N. Ding, and L. Hou, “Talking-heads attention”, *arXiv preprint arXiv:2003.02436*, 2020.
- [110] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges”, *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [111] Y. Shu, S. Cai, Z. Dai, B. C. Ooi, and B. K. H. Low, “NASI: Label- and data-agnostic neural architecture search at initialization”, in *International Conference on Learning Representations*, 2022.
- [112] C. Si, W. Yu, P. Zhou, Y. Zhou, X. Wang, and S. Yan, “Inception transformer”, *Advances in neural information processing systems*, vol. 35, pp. 23 495–23 509, 2022.
- [113] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms”, *Advances in neural information processing systems*, vol. 25, 2012.
- [114] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for deep learning in nlp”, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3645–3650.
- [115] X. Su et al., “Vitas: Vision transformer architecture search”, in *ECCV*, 2022.
- [116] M. Suganuma, M. Kobayashi, S. Shirakawa, and T. Nagao, “Evolution of deep convolutional neural networks using cartesian genetic programming”, *Evolutionary Computation*, vol. 28, no. 1, pp. 141–163, 2020.
- [117] M. Sugiyama, M. Krauledat, and K.-R. Müller, “Covariate shift adaptation by importance weighted cross validation.”, *Journal of Machine Learning Research*, vol. 8, no. 5, 2007.

- [118] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Evolving deep convolutional neural networks for image classification”, *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2020.
- [119] Y. Sun, E. Tzeng, T. Darrell, and A. A. Efros, “Unsupervised domain adaptation through self-supervision”, *arXiv preprint arXiv:1909.11825*, 2019.
- [120] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [121] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks”, in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.
- [122] M. Tan et al., “Mnasnet: Platform-aware neural architecture search for mobile”, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2820–2828.
- [123] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, “Pruning neural networks without any data by iteratively conserving synaptic flow”, *Advances in neural information processing systems*, vol. 33, pp. 6377–6389, 2020.
- [124] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention”, in *ICML*, 2021.
- [125] J. Turner, E. J. Crowley, M. O’Boyle, A. Storkey, and G. Gray, “Blockswap: Fisher-guided block substitution for network compression on a budget”, in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=Sk1kDkSFPB>
- [126] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, “Simultaneous deep transfer across domains and tasks”, in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4068–4076.
- [127] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.
- [128] A. Vaswani et al., “Attention is all you need”, *Advances in neural information processing systems*, vol. 30, 2017.
- [129] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, “Deep hashing network for unsupervised domain adaptation”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5018–5027.

- [130] Z. Wan et al., “Efficient large language models: A survey”, *Transactions on Machine Learning Research*, 2024, Survey Certification, ISSN: 2835-8856. [Online]. Available: <https://openreview.net/forum?id=bsCCJHb08A>
- [131] C. Wang, G. Zhang, and R. Grosse, “Picking winning tickets before training by preserving gradient flow”, in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=SkgsACVKPH>
- [132] H. Wang, S. Ge, Z. Lipton, and E. P. Xing, “Learning robust global representations by penalizing local predictive power”, *Advances in neural information processing systems*, vol. 32, pp. 10 506–10 518, 2019.
- [133] C. White, W. Neiswanger, and Y. Savani, “Bananas: Bayesian optimization with neural architectures for neural architecture search”, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 10 293–10 301.
- [134] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning”, *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.
- [135] M.-T. Wu, H.-I. Lin, and C.-W. Tsai, “A training-free genetic neural architecture search”, in *Proceedings of the 2021 ACM International Conference on Intelligent Computing and Its Emerging Applications*, 2021, pp. 65–70.
- [136] X. Wu, Q. Zhou, Z. Yang, C. Zhao, L. J. Latecki, et al., “Entropy minimization vs. diversity maximization for domain adaptation”, *arXiv preprint arXiv:2002.01690*, 2020.
- [137] Y. Wu, D. Inkpen, and A. El-Roby, “Dual mixup regularized learning for adversarial domain adaptation”, in *European Conference on Computer Vision*, Springer, 2020, pp. 540–555.
- [138] L. Xie and A. Yuille, “Genetic cnn”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1379–1388.
- [139] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.
- [140] M. Xu et al., “Adversarial domain adaptation with domain mixup”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 6502–6509.
- [141] A. Yang, P. M. Esperança, and F. M. Carlucci, “Nas evaluation is frustratingly hard”, in *International Conference on Learning Representations*, 2020.

- [142] L. Yang et al., “Searching for burgerformer with micro-meso-macro space design”, in *International conference on machine learning*, PMLR, 2022, pp. 25 055–25 069.
- [143] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, “NAS-bench-101: Towards reproducible neural architecture search”, in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, Long Beach, California, USA: PMLR, Sep. 2019, pp. 7105–7114. [Online]. Available: <http://proceedings.mlr.press/v97/ying19a.html>
- [144] K. You, X. Wang, M. Long, and M. Jordan, “Towards accurate model selection in deep unsupervised domain adaptation”, in *International Conference on Machine Learning*, PMLR, 2019, pp. 7124–7133.
- [145] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features”, in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.
- [146] D. Zha et al., “Data-centric artificial intelligence: A survey”, *ACM Computing Surveys*, vol. 57, no. 5, pp. 1–42, 2025.
- [147] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization”, in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=r1Ddp1-Rb>
- [148] W. Zhang, W. Ouyang, W. Li, and D. Xu, “Collaborative and adversarial network for unsupervised domain adaptation”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3801–3809.
- [149] S. Zhao et al., “A review of single-source deep unsupervised visual domain adaptation”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 473–493, 2020.
- [150] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation”, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 13 001–13 008.
- [151] Q. Zhou et al., “Training-free transformer architecture search”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 894–10 903.
- [152] T. Zhou, S. Wang, and J. Bilmes, “Time-consistent self-supervision for semi-supervised learning”, in *International conference on machine learning*, PMLR, 2020, pp. 11 523–11 533.

- [153] B. Zoph and Q. Le, “Neural architecture search with reinforcement learning”, in *International Conference on Learning Representations*, 2017.
- [154] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.
- [155] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang, “Unsupervised domain adaptation for semantic segmentation via class-balanced self-training”, in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 289–305.