NAS Without Priors: A Robust Architecture Search Framework for Unseen-Data

Anonymous authors
Paper under double-blind review

Abstract

Neural architecture search (NAS) has been widely used to automate neural network design for image classification; however, most NAS research has focused on CIFAR, ImageNet, and their derivative benchmarks. These datasets benefit from well-established architecture design practices, data preprocessing techniques, and training protocols developed prior to NAS, causing many NAS methods to meta-overfit and struggle to generalize to entirely novel datasets. In this work, we analyze the limitations of existing NAS practices and propose a framework specifically designed to generalize to unseen-data. In contrast to the prevailing paradigm of exploring extremely large search spaces using low-fidelity evaluations, we advocate sparser exploration combined with high-fidelity performance estimation. We demonstrate that macro-architecture variations alone induce substantial architectural diversity, and that concentrating computational resources on high-fidelity evaluation of fewer candidates produces reliable reward signals enabling better architecture discovery. To obtain robust candidate rankings, we repeatedly train architectures on the entire training set using multiple random seeds. While this approach substantially reduces performance variance due to random seed variability and enables accurate candidate ranking, it comes at a significant computational cost. To mitigate the cost of such high-fidelity evaluation, particularly for larger or high-resolution datasets, we introduce a dataset- and architecture-aware multi-fidelity search strategy that both reduces computational overhead and stabilizes candidate rankings under varying fidelity levels. We evaluate our framework on the NAS Unseen-Data Challenge[1], where, under a strict time constraint of 8 hours per dataset for both search and training, it outperforms manually designed architectures across all three challenge datasets and achieves first place with a combined score of 12.19, compared to 10.89 and 10.43 for the second- and third-place NAS solutions, respectively. Moreover, we achieve competitive error rates against several macro- as well as micro-NAS (DARTS-based) search methods on CIFAR-10.

1 Introduction

Neural architecture search (NAS) aims to automatically discover an optimal neural network architecture for a given dataset. Existing NAS works (Zoph & Le, 2016; Baker et al., 2016; Real et al., 2017; Suganuma et al., 2017; Zoph et al., 2018; Liu et al., 2018) have produced competitive results on CIFAR (Krizhevsky et al., 2009) and ImageNet (Russakovsky et al., 2015) datasets; however, prior advances in data augmentation and training tricks have been shown to contribute more substantially to accuracy gains than the NAS methods themselves (Geada et al., 2024; Yang et al., 2019). Moreover, even NAS-specific benchmarks (Ying et al., 2019; Dong & Yang, 2020) are derived from these same datasets, and continued reliance on them has limited the applicability of NAS to truly novel tasks (Mehta et al., 2022). For instance, the widely adopted cell-only search paradigm (Zoph et al., 2018; Liu et al., 2018) does not generalize well to new datasets, as macro-architecture must still be manually specified. While effective macro-level configurations for CIFAR and ImageNet were established through prior expert knowledge, the optimal settings for a new dataset are generally unknown. Further, despite encompassing an enormous number of candidate architectures (e.g.,

[1]NAS Unseen-Data Challenge 2025 Leaderboard

10^{35} and 10^{18} in Zoph et al. (2018) and Liu et al. (2018), respectively), such search spaces are known to exhibit limited performance variance (Yang et al., 2019). In contrast, early NAS works (Zoph & Le, 2016; Baker et al., 2016; Real et al., 2017; Cai et al., 2018) enable exploration of the entire architecture, but navigating these large spaces using inefficient evaluation schemes renders them computationally impractical (e.g., Zoph & Le (2016) trained 12,800 architectures for 50 epochs each, requiring 22400 GPU days).

The extreme computational cost of NAS, driven by enormous search spaces, has motivated the development of numerous techniques for accelerating candidate evaluation. One simple strategy is learning-curve extrapolation, which evaluates architectures using only a few training epochs (Domhan et al., 2015; Yan et al., 2021; Baker et al., 2017); however, newer datasets can exhibit substantially different convergence behaviors, making it difficult to predefine a fixed early stopping criterion that generalizes reliably. Another widely used low-fidelity approximation trains candidates on a subset of the data (Klein et al., 2017; Prasad et al., 2024; Shim et al., 2021; Na et al., 2021), but identifying a representative subset, one that yields performance comparable to full-dataset training, implicitly requires knowledge of the highest achievable accuracy, which is unknown *a priori* for unseen datasets. Similarly, model-based performance predictors (Wen et al., 2020; White et al., 2021b) rely on architecture–accuracy pairs that must be collected anew for each dataset, limiting their practicality for newer datasets. More aggressive speedup strategies attempt to estimate architecture performance with minimal or no training (Krishnakumar et al., 2022); however, decoupling evaluation from the target dataset undermines the core objective of NAS, that is, discovering dataset-specific architectures. Other approaches transfer weights between architecturally similar candidates (Real et al., 2017; Chen et al., 2016), introducing bias by favoring candidates evaluated later in the search, while supernet based methods (Saxena & Verbeek, 2016; Liu et al., 2018) rely on weight sharing that has been shown to produce inaccurate candidate rankings (Yang et al., 2019; Yu et al., 2019). Overall, existing evaluation schemes largely prioritize accelerating the assessment of large candidate sets, at the expense of accurately estimating the relative performance of competing architectures and many of these do not generalize well to newer datasets. Table 1 draws a comparison of different NAS components for standard datasets such as CIFAR and Imagenet versus a few newer datasets presented by (Geada et al., 2024) for the NAS Unseen-data challenge 2025 (Towers et al., 2025a). No prior information about the nature, dimensions, sample size and convergence rate of the unseen data makes the transferability of existing NAS frameworks non-trivial. Moreover, the variability across different unseen data challenges their generalization capabilities.

In this work, we revisit existing NAS search spaces and evaluation strategies, arguing that allocating computational resources to obtain reliable rankings over a smaller set of candidates is more effective than conducting a large number of noisy evaluations, particularly when large candidate sets exhibit limited performance variance. Accordingly, we first analyze why existing search spaces do not generalize well and propose a substantially smaller yet dataset-adaptive design space with higher performance variability. We then examine candidate evaluation and address a well-known but largely overlooked issue of high performance variability across different random seeds (Li & Talwalkar, 2020; Yang et al., 2019; Lindauer & Hutter, 2020). Prior work has mainly treated this variability as a reproducibility concern, leading to recommendations for reporting averaged results across seeds. In contrast, we emphasize its impact during the search process itself, where reliance on a single seed can directly bias architecture selection. Ideally, to some extent, architecture discovery should be dataset-responsive rather than seed-sensitive. To achieve reliable candidate ranking, we incorporate multi-seed evaluation directly into the search process, evaluating each candidate across multiple seeds and ranking architectures based on their best observed performance. Furthermore, unlike existing approaches, we leverage the entire training set during search, hence, reducing the data-gap between search and final training, and ensuring correct suitability of the discovered architecture for the given dataset. While this high-fidelity evaluation improves robustness, it remains computationally expensive even for a small number of candidates; therefore, we introduce a multi-fidelity search algorithm that reduces overall search cost while further ensuring better architecture selection among competing architectures.

Multi-fidelity search algorithms, such as Successive Halving (SH) (Jamieson & Talwalkar, 2016), Hyperband (HB) (Li et al., 2018), and Bayesian Optimization with Hyperband (BOHB) (Falkner et al., 2018), progressively increase the evaluation budget for promising candidates, unlike majority of fixed-budget NAS works (Zoph & Le, 2016; Real et al., 2017; Liu et al., 2018; Domhan et al., 2015; Klein et al., 2017). Although originally designed to speed up the search, some studies have shown that jointly searching for architecture

Table 1: Comparison of NAS search spaces, search strategies and evaluation schemes for standard seen versus unseen data.

Dataset	Name-Type Dims-Size	Search Spaces	Evaluation Strategies	Search Strategies
Seen	CIFAR-Objects 3x32x32-10 50K/10K ImageNet-Objects 3x224x224-1K 12M/100K	1. Insights into data characteristics e.g. 'color images with spatially connected shape features' influence search space design from state-of-the-art hand-crafted architectures for CIFAR10 and ImageNet. 2. Known data dimensions and effective parameters/FLOP budget allows fixing network outer skeleton such as stem, reduction blocks and terminal layers. This also allows constructing supernet search spaces.	1. Known number of train-val samples, dimensions, evaluation cost expected validation performance allows low-fidelity schemes e.g. using 50% data with 25-50 epochs for CIFAR10. 2. Established augmentations such as random crop and horizontal flip and training settings such as epochs, batch size, learning rate and scheduling, momentum, weight decay, and cutout.	1. Differentialble NAS suffers from stability issues e.g. inaccurate relative rankings. 2. Blackbox strategies such as RL, and evolution work well assuming all candidates comparable under a single evaluation protocol. 3. Joint architecture and training hyper-parameter search (JAHS) is desirable however the search becomes expensive.
Unseen	Conway-Synthetic 1x10x10-25 50K/10K Cryptic-Language 1x6x768-2 59K/17K Windspeed-Weather 12x32x32-20 20K/7K	1. The underlying nature of patterns is unknown and varies e.g. spatial clues are relevant for Conway but not for Windspeed. Prior state-of-the-art baseline networks are unavailable. 2. Non-trivial data dimensions and output classes demand flexibility in outer skeleton. Required params/FLOPS for a given dataset are also unknown. Since no prior information about macro-architecture, a generalizable search super-network is unavailable. Hence macro search spaces are more suitable.	1. Unknown dimensions, and train-val sizes, incur substantially varying eval costs across datasets. Unknown convergence behaviors hinder specifying a generalized termination criteria. Random data subsets may not be representative if fewer or unbalanced samples per class. 2. CIFAR/ImageNet augmentations do not generalize to unseen data and unseen data may Over-/under- fit under existing protocols.	1. One-shot NAS such as OFA is possible but initial supernet training is expensive for larger dimension datasets and the subnetworks may still require evaluation to find the optimal one. 2. Varying datasets with unknown training protocols require adaptive eval strategies. 3. Multi-fidelity search is applicable allowing evaluation at various training fidelities. This is similar to JAHS but implicit and partial hyperparameter search.

and training protocol yields better results (Yang et al., 2019; Bansal et al., 2022; Zela et al., 2018). For unseen data, where training protocols are also unknown, multi-fidelity evaluation can serve dual purpose; speeding up evaluation as well as allocating multiple training protocols to candidates for more reliable assessments. However, existing multi-fidelity methods typically allocate increased budgets only to high-performing candidates, potentially disadvantaging models that may require longer training to reach their full potential. Moreover, they either rely on unguided random search (Jamieson & Talwalkar, 2016; Li et al., 2018) or build surrogate performance models (Falkner et al., 2018), which can become costly for larger datasets. To address these limitations, we propose a novel search strategy that allocates dataset- and model-adaptive fidelity budgets. The strategy consists of two stages. In Phase 1, we collect a pool of promising candidates for a given dataset using local search, which has been shown to be a fast and competitive baseline for NAS (Den Ottelander et al., 2021; White et al., 2021a). These candidates are then further evaluated in Phase 2 using a model-aware fidelity allocation scheme, where the best architecture is identified through iterative elimination. The proposed strategy enables the discovery of dataset-specific architectures by ensuring fair comparison across models of varying sizes while remaining computationally feasible.

Our contributions can be summarized as follows:

- We revisit common NAS practices that rely on low-fidelity performance estimation over large search spaces and examine their limitations when applied to unseen datasets, motivating the use of higher-fidelity evaluation with a smaller set of candidates.
- We demonstrate that competitive, dataset-adaptive architectures can be obtained from a compact search space defined by two macro-architectural variables, namely network depth and width.
- We propose a candidate evaluation strategy that uses the full training dataset and multiple random seeds to obtain stable performance estimates during the search process.
- To manage the computational cost associated with high-fidelity evaluation, we introduce a dataset- and architecture-aware multi-fidelity search algorithm that balances evaluation accuracy with efficiency.

To evaluate our framework, we follow recent work (Geada et al., 2024) that highlights the importance of continuously evolving NAS benchmarks for assessing generalization to novel datasets. This motivation led to the annual NAS Unseen-Data Challenge (Towers et al., 2025a), where participants are required to design

a complete NAS pipeline and apply it to three entirely new and previously unseen datasets. Our proposed approach outperforms manually designed baseline architectures across all unseen datasets and achieves the highest combined score of 12.19, exceeding the second- and third-place solutions, which obtained scores of 10.89 and 10.43, respectively. However, since the majority of existing macro-NAS work is evaluated on standard benchmarks, we also conduct comprehensive experiments on CIFAR-10. Further, to enable fair comparison with existing search spaces, and to disentangle the contribution of proposed search strategy, we also evaluate the transferability of our proposed search strategy on the widely used DARTS (micro) search space, achieving a competitive error rate relative to recent methods. Our macro-search results outperform several macro- and micro-search approaches, highlighting the effectiveness of macro-level search over micro-level design.

2 Related Work

The **search space** defines the set of neural architectures that the NAS method is allowed to explore, while the **search strategy** determines how candidate architectures are selected and refined within this space. To identify architectures that maximize performance on a given dataset, **candidate evaluation** scheme estimates their performance and provides a reward signal to search strategy. In the following, we review related work corresponding to each of these components.

2.1 Search Space

We categorize existing search spaces into three types: **macro-micro**, **cell-based** and **macro-only** and observe pros and cons associated with each. For a similar, yet detailed categorization of search spaces, we refer the reader to White et al. (2023). From here on, a search space may simply be referred to as space.

We refer to macro-micro spaces as those which allow searching for networks' macro (e.g. depth, width, skip connections) as well as micro/layer-level architecture (e.g. kernel size, layer type, strides). We list some of the prominent relevant spaces in Table 2. Some of these works are inspired by ResNet architecture (He et al., 2016) and incorporate skip connections resulting in multi-branch candidates (Zoph & Le, 2016; Real et al., 2017; Elsken et al., 2017; Kandasamy et al., 2018), while the rest follow chain-structured architecture paradigm where a network is simply composed of a sequence of layers. Moreover, most spaces incorporate kernel size with some works additionally searching for strides, pooling and fully-connected layers (Baker et al., 2016; Cai et al., 2018; Kandasamy et al., 2018).

To enhance search efficiency, Zoph et al. (2018), Zhong et al. (2018), and Liu et al. (2018) restrict the search to discovering transferable network building blocks (cells), while macro-architecture i.e. depth, width, and reduction blocks are chosen manually. While discovered cells transfer well across datasets, the corresponding macro-architectures vary significantly with dataset characteristics. For instance, Liu et al. (2018) manually choose 20 layers, 36 initial channels and two reduction blocks for CIFAR-10, but switches to 14 layers, 48 initial channels, and 4 reduction blocks for ImageNet. *This illustrates that as dataset difficulty changes, macro-architecture become increasingly critical, whereas the cell remains transferable.* Duan et al. (2021) jointly search for cells and macro-architecture to enhance NAS generalizability to diverse tasks. However, Wan et al. (2022); Yang et al. (2019) have shown that cell-only search spaces tend to exhibit limited performance variance despite containing an enormous number of candidates (e.g., on the order of 10^{18} architectures in Liu et al. (2018)), indicating that most of the performance variance is a result of macro-architecture.

Tan & Le (2019) focus solely on searching networks' macro-architecture/outer-skeleton (depth, width and input resolution) while using a pre-determined chain-structured architectural topology with fixed micro variables. Interestingly, without searching for micro variables, various model variants, from EfficientNet-B0 to EfficientNet-B7, achieve varying Top-1 ImageNet accuracy, i.e. from 77.1% to 84.3%, achieving the (then) state-of-the-art results. Moreover Siddiqui et al. (2025), also reports most of the performance gains using macro search.

AOWS (Berman et al., 2020) has a search space for finding suitable number of channels for each layer, which can get computationally intensive for a larger number of layers, each having different possible channels. The most closely related search space to us is that of CompOFA (Sahni et al., 2021). CompOFA down-samples

Table 2: An overview of search spaces containing macro-architecture search variables.

NAS Method	Macro Search Space Architectural Variables								
	Depth (Layers)	Width (Channels)	Operation Type	Kernel Size	Strides	Pooling Layers	Fully Connected Layers	Skip Connections	Input Resolution
Elsken et al. (2017)	✓	✓		✓				✓	
Zoph & Le (2016)	✓	✓		✓	✓			✓	
Real et al. (2017)	✓	✓		✓	✓			✓	
Kandasamy et al. (2018)	✓	✓		✓	✓	✓	✓	✓	
Cai et al. (2018)	✓	✓		✓	✓	✓	✓		
Baker et al. (2016)	✓	✓		✓	✓	✓	✓		
Siddiqui et al. (2025)	✓	✓	✓	✓					
Cai et al. (2020)	✓	✓		✓					✓
Tan & Le (2019)	✓	✓							✓
AOWS Berman et al. (2020)	✓								
CompOFA Sahni et al. (2021)	✓	✓							
Ours	✓	✓							

OFA's (Cai et al., 2020) huge search space based on the effectiveness of compound scaling of depth and width reported by Tan & Le (2019) and quantized-linear-scaling of Radosavovic et al. (2020). However, these studies are conducted for the ImageNet dataset, that has sufficiently large spatial resolution to experiment with several depth and width scaling rules. While it is possible to search for compound coefficients for datasets with unknown resolution, some of the datasets pose dimensions where scaling laws discovered for ImageNet may break. For instance, Tan & Le (2019) argues that larger spatial dimensions require deeper networks to capture complex features, whereas as shown in Table 1, Conway dataset has a very small resolution of 10X10 and it is quite possible that this dataset simply requires a wider rather than deeper architecture. The same might be true for extremely unequal dimensions of 6X768 for Windspeed data. We therefore allow exploration in either dimension that improves accuracy, which is neither exactly compound- nor quantized-linear-scaling, but rather an indirect approach to evaluating candidates in increasing dimensions. In Section 3.1, we propose a simple macro search space.

2.2 Search Strategy

Broadly, NAS search strategies fall into two main categories; black box optimization and one-shot techniques. The most notable black box NAS search strategies are based on reinforcement learning (RL) (Zoph & Le, 2016; Baker et al., 2016; Cai et al., 2018), evolutionary algorithms (Real et al., 2017; Suganuma et al., 2017; Elsken et al., 2017; Lopes & Alexandre, 2023) and bayesian optimization (BO) (Falkner et al., 2018). In general, black-box optimization techniques are computationally more demanding than one-shot techniques due to independent architectural evaluations, however, they are more robust and generalizable to new datasets. One-shot methods offer tremendous speed-ups, however, weight sharing across architectures is known to have several issues, the most notable of which is unstable relative rankings (Zela et al., 2020; Yu et al., 2019; Zhang et al., 2020).

Most of the black box methods evaluate all candidates under one training hyperparameter setting, however, various works have highlighted that training protocol plays a significant role in achieving higher accuracy and different architectures may require different hyperparameters (Yang et al., 2019; Bansal et al., 2022; Zela et al., 2018). Hence, a fixed training setting may favor some models over others leading to premature elimination of potential candidates. Moreover, since both architecture and training protocol is unknown, this effect may further pronounce for unseen data. Therefore, we opt for a search algorithm which propose progressively increase the evaluation budget for promising candidates. Our search algorithm therefore is most closely related to multi-fidelity search strategies proposed by Jamieson & Talwalkar (2016), Li et al. (2018) and Falkner et al. (2018).

Jamieson & Talwalkar (2016) propose a simple search strategy with Successive halving (SH). Candidates are first evaluated for a given initial budget. The lowest performing half is then discarded, and the budget is doubled for the remaining candidates, and this process is repeated iteratively until only one candidate remains. SH however, requires pre-determining a trade-off between the number of configurations evaluated

and the budget allocated to each. With a fixed total budget, using smaller budgets may prematurely discard strong candidates, while larger budgets may waste resources on poor configurations. To solve this problem, Li et al. (2018) propose Hyperband (HB) which divides the total budget into several combinations of number of configurations vs. budget and run SH for each. Some configurations run on maximal budgets while others on lowest. This suffers from the same issue of some models having an unfair advantage over others and may result in inaccurate relative rankings. Moreover, both SH and HB use randomly sampled configurations that may lead to worse final performance. (Falkner et al., 2018) propose to use bayesian optimization with Hyperband (BOHB) instead of random search and demonstrates strong anytime as well as final performance. BOHB, however, requires building a surrogate model during search which adaptively improves with the number of candidate evaluations. It is possible to apply BOHB for unseen data when there is a 'good' number of evaluations, but in case of larger data, the time constraint may not allow a large number of evaluations and a surrogate model with fewer evaluation points may not be accurate.

Although similar in principal to methods mentioned above, we propose a dataset-adaptive and architecture-aware fidelity budget. Moreover, instead of using random search (Jamieson & Talwalkar, 2016; Li et al., 2018) or surrogate performance predictors (Falkner et al., 2018), we use local search, since it has been shown to be a fast and competitive baseline for NAS (Den Ottelander et al., 2021; White et al., 2021a).

2.3 Performance Estimation

To identify the optimal architecture, early NAS approaches followed the traditional train/validate paradigm, but the large search spaces they explored required thousands of GPU days (Zoph & Le, 2016; Zoph et al., 2018; Real et al., 2017). To reduce the evaluation cost, NAS methods often rely on low-fidelity proxy estimates, such as using fewer training epochs as in learning curve extrapolation methods (Domhan et al., 2015; Yan et al., 2021; Baker et al., 2017), or training on small data subsets (Klein et al., 2017; Prasad et al., 2024; Shim et al., 2021; Na et al., 2021). However, unseen datasets may exhibit varying convergence rates, making it difficult to predefine a fixed early-termination criterion that generalizes reliably across datasets. Moreover, the achievable accuracy is unknown *a priori*; therefore, the criterion for selecting a representative subset does not exist to begin with. Other low-fidelity estimation methods use reduced input resolutions (Chrabaszcz et al., 2017), or searching with decreased network width (Zoph et al., 2018). While these approximations greatly lower computation, large fidelity gaps can significantly alter rankings (Zela et al., 2018). We discuss these further in Section 3.2.

Another line of research is zero-cost proxies (Krishnakumar et al., 2022), which estimate an architecture's performance with minimal or no training. Such decoupling of architectures from target dataset contradicts the fundamental goal of NAS; discovering dataset specific architectures. Moreover, studies such as (Chen et al., 2021a; Ning et al., 2021) show that proxies developed for modular search spaces do not transfer well to macro spaces.

Some works reuse learned weights by transferring to architecturally similar candidates to speed up evaluation (Real et al., 2017; Chen et al., 2016). However, this gives an unfair advantage to candidates evaluated later in the search process. A similar idea is that of using a supernet, first introduced by Saxena & Verbeek (2016) and later popularized by Liu et al. (2018), where subgraphs inherit weights from a large over-parameterized network. However, Yu et al. (2019) demonstrate that weight sharing can produce inaccurate rankings. Another strategy is to use model-based accuracy predictors (Wen et al., 2020; White et al., 2021b) but these require architecture–accuracy paired data, which itself is costly to acquire.

In this work, we favor rigorous evaluation of a compact yet architecturally diverse candidate set over shallow assessment of an expansive but largely homogeneous architecture space. We describe our detailed method in Section 3.2.

3 Methodology

An overview of a standard NAS pipeline, along with the summary of our proposed framework, is shown in Figure 1. Since the datasets are assumed to be unseen, we employ only broadly applicable preprocessing techniques such as normalization and image resizing. Likewise, under strict time constraints, we fix the

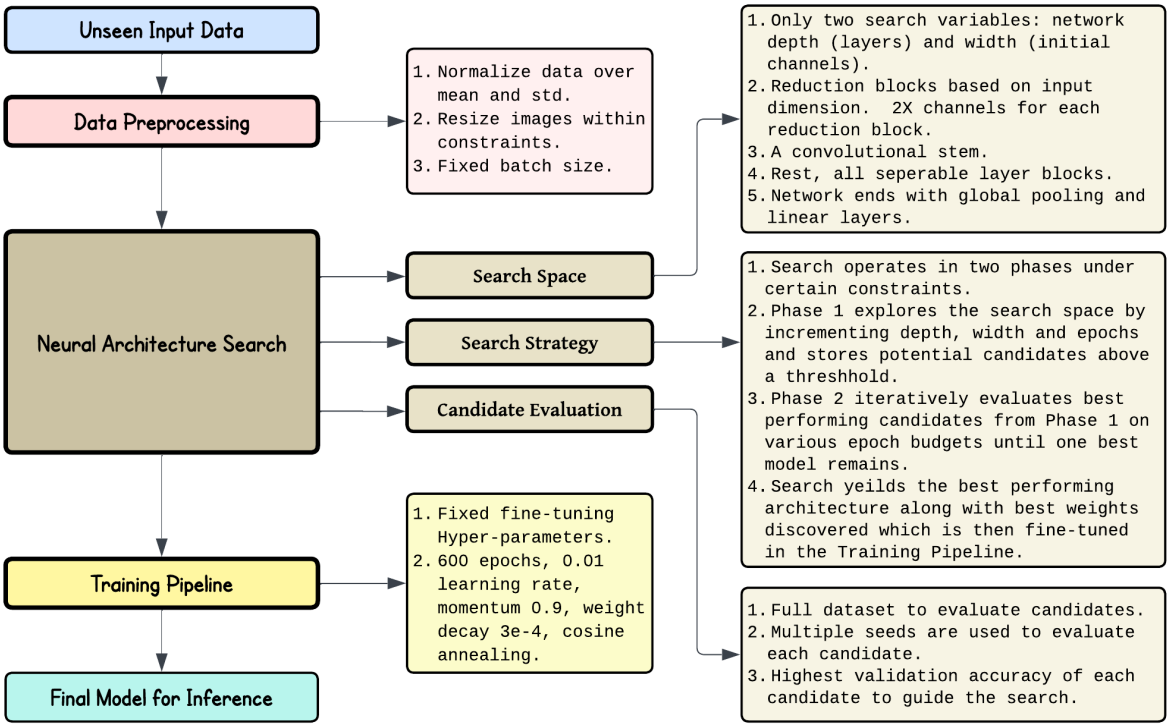

Figure 1: An overview of the NAS Unseen-Data Pipeline. Data Pre-processing and Training Pipeline is fixed for all datasets. Our contribution lies in the Neural Architecture Search block: a minimal yet flexible search space, a multi-fidelity search strategy and a high-fidelity performance estimation.

final training hyperparameters across all datasets and focus solely on NAS, although joint optimization of hyperparameters and architectures has been shown to improve performance (Bansal et al., 2022; Zela et al., 2018). Next, we describe our search space design followed by performance estimation methodology and search strategy.

3.1 Search Space Design

As discussed in Section 2.1, we focus exclusively on macro-only search due to its sufficient representational capacity and reduced search complexity. More specifically, we limit the search to just two integer-valued variables, i.e. number of layers (depth) and number of initial channels (width) of the network. Next, we describe why and how certain search variables shown in Table 2 can effectively be removed from the design space.

3.1.1 Macro-level Hyperparameters

We limit the depth to a maximum of 100 layers i.e. not targeting very deep architectures and propose to have a simple chain-structured architecture where a network is a sequence of layers without any branching layers, therefore, we drop *Skip Connections* variable from the design space. Moreover, we confine the channel search to initial layer with a fixed channel-doubling rule whenever the spatial resolution halves, as in Liu et al. (2018). Table 2 shows that Tan & Le (2019) is the only one searching for the resolution under the intuition that higher-resolution data require deeper and wider networks to capture complex patterns. While resolution search provides a decent accuracy-latency trade-off for ImageNet dataset, in practice, unseen-data can exhibit arbitrary resolutions and it is not something we can control or anticipate. Moreover, for the core

objective of achieving higher accuracy, it is desirable to leverage the available spatial information without excessive resampling. The reason is that up-scaling does not add more to information than there already is but substantially increases network size and hence computational cost, while down-sampling may degrade certain clues in the data. Moreover, natural dataset dimensions can effectively be utilized to calculate a suitable number of reduction blocks. This simple technique offers two advantages: 1) a dataset adaptive sub design space is crafted within suitable macro-level bounds, that is, search only what might be effective, and 2) a critical aspect of macro-architecture, that is, the number of reduction blocks, is 'discovered' prior to search, as opposed to a manually determined fixed number as in Zoph et al. (2018), Liu et al. (2018), and Siddiqui et al. (2025) or explicitly searching for these as in Baker et al. (2016). Therefore, we adopt the native resolution and hence drop *Resolution* search variable.

3.1.2 Micro-Architecture Hyperparameters

Following Siddiqui et al. (2025), we use FCN like architectures (Long et al., 2015), replacing fully-connected layers with global pooling and pooling layers with stride-2-convolutions for reduction layers (stride-1 otherwise). It should be noted that global pooling size changes with the number of channels in the final feature map, hence, it is still flexible to accommodate varying datasets. Moreover, for reducing spatial image dimensions, stride-2-convolution is better than pooling due to learnable parameters. These design decisions drop micro-hyperparameters *Strides*, *Pooling Layers* and *Fully Connected* from our search space. Further, as shown in Table 2, nearly all spaces include kernel size search although a 3×3 kernel offers an effective accuracy-computational trade-off. Moreover, Siddiqui et al. (2025) reports only marginal gains during kernel and operation search (i.e. 95.82% vs. 96%) while possessing substantial performance variance across the full search space (76.11% for the worst and 94.65% for the best randomly sampled networks). Therefore, following Tan & Le (2019), we fix *Kernel Size* as well as *Operation Type* variables. However, our layer is composed of ReLU-Drop-Seperable3x3-BN instead of mobilenet/resnet blocks in Tan & Le (2019). We assume this is a transferable cell/block/layer across different datasets.

To this end, we propose a compact search space defined by two independently acting variables, ***layers*** and ***channels***, as summarized in Table 2. When combined with dataset-dependent input resolutions, searchable network depth and width induce substantial non-compound architectural diversity, yielding a flexible yet expressive design space for varying unseen-data. With these search variables defined, we can now formally state the architecture search problem.

3.1.3 Search Problem

Let $\mathcal{D}$ denote a dataset with fixed input resolution $R_{\mathcal{D}} \in \mathbb{N}^2$. An architecture is denoted by $x := (L, C)$, where $L \in \mathbb{N}$ and $C \in \mathbb{N}$ represent the number of layers and channels, respectively. For a given dataset $\mathcal{D}$, we define the dataset-specific architecture search space as

$$\mathcal{X}_{\mathcal{D}} \equiv \mathcal{X}_{\mathcal{D}}(R_{\mathcal{D}}) = \{(L, C) \mid L \in [L_{\min}, L_{\max}]_{\Delta_L},\ C \in [C_{\min}, C_{\max}]_{\Delta_C}\}\,, \tag{1}$$

where Δ_L and Δ_C specify the discretization step sizes for L and C, respectively, inducing finite discrete sets of admissible layer and channel counts within the bounds $[L_{\min}, L_{\max}]$ and $[C_{\min}, C_{\max}]$. The dependence on $R_{\mathcal{D}}$ reflects that admissible architectural variations are naturally bounded by the spatial resolution of the dataset.

For an architecture $x \in \mathcal{X}_{\mathcal{D}}$ and network parameters $\theta \in \mathbb{R}^{d(x)}$, let $\mathcal{L}_{\text{train}}(\theta, x; \mathcal{D})$ and $\mathcal{L}_{\text{val}}(\theta, x; \mathcal{D})$ denote the training and validation losses, respectively. We assume that for every $x \in \mathcal{X}_{\mathcal{D}}$, the inner optimization problem admits at least one minimizer.

For a fixed architecture x, the optimal network parameters are defined as

$$\theta^*(x; \mathcal{D}) \in \arg\min_{\theta}\ \mathcal{L}_{\text{train}}(\theta, x; \mathcal{D}). \tag{2}$$

The dataset-specific optimal architecture is obtained by solving the following bilevel optimization problem, in the spirit of Liu et al. (2018):

$$x^*_{\mathcal{D}} \in \arg\min_{x \in \mathcal{X}_{\mathcal{D}}} \mathcal{L}_{\text{val}}\big(\theta^*(x;\mathcal{D}), x; \mathcal{D}\big). \tag{3}$$

For the challenge, the search is restricted by time and GPU memory constraints. Let $\text{Params}(x)$ and $\text{Time}(x)$ denote the number of trainable parameters and the computational cost associated with architecture x, respectively. Given a maximum allowable model size $P_{\max}$ and search budget $T_{\max}$, the feasible architecture set is

$$\widetilde{\mathcal{X}}_{\mathcal{D}} = \{x \in \mathcal{X}_{\mathcal{D}} \mid \text{Params}(x) \leq P_{\max},\ \text{Time}(x) \leq T_{\max}\}\,. \tag{4}$$

Accordingly, the outer optimization in equation 3 is performed over $\widetilde{\mathcal{X}}_{\mathcal{D}}$.

3.1.4 Search Space Complexity

Since unseen datasets may exhibit varying input resolutions, our search space implicitly accounts for this variability by parameterizing architectures with the triplet (L, C, R), where the resolution R may vary over a finite discrete set $\mathcal{R} \subset \mathbb{N}^2$ across datasets. Let L_n and C_n denote the number of discrete values induced by $[L_{\min}, L_{\max}]\Delta_L$ and $[C\min, C_{\max}]\Delta_C$, respectively, and let $R_n := |\mathcal{R}|$ denote the number of admissible resolutions. The total number of possible architectures in the search space is then $N\text{arch} = L_n \times C_n \times R_n$. In this work, the depth varies between 10 and 100 layers and the channel count between 16 and 2048, with step sizes of 2 and 48, respectively. Moreover, we allow resolutions between 8×8 and 128×128, including non-square configurations. The total number of possible configurations is therefore 28.9×10^6. For comparison, the search space of Liu et al. (2018) contains approximately 10^{18} candidates but with limited variance (Yang et al., 2019). For a single given dataset, however, since the resolution is fixed to the dataset's native resolution $R_{\mathcal{D}}$, the total number of candidates in the search space reduces to $N_{\text{arch}} = L_n \times C_n$, resulting in 1,978 candidate architectures.

3.2 Performance Evaluation

Existing NAS works use a part of training dataset for search, except (Siddiqui et al., 2023; 2025), where some works simply use a random subset of data (Zoph & Le, 2016; Real et al., 2017; Zoph et al., 2018; Liu et al., 2018), while others propose to find a representative data subset (Prasad et al., 2024; Shim et al., 2021; Na et al., 2021). However, this is not generally applicable for the reasons discussed in the subsequent section.

Another, known yet overlooked aspect is effect of random seed on the rankings of the candidates (Yang et al., 2019; Li & Talwalkar, 2020). NAS involves numerous stochastic components, and established best practices proposed by Lindauer & Hutter (2020) and Li & Talwalkar (2020) recommend reporting random seeds used for search and training to ensure reproducibility and fair comparison. However, beyond reproducibility, randomness introduced by stochastic gradient descent (SGD), weight initialization, and data-loader shuffling can cause substantial variation in validation accuracy across different seeds leading to drastic shifting of relative candidate rankings. In the subsequent section, we present and discuss the details of our experiment to study the effect of data subsets and random seeds.

3.2.1 Effect of Data Subset and Random Seeds

We conduct a simple experiment where we train a single model with 10 distinct random seeds for 100 epochs. Moreover, we repeat the experiment using 12%, 25%, 50% and 100% random data samples of CIFAR-10. Fig. 2 shows, validation accuracy variance using different subsets of CIFAR-10 data at different epochs. As the number of seeds increase, a new minimum or maximum of validation accuracy is found, enlarging performance gap across all data. This effect is more pronounced for 12%, 25% and 50% subsets with the exception of 12% data at epochs 5 and 6. This is only because the subset is too small to reach higher accuracy. However, when the effect is observed at all epochs collectively, the lowest variance is observed for 100% data with a gradual decrease in variance with epoch increase.

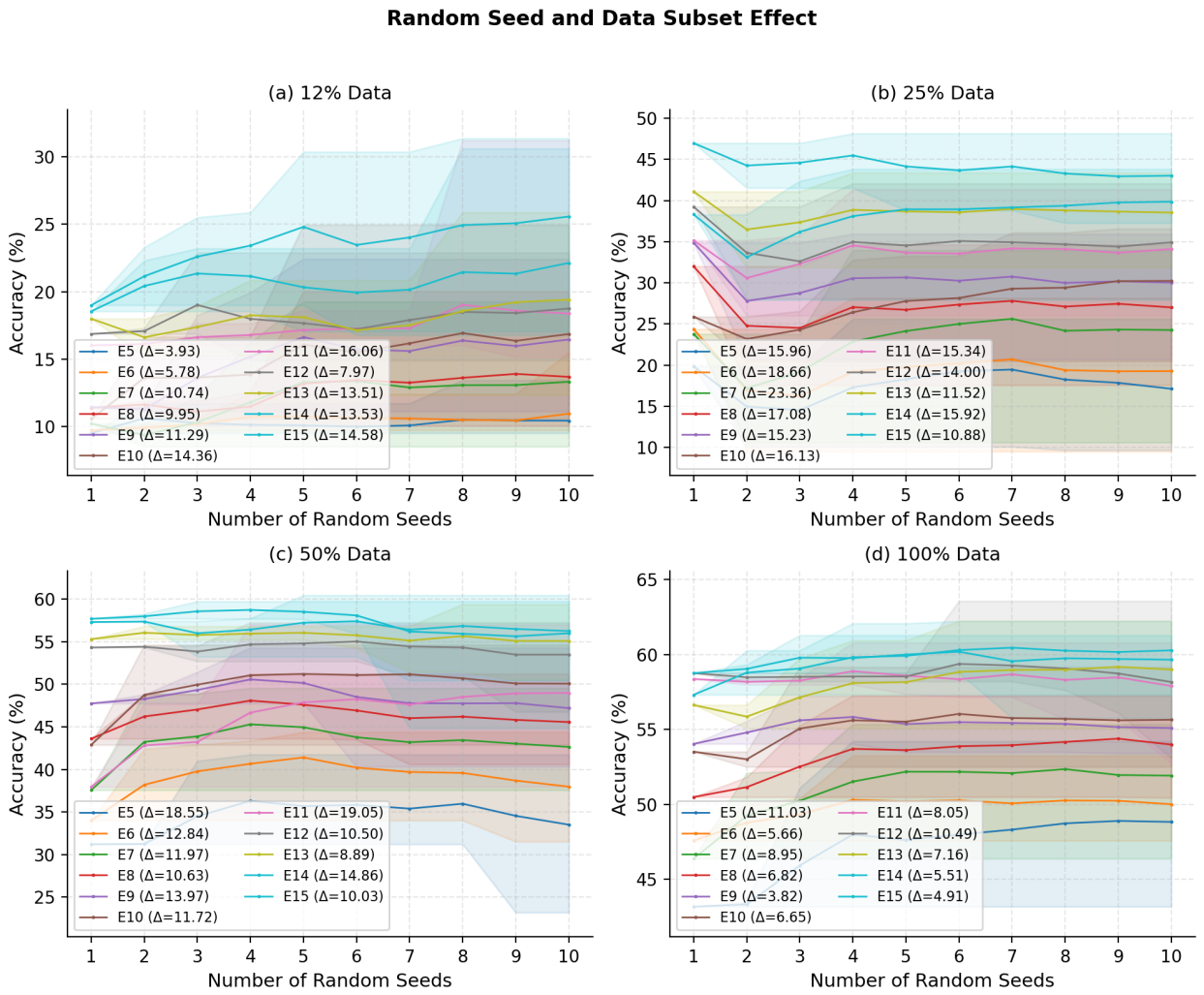

Figure 2: Effect of random seeds at various data and training fidelities. As the number of runs increases with random seeds, new minimum and maximum validation accuracies are observed. Although the performance gap decreases at higher epochs but is still large enough to ignore and fair comparison of architectures under a single seed is questionable.

Further, the overlapping mean accuracy lines show that a model achieving a higher accuracy with a certain seed on a particular epoch may produce a lower accuracy at next epochs on different seeds, (e.g. see figure 2 (d) at epochs E11 and E13). Model performs better at 11 epochs on seeds 1-5, whereas it performs better at 13 epochs on seeds 6-10. Ideally, any model should improve accuracy at increased evaluation budget when this far from saturation ($\tilde{6}0\%$ validation accuracy). However, this effect hasn't been captured unless multiple seeds are employed.

These results indicate three important aspects: (1) relying on a single seed and a fixed evaluation budget may hinder accessing the capability of a given model, (2) using 100% of the data generally produces the least performance variance across seeds and various epochs, (3) with higher evaluation budgets, relatively lower variance can be achieved, however, there is no single best evaluation budget.

We would also like to highlight that even at 100 epochs, the difference across seeds remains fairly high, that is, 6.69%, 6.37%, 4.39% and 5.76%, for 12%, 25%, 50% and 100% data, respectively. Moreover, such high fidelity evaluation with multi-seed is not computationally suitable. Therefore, we have kept our evaluation budget between 1-20 epochs and shown the slightly trimmed version in figure 2.

To this end, we show that multiple seeds provide candidates with several opportunities to reach their best performance. Therefore, in our experiments, we use five random seeds to evaluate candidates at various epoch budgets and use the highest-seed accuracy. This is because, during search, we are generally navigating models from small to large and increasing evaluation budget, hence ideally expecting a performance increase. Moreover, our core objective is to achieve the highest accuracy and we eventually use the best weights found during search as initial weights for final training, expecting a lucky initialization to yield higher accuracy.

3.2.2 Closing the Search-Training Data Domain Gap

Existing works often assume that an architecture discovered using a random or representative subset of the training data will also perform well when trained on the full dataset. This assumption holds only if the subset is truly representative and capable of yielding performance comparable to that obtained using the entire dataset. However, randomly sampled subsets generally violate this assumption, while selecting an appropriate coreset (Prasad et al., 2024; Shim et al., 2021; Na et al., 2021) is itself a challenging problem. Moreover, for unseen datasets, the optimal accuracy achievable on the full training set is unknown a priori, yet this metric is required to identify a representative subset in the first place. In addition, training data are typically augmented during final training to improve performance, implying that a representative subset should also account for such augmentations. Otherwise, an architecture discovered using a subset is optimized for that subset rather than for the final training distribution. Consequently, subset-based search decouples the architecture discovery process from the target dataset, undermining the core objective of NAS: discovering dataset-specific architectures.

In this work, we perform both search and training on the full dataset to ensure that the architecture identified during the search phase is genuinely optimal for the given data and can effectively exploit the entire training set to maximize final performance. An additional benefit of using the full dataset is that warm-up weights obtained during search can be reused during subsequent training. We note that suitable data augmentation strategies are also unknown for unseen datasets and are not optimized in this work; therefore, the provided training datasets are used as is. Finally, in Section 4.3, we conduct extensive experiments to show why using the entire data is useful for NAS generalization to unseen data.

3.3 Search Strategy

Using the entire dataset for search as well as evaluating each candidate across multiple seeds aids high-fidelity performance estimates but becomes computationally prohibitive. Therefore, we propose a bi-phase multi-fidelity search strategy as shown in algorithm 1. Phase 1 progressively expands architectures and increases evaluation budget only as required by the dataset, whereas Phase 2 uses an always-increasing, unique, candidate-aware budget to enforce an equal opportunity across candidates.

3.3.1 Phase 1: Progressive Architectural Expansion

The search begins in Phase 1 using a baseline model's validation score. The algorithm then explores the search space by first increasing network depth and subsequently increasing width. Depth or width is expanded as long as each modification yields an improvement in validation accuracy. Any candidate that surpasses the current best score is retained for further evaluation in Phase 2. When neither depth nor width leads to additional gains, the training budget is increased by one epoch, and depth–width exploration resumes. This process repeats until reaching either a time limit or a parameter budget. Phase 1 concludes with a pool of promising candidates, each identified as the best-performing configuration for its respective depth–width–epoch setting.

3.3.2 Phase 2: Iterative Multi-Fidelity Candidate Refinement

Phase 2 operates iteratively by evaluating models under progressively increasing training budgets. Candidates are first sorted in ascending order by parameter count, and each smaller model is granted one additional training epoch. This provides lightweight models with improved learning opportunities, mitigating concerns that their performance may appear inferior simply due to insufficient training budgets. As in Phase 1, any candidate that exceeds the current best validation score is retained for further evaluation. After all Phase 1

Algorithm 1: NAS Search Algorithm

Input: Given *Dataset*, Total Time T, and Maximum Parameters P_{max}
Initialization:
Split T into T_{Phase1}, T_{Phase2}, T_{train} and specify search bounds with depth L, width C, evaluation epochs E, maximum parameters P_{max}, and empty candidate model list $List = [.]$
Train baseline model, get validation accuracy Acc_{best} and evaluate next model with $L = L + 1$
Phase 1: Architecture Expansion
1. ***while*** $Acc_{next} > Acc_{best}$, add model to *List* and update Acc_{best}, continue **depth** expansion
2. ***while*** $Acc_{next} > Acc_{best}$, add model to *List* and update Acc_{best}, continue **width** expansion
3. Increment **epochs** and ***Repeat*** 1, 2 until $t < T_{Phase1}$ **AND** $P < P_{max}$
Return promising candidate *List* to Phase 2
Phase 2: Iterative Elimination
while Length of $List > 1$ ***do:***
1. **Sort** models w.r.t to number of parameters
2. Starting from 2nd largest, evaluate each smaller model with $E = E + 1$
3. **Eliminate** ***if*** $Acc_{next} < Acc_{best}$ ***else*** update Acc_{best}
4. **Update** *List*
5. **Repeat** until $t < T_{Phase2}$ **AND** $E < E_{max}$
Return *Architecture and weights for* Acc_{best}

candidates have been assessed, they are resorted by parameter count, and the iterative evaluation continues until all but one candidate have been eliminated. If the search reaches the time limit before convergence, the model with the highest validation accuracy so far is selected as the final architecture. We observe best anytime as well as the best final performance for the challenge datasets, as discussed in Section 5.5.

4 Experiments and Results

Our work primarily focuses on the generalization of NAS beyond standard benchmarks. However, a large body of NAS research relies on the CIFAR-10 dataset (Krizhevsky et al., 2009). Hence, to enable a comparison against existing methods, we evaluate our framework on CIFAR-10 which consists of 60,000 images, each with a spatial resolution of 3×32x32 . These images are evenly distributed across 10 classes, with 50,000 used for training and 10,000 for testing.

Moreover, our primary focus is macro-search, however, we also evaluate our approach within the widely used DARTS search space (Liu et al., 2018) to facilitate a fair comparison against broader NAS literature. Accordingly, this section is divided into two parts: Section 4.1 presents the details and results of our proposed **macro-search** framework, while Section 4.2 examines the transferability of our approach to the DARTS search space (**micro-search**). Section 4.3 presents details of various ablation experiments.

4.1 Macro-Search Experiments

We conduct macro-search experiments under two different settings. In the first setting, a layer is defined as a simple ReLU–Dropout–Separable-3×3–BN block, as described in Section 3.1. We refer to this as the *Plain-layer* setting. In the second setting, a layer is defined as a cell, following (Liu et al., 2018). Here, we perform macro-search using a DARTS-discovered cell used as a layer, and refer to this as the *DARTS-layer* setting.

4.1.1 Macro-Search Settings

For *Plain-layer* setting, we run search with L_{min}=8, L_{max}=100, C_{min}=16, C_{max}=2048, L_{res}=2, C_{res}=32, E_{min}=1 and E_{max}=50. We use the entire CIFAR10 training set for search and employ 5 random seeds for each candidate evaluation. Further, we restrict the maximum model size to 4.2M parameters.

Table 3: Comparison with state-of-the-art NAS architectures for CIFAR-10. Top block presents methods supporting macro-architecture discovery while bottom block presents modular (cell-based) architecture search methods.

NAS Method	Test Err. (%)	Params (M)	Search Cost (GPU-days)	Search Space	Search Algorithm
NAS-RL (Zoph & Le, 2016)	3.65	37.4	22400	Macro	RL
Meta-QNN (Baker et al., 2016)	6.92	11.2	100	Macro	RL
EAS (Cai et al., 2018)	4.23	23.4	10	Macro	RL
Evolution (Real et al., 2017)	5.40	5.4	2600	Macro	EA
Genetic-CNN (Suganuma et al., 2017)	5.98	1.7	14.9	Macro	EA
NASH-Net (Elsken et al., 2017)	5.20	19.7	1	Macro	EA
RandGrow (Hu et al., 2018)	3.38	3.1	6	Macro	RS
Petridish (Hu et al., 2019)	2.83	2.2	5	Macro	Gradient
NASBOT (Kandasamy et al., 2018)	8.69	N/A	1.7	Macro	SMBO
NSGA-NET (Lu et al., 2019)	3.85	3.3	8	Macro	SMBO
Macro-NAS (Lopes & Alexandre, 2023)	4.23	6.7	1.03	Macro	EA
EGNAS (Siddiqui et al., 2025)	3.17	2.49	**0.43**	Macro	Greedy
Ours (Plain-layer)	3.38	3.7	0.54	Macro	Multi-fidelity
Ours (DARTS-layer)	**2.39**	10.4	1.18	Macro	Multi-fidelity
NASNet-A (Zoph et al., 2018)	2.40	27.6	2000	Modular	RL
AmoebaNet-B (Real et al., 2019)	2.13	34.9	3150	Modular	Evolution
DARTS (Liu et al., 2018)	2.76	3.3	4	DARTS	Gradient
GDAS (Dong & Yang, 2019)	2.82	2.5	0.17	DARTS	Gradient
PC-DARTS (Xu et al., 2019)	2.57	3.6	0.1	DARTS	Gradient
P-DARTS (Chen et al., 2021b)	2.5	3.4	0.3	DARTS	Gradient
AGNAS (Sun et al., 2022)	2.46	3.6	0.4	DARTS	Gradient
FP-DARTS (Wang et al., 2023)	2.5	3.9	0.08	DARTS	Gradient
RF-DARTS (Zhang et al., 2023)	2.6	4.6	0.4	DARTS	Gradient
IS-DARTS (He et al., 2024)	2.56	4.25	0.42	DARTS	Gradient
SWAP (Peng et al., 2024)	2.48	4.3	0.004	DARTS	Training-free
NCD-NWOT (Kang et al., 2025)	**2.44**	4.4	**0.002**	DARTS	Training-free
Ours (Micro-Search)	2.69	4.2	0.94	DARTS	Multi-fidelity

For *DARTS-layer* setting, we change C_{min} and C_{res} to 8 and 16, respectively and increase the maximum parameter budget to 12.5M since a single *DARTS-layer* contains significantly more parameters than a *Plain-layer*. We reduce the channel resolution and increase the parameter budget to allow evaluation of a reasonable number of candidates.

4.1.2 Evaluation and Training Settings

For both search and training, we follow the settings used by Liu et al. (2018); Xu et al. (2019); Chen et al. (2021b) including augmentations such as random crop, horizontal flip and cutout. All candidates use SGD with momentum 0.9, weight decay of 3e-4 and initial learning rate of 0.025. However, for search, the learning is fixed at 0.025 and is not decayed as for final training and the number of epoch vary adaptively across candidates as required by our multi-fidelity search algorithm unlike the fixed 600 for final training.

For final training of *Plain-layer* network, we use the best weights discovered during search, whereas for the final training of *DARTS-layer* network, we use the exact code and settings used by Xu et al. (2019), (training from scratch) and only change the number of layers and channels as discovered by our method.

4.1.3 Macro-Search Results

Using a single GPU [2], the search takes a maximum 0.54 and 1.18 GPU days for *Plain-layer* and *DARTS-layer* settings, respectively. For *Plain-layer*, a total of 33 candidates are evaluated at different budgets

[2]All experiments are conducted on an Nvidia Quadro RTX 8000

during search, with 19 and 16 models being evaluated in Phase 1 and 2, respectively. However, the number of unique models within 4.2M parameter bounds is limited to 16, where promising candidates are re-evaluated at different budgets ranging between 2-21 epochs. The evaluation cost across candidates vary due to adaptive evaluation fidelity. For *DARTS-layer* search, a total 24 candidates are evaluated with 14 and 12 in Phase 1 and 2, respectively. The 12.5M parameter constraint restrict the maximum unique models to just 12. Promising candidates are evaluated for between 2-16 epochs.

Top block of Table 3 shows a comparison with several macro-NAS methods. We achieve a competitive test error of **3.38%**, *(best across 10 runs)*, with our *Plain-layer* setting while achieving the lowest error of **2.39%**, *(3 search runs yielding same model twice, hence trained only once)*, for our *DARTS-layer*. This is the lowest error among all search spaces except AmoebaNet (2.13%, 2nd block), which is order of magnitude expensive, i.e 3150 GPU days and has a modular space different than DARTS. While performance results across different studies are not directly comparable due to variations in search spaces and optimization strategies, it is nonetheless informative to consider this result in a broader context of prior work on macro-search.

It should also be noted that our results are the better than all DARTS search space method (bottom block), even though we simply use a DARTS (2nd order) cell. Interestingly, for *DARTS-layer*, our search algorithms discovers a model of 16-64 (layers-channels) as compared to the default DARTS setting of 20-36, indicating the importance of macro-search.

4.2 DARTS-Search Experiments

To compare our work with broader NAS community, we choose the well studied and widely adopted DARTS search space Liu et al. (2018). DARTS search space offers cell-level architecture discovery while keeping the macro-architecture fixed (20 layers/cells and 36 initial channels for CIFAR10). Since our search strategy is centered around navigating layers and channels variables, we slightly modify it to work with DARTS space.

4.2.1 Macro to Micro Search

As described in section 3.3, for macro search, we begin Phase 1 with a tiny baseline model and navigate depth and width alternatively, and in increasing order. If the validation accuracy does not improve, the number of evaluation epochs is increased. This allows us to collect a pool of promising candidates where each subsequent candidate is larger in terms of learnable parameters. Candidates which do not perform better than the previous best are discarded. For DARTS space, since the macro-architecture is fixed, we rely on the number of parameters of networks to create suitable candidates for search. To achieve this, we fix the smallest and largest possible network by determining the respective cell size. The smallest network is created by choosing only one learnable operation i.e. *seperable-3X3* for the normal cell, while setting all the rest as either *skip* or *none* operations, for both normal and reduction cells. The largest network is created by setting all connections to *seperable-5X5* both cells. Then we create a fixed number of equally spaced (parameter-count-wise) candidate architectures by randomly sampling cells which satisfy the given parameter requirements for each candidate. Hence, we acquire a small yet diverse search space, where cells impact the final architecture significantly even though the macro-architecture is fixed.

Phase 1 search strategy is also modified slightly since depth-width exploration is replaced by parameter count. When a certain candidate does not improve, we evaluate it with one additional epoch. This has a similar effect to increasing epochs of the latest candidate when neither depth, nor width increase yields any improvements during macro-search. The rest of the algorithm remains the same.

4.2.2 Micro-Search Settings

For micro search, we fix the number of layers and channels to 8 and 16, respectively and hence other macro-search parameters are irrelevant. This is similar to the setting used by (Liu et al., 2018; Xu et al., 2019) when creating a super network for search. We limit the maximum number of equally-spaced candidate networks to 16, excluding smallest and largest network. We use the entire CIFAR10 training set for search and employ 3

Table 4: Accuracy and search cost trade-off when using different subsets of data.

	Macro-Search						Micro Search		
	Plain-Layer			DARTS-Layer					
Dataset Size	**Model Size Params (M)**	**Accuracy**	**Search Cost GPU Hours**	**Model Size Params (M)**	**Accuracy**	**Search Cost GPU Hours**	**Model Size Params (M)**	**Accuracy**	**Search Cost GPU Hours**
D-12	1.33	95.23	1.8	0.15	94.16	2.67	1.9	96.80	4.6
D-25	1.95	95.94	4.9	1.38	97.22	11.3	2.9	97.28	9.2
D-50	2.13	95.93	6.5	12.2	97.55	20.4	2.9	97.28	14.1
D-100	2.71	**96.30**	13	10.4	**97.61**	28.3	4.2	**97.31**	22.7

random seeds for each candidate evaluation. Further, the maximum final model size corresponding to 20-36 setting is by default 5.6M parameters.

4.2.3 Evaluation and Training Settings

The evaluation and training settings remain the same as for macro-search. For final training, we use the cell discovered by our algorithm and we use the exact code and settings used by Xu et al. (2019). It should be noted that layers-channels are fixed at 20-36 for this experiment.

4.2.4 Micro-Search Results

The search takes 0.94 GPU-days for this setting. Bottom block of Table 3 shows a comparison with several DARTS related methods. We discover a cell creating a model of size 4.2M and achieve a competitive error rate of **2.69%**. It should be noted that DARTS search space contains 10^{18} possible cell configurations, whereas we just pick 16 random cells based on their learnable parameters, and hence model sizes. In this search space, some models under 3M paramters, producing error rates > 3 exist, (see Table 4). However, our proposed search strategy chooses better architecture indicating its effectiveness across fewer yet architecturally diverse search space.

4.3 Ablation Studies

In this section, we conduct experiments to study the effect of using data subsets on discovered networks and the associated trade-offs. We present results for both macro- and micro-search (DARTS).

4.3.1 Models Discovered Using Data Subsets

For *Plain-layer* macro-search, we run search 10 times for each of the data subsets of 12%, 25%, and 50% and the entire 100% training set. We refer this to as D-12, D-25, D-50 and D-100 settings in subsequent text and in Table 4. We use the same search settings as described in Section 4.1.1. Table 4 (left) reports the mean number of parameters and accuracy achieved across 10 search runs. Although, searching on subsets is faster, but a clear decline in final accuracy is observed. Moreover, searching with more data tends to discover larger models in terms of learnable parameters. This shows the dataset-adaptive nature of the proposed search strategy. Exactly similar trend is observed for *DARTS-layer* macro-search, with the exception of D-50 giving a larger 12.2M model against one out of 3 runs, as shown in Table 4 center column. The overall trend nonetheless is the same.

For micro-search experiments, among 16 cell configurations, D-12, D25, D-50 and D-100 consistently yield genotypes G1, G5, G5 and G11 (we name the 16 cells/genotypes as G1-G16, from smallest to largest), respectively, across 3 different runs. The dataset-adaptive model assignment trend remains consistent in cell based search space too. This shows that a high-fidelity evaluation mechanism coupled with multi-fidelity search strategy can generalize well to a given dataset, specifically, when the candidate architectures are substantially different in their representational capacity.

4.3.2 Search Cost

The search cost reported in Table 4 is for each candidate evaluated with 5 random seeds. Assuming the search will roughly run the same track, the search cost can be reduced 5X using a single seed. However, the discovery of the best final architecture is can not be guaranteed.

5 Unseen-Data Challenge Results and Analysis

This section presents the details of our winning solution for the 5th Unseen-Data Challenge which was conducted as part of the AutoML conference 2025. This section is divided into 5 subsections. Section 5.2 describes the unseen-datasets introduced in the latest challenge, section 5.3 is focused on search and training settings, section 5.4 discusses the challenge results and section 5.5 discusses post-challenge analysis.

5.1 Challenge Details

5.1.1 Overview

The main objective of the challenge is to design an end-to-end pipeline that includes data preprocessing, neural architecture search, and final training that can generalize out-of-the-box to diverse unseen datasets. The annual challenge is conducted in three phases. Phase 1, beginning around May, focuses on familiarizing participants with the challenge rules and the provided starter toolkit. In Phase 2, participants submit their solutions, which are evaluated by the organizers over a shorter duration to verify compliance with the challenge requirements before advancing to the final 24-hour run in Phase 3. Phase 2 runs from July 1 to July 31, during which up to seven submissions may be evaluated. Participants are also permitted to download and test their solutions on datasets from previous editions of the challenge (Geada et al., 2024). Notably, Phase 2 datasets are used only for validation of submission correctness and are unrelated to the unseen datasets used in Phase 3. Phase 3 begins on August 1 and ends until all submissions are evaluated by the organizers.

5.1.2 Rules

The challenge imposes two key requirements: (1) participants must strictly adhere to a 24-hour time budget across all three datasets, and (2) submissions must run on a single GPU and adapt to the available memory. Solutions that violate these time or memory constraints are disqualified.

5.2 Datasets

As shown in Table 1, three datasets, each with a different number of output classes, input dimensions, and number of training, validation and test samples were introduced in the 2025 NAS Unseen-Data challenge. The Conway dataset (Towers et al., 2025b) is an algorithmically generated dataset based off John Horton Conway's Game of Life. The goal of this dataset is to predict the number of cells alive in the next generation. This task is relatively simple for a human to do and should theoretically be simple for Machine Learning algorithms. Cryptic (Ericsson et al., 2025a) dataset is related to language understanding and solving cryptic clues that are generally hard for humans too, and is more of a medium difficulty compared to the others. Windspeed (Ericsson et al., 2025b) dataset is based on predicting the average wind speed in an area given temperature and pressure values. It is a challenging dataset and the baseline only achieves about 13.5% accuracy. This range of difficulties was created to test whether the algorithms can deal with the varying scenarios of real-world applications. It is interesting to note that not only the dimensions of these datasets are non-trivial, but the varying number of training/validation samples also pose a challenge in determining an evaluation scheme and precisely ranking different candidates given limited time for search and training. Moreover, suitable data pre-processing and post search training protocols are also unknown, which we aim to address in the future work.

Table 5: NAS Unseen-Data 2025 Challenge Results: Baseline scores are provided by the organizers on a manually designed network. Raw Acc. represents accuracy achieved on the held-out test set, whereas Adjacent (Adj.) Score represents score relative to the baseline. Official leaderboard can be found at: NAS Unseen-Data Challenge 2025 Leaderboard

Solutions	**GameOfLife**		**WindSpeed**		**Cryptic**		**Final Score**
	Raw Acc.	Adj. Score	Raw Acc.	Adj. Score	Raw Acc.	Adj. Score	
Basline	41.0	-	13.492	-	71.104	-	-
3rd-Place	99.97	9.995	13.776	0.033	72.265	0.402	10.430
Runner-up	99.66	9.942	15.278	0.206	73.235	0.737	10.885
Winner (Ours)	**99.99**	**9.998**	**17.375**	**0.449**	**76.145**	**1.745**	**12.192**

5.3 Search and Training Settings

The total time allocated for all 3 challenge datasets was 24 hours. We allocate 8 hours to each dataset, with 4 hours for search and training, each. The search time is further divided into 40-60% for phase 1 and 2, respectively. Moreover, if either of the dimensions of the dataset is greater than 128, or lesser than 8 we restrict the input size to 128x128 and 8x8, respectively.

The search and final training configuration used for challenge are exactly the same as described in Section 4.1.1 for *Plain-layer* settings except that the maximum model size is restricted to 3.2M instead of 4.2M and the final training learning rate is started from 0.01 instead of 0.025.

5.4 Results

As shown in Table 5, our method outperforms competing solutions across all datasets by a noticeable margin, except on GameOfLife (easier) dataset. The training and validation sets are available during the search and training phases, while the final score is determined using held-out test samples. On the relatively easy GameOfLife/Conway dataset, our framework discovers a near-perfect model, achieving 99.99% raw accuracy score. On the more challenging Cryptic and Windspeed datasets, it attains 5% and 4% higher raw accuracy than the respective baselines. For the challenge, Geada et al. (2024) propose computing the score for each dataset using $\max(-10, relative_score)$, where the $relative_score = (test_{acc} - base_{acc}) * s$ and $s = 10/(100 - base_{acc})$. The final score is obtained by summing the individual dataset scores. This scoring scheme prevents a single dataset from dominating the overall score and instead emphasizes consistent generalization across datasets. Using this metric, our solution achieves a total score of **12.192**, outperforming the second- and third-place solutions, which score **10.885** and **10.430**, respectively, thereby highlighting the strong generalization capability of our framework.

5.5 Post Challenge Analysis

In this section, we analyze the framework's robustness to unseen datasets observed during the actual challenge run.

5.5.1 Dataset-Adaptive Search

Figure 3 provides a visual overview of the search process for each challenge dataset. Under a strict time budget of 8 hours per dataset, the proposed framework adapts its behavior by dynamically adjusting the allocated search space, number of iterations, and fidelity budget according to dataset characteristics.

Because the Conway dataset has much smaller spatial dimensions than Cryptic (1×10×10 versus 1×128×128), a larger number of candidates can be evaluated during Phase-1 (20 for Conway compared to 4 for Cryptic). During this phase, both Conway and Windspeed exhibit architecture-level performance saturation, prompting the allocation of additional training budget, whereas Cryptic continues to benefit primarily from architectural growth. In Phase-2, Cryptic, Windspeed, and Conway undergo 4, 3, and 2 elimination rounds, respectively. Although not shown in the figure, challenge logs indicate that Windspeed and

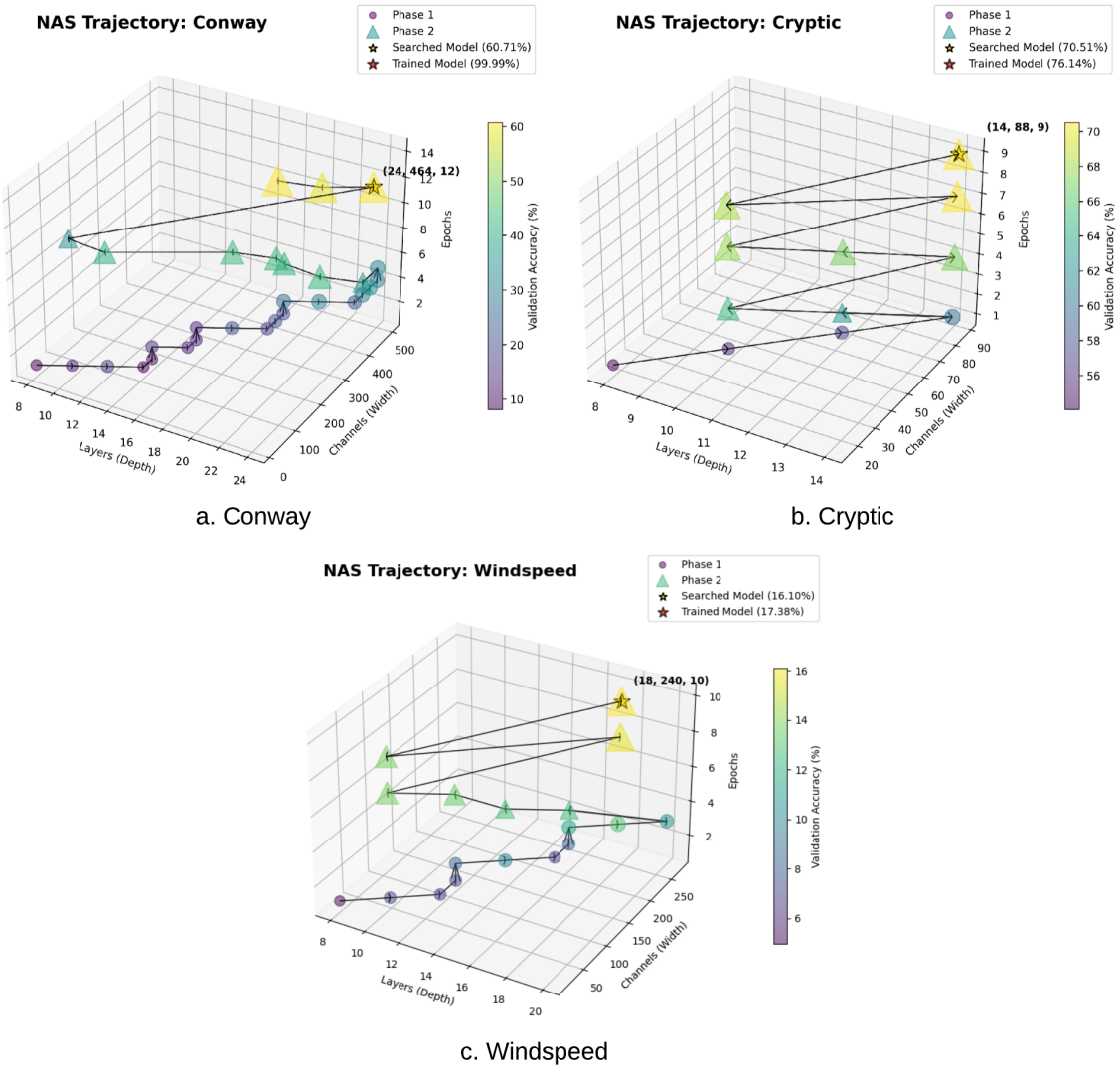

Figure 3: Search Trajectories for Conway, Cryptic and Windspeed datasets during challenge run. Phase 1 evaluation is represented by spheres while Phase 2 using triangles. Moreover, arrows represent the search trajectory and the size of a point represents accuracy. In Phase 1, epochs are increased only when the validation accuracy plateaus, while they are increased for every next candidate evaluation in Phase 2 until a winning model is discovered.

Cryptic reach the model size constraint before exhausting the Phase-1 time budget, demonstrating strong anytime performance. Conway, in contrast, reaches the time limit in both phases but nonetheless achieves the best anytime and final performance.

Throughout the search, competing models vary substantially in size, and smaller architectures are intentionally allocated additional training budget to prevent premature elimination due to insufficient evaluation. This ensures fair comparison across candidates. Moreover, the proposed search space exhibits substantial performance variance despite its compact size: candidate validation accuracies range from 8–60%, 54–70%, and 5–16% for Conway, Cryptic, and Windspeed, respectively. These results confirm that the search space

Table 6: Effect of random seed on candidate rankings. C, W and G represent the first three (1,2,3) candidate architectures evaluated for Cryptic, Windspeed and Conway datasets, respectively. The Avg and High rankings are average and highest accuracy across seeds, respectively. The rankings shift arbitrarily across seeds, averaged accuracy and the highest seed accuracy.

	Cryptic					**Windspeed**					**Conway**				
Rank	Seed			Avg	High	Seed			Avg	High	Seed			Avg	High
	0	1	2			0	1	2			0	1	2		
1	C-3	C-1	C-2	C-3	C-3	W-2	W-1	W-1	W-3	W-1	G-1	G-1	G-2	G-2	G-2
2	C-2	C-3	C-1	C-1	C-2	W-1	W-3	W-2	W-2	W-3	G-2	G-2	G-3	G-1	G-1
3	C-1	C-2	C-3	C-2	C-1	W-3	W-2	W-3	W-1	W-2	G-3	G-3	G-1	G-3	G-3

can generate diverse, dataset-adaptive architectures. During the search, models outperforming the manually designed baseline architectures are discovered for Conway and Windspeed, while performance on Cryptic remains competitive. Subsequent full training further improves accuracy across all datasets, with gains varying according to dataset difficulty.

Overall, the results demonstrate that the proposed search framework efficiently adapts to dataset complexity, maintains fair evaluation across architectures, and delivers strong anytime and final performance under strict time constraints.

5.5.2 Multi-Seed Evaluation

For the challenge run, we evaluate each candidate using five random seeds, and Table 6 illustrates the effect of random seeds on candidate rankings. A key observation is that changing the seed leads to substantial variations in the relative rankings of candidates across all datasets, consistent with the findings of Yang et al. (2019). For instance, for Cryptic dataset, model C-3 and C-1 are ranked 1 and 3rd, respectively for seed 0 but vice versa for seed 2. Similar variations can be observed for Windspeed and Conway datasets. Furthermore, rankings obtained by averaging performance across multiple seeds are generally inconsistent with those derived from the best-performing seed, with the exception of the Conway dataset. While the choice between averaged and best-seed performance remains debatable, these results clearly indicate that relying on a single seed yields unstable rankings, even when evaluation is performed on the full dataset. In contrast, multi-seed evaluation captures performance variability and enables more reliable relative ranking of candidates, making it an important consideration for explainable NAS.

5.5.3 Multi-Fidelity Search

Multi-seed evaluation also reveals performance variance trends similar to those observed on CIFAR datasets (Section 3.2). For instance, the winning model on the Windspeed dataset exhibits performance variances of 4.84%, 2.73%, and 2.86% across different seeds when evaluated at 3, 8, and 10 epochs, respectively. The winning model for the Cryptic dataset shows variances of 9.22%, 1.84%, 1.34%, and 1.02% at 1, 4, 7, and 9 epochs, respectively. Similarly, the model discovered for the Conway dataset exhibits variances of 15.2% and 13.39% at 4 and 12 epochs, respectively. While performance variance is pronounced at lower training fidelities, it decreases substantially as fidelity increases, emphasizing the effectiveness of the proposed multi-fidelity strategy in mitigating variance and stabilizing candidate ranking.

5.5.4 Discovered Architectures

Figure 4 illustrates the architectures discovered for the challenge datasets, each characterized by distinct spatial input resolutions, network depth and width, global pooling sizes, numbers of reduction blocks, and total learnable parameters. Notably, no reduction block is selected for the Conway dataset, which has a relatively small input resolution of 10×10, whereas four and two reduction blocks are assigned to the Cryptic and Windspeed datasets, respectively.

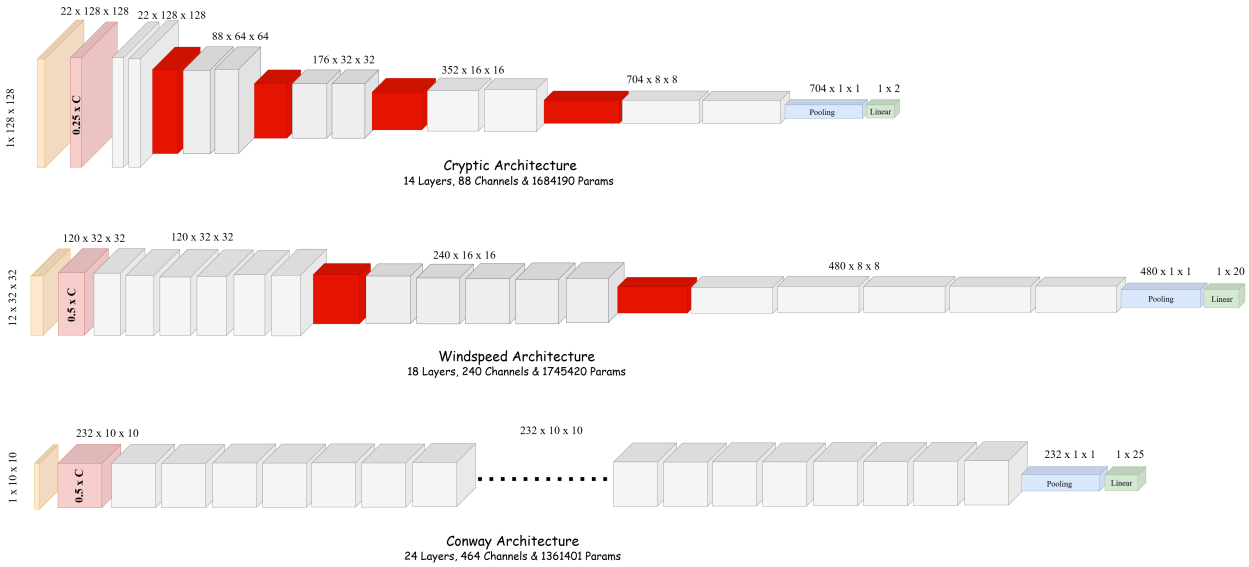

Figure 4: Architectures discovered for the challenge datasets. Each of the three architectures end up with different spatial dimensions, depth, width, global pooling size and number of reduction blocks (shown in red). Block shown in pink is a stem. Note that the stem and first blocks use 0.5×C discovered channels for Conway and Windspeed, and 0.25×C for Cryptic.

5.5.5 Limitations and Future Work

Our work primarily focuses on architecture search, whereas for unseen datasets, other factors such as data augmentation techniques and training hyperparameters (e.g., learning rate and number of training epochs) remain unexplored, presenting a promising direction for future research. Furthermore, the time and GPU memory constraints of the challenge impose limits on the search space, including a parameter budget (e.g., around 3 million) and maximum input image dimensions. As a result, for high-resolution datasets, only a small number of candidate architectures can be evaluated, even though larger models might ultimately prove more effective. Finally, since candidates are trained on the entire dataset during search, both dataset size and image resolution directly impact training time and, consequently, search efficiency. Future work could therefore investigate techniques to accelerate the search process, particularly for large-scale datasets.

6 Conclusion

This work addresses the limitations of existing NAS methods that arise from their heavy reliance on CIFAR, ImageNet, and their derivative benchmarks. Our proposed framework departs from relying on low-fidelity evaluations required to navigate large and complex search spaces, demonstrating that macro-architecture variations alone provide sufficient architectural diversity to adapt to datasets with differing characteristics, and that precise evaluation of a small number of candidates leads to stronger generalization on unseen data. However, since high-fidelity evaluations are computationally expensive and unseen datasets lack well-defined training protocols, we introduce a novel dual-purpose multi-fidelity search algorithm that both accelerates the search process and evaluates candidates under multiple training budgets, thereby improving robustness across datasets. Under strict time constraints, the proposed framework outperforms manually designed baseline architectures across all NAS Unseen-Data Challenge datasets and achieves an overall score of 12.19, outperforming the second- and third-place NAS solutions (10.89 and 10.43), respectively.

References

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*, 2017.

Archit Bansal, Danny Stoll, Maciej Janowski, Arber Zela, and Frank Hutter. Jahs-bench-201: A foundation for research on joint architecture and hyperparameter search. *Advances in Neural Information Processing Systems*, 35:38788–38802, 2022.

Maxim Berman, Leonid Pishchulin, Ning Xu, Matthew B Blaschko, and Gérard Medioni. Aows: Adaptive and optimal network width search with latency constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11217–11226, 2020.

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HylxE1HKwS`.

Hanlin Chen, Ming Lin, Xiuyu Sun, and Hao Li. Nas-bench-zero: A large scale dataset for understanding zero-shot neural architecture search. 2021a.

T Chen, I Goodfellow, J Shlens, et al. Accelerating learning via knowledge transfer. 2016.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive darts: Bridging the optimization gap for nas in the wild. *International Journal of Computer Vision*, 129(3):638–655, 2021b.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Tom Den Ottelander, Arkadiy Dushatskiy, Marco Virgolin, and Peter AN Bosman. Local search is a remarkably strong baseline for neural architecture search. In *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 465–479. Springer, 2021.

Tobias Domhan, Jost Tobias Springenberg, Frank Hutter, et al. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, volume 15, pp. 3460–8, 2015.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1761–1770, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5251–5260, 2021.

Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017.

Linus Ericsson, Elliot J. Crowley, Andrew Stephen McGough, David Towers, Felix Moeller, and Amir Atapour-Abarghouei. Cryptic Dataset. 2025a. doi: 10.7488/ds/8054. URL `https://doi.org/10.7488/ds/8054`. Dataset, University of Edinburgh, School of Engineering.

Linus Ericsson, Elliot J. Crowley, Andrew Stephen McGough, David Towers, Felix Moeller, and Amir Atapour-Abarghouei. Windspeed Dataset. 2025b. doi: 10.7488/ds/8053. URL https://doi.org/10.7488/ds/8053. Dataset, University of Edinburgh, School of Engineering.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pp. 1437–1446. PMLR, 2018.

Rob Geada, David Towers, Matthew Forshaw, Amir Atapour-Abarghouei, and A Stephen McGough. Insights from the use of previously unseen neural architecture search datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22541–22550, 2024.

Hongyi He, Longjun Liu, Haonan Zhang, and Nanning Zheng. Is-darts: Stabilizing darts through precise measurement on candidate importance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12367–12375, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hanzhang Hu, John Langford, Rich Caruana, Eric Horvitz, and Debadeepta Dey. Macro neural architecture search revisited. In *2nd Workshop on Meta-Learning at NeurIPS*, 2018.

Hanzhang Hu, John Langford, Rich Caruana, Saurajit Mukherjee, Eric J Horvitz, and Debadeepta Dey. Efficient forward architecture search. *Advances in Neural Information Processing Systems*, 32, 2019.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial intelligence and statistics*, pp. 240–248. PMLR, 2016.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018.

Haidong Kang, Lianbo Ma, Pengjun Chen, Guo Yu, Xingwei Wang, and Min Huang. Beyond the limits: Overcoming negative correlation of activation-based training-free nas. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 796–805, 2025.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pp. 528–536. PMLR, 2017.

Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. Nas-bench-suite-zero: Accelerating research on zero cost proxies. *Advances in Neural Information Processing Systems*, 35:28037–28051, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pp. 367–377. PMLR, 2020.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18 (185):1–52, 2018.

Marius Lindauer and Frank Hutter. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020.

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

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Vasco Lopes and Luis A Alexandre. Toward less constrained macro-neural architecture search. *IEEE transactions on neural networks and learning systems*, 2023.

Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the genetic and evolutionary computation conference*, pp. 419–427, 2019.

Yash Mehta, Colin White, Arber Zela, Arjun Krishnakumar, Guri Zabergja, Shakiba Moradian, Mahmoud Safari, Kaicheng Yu, and Frank Hutter. Nas-bench-suite: Nas evaluation is (now) surprisingly easy. *arXiv preprint arXiv:2201.13396*, 2022.

Byunggook Na, Jisoo Mok, Hyeokjun Choe, and Sungroh Yoon. Accelerating neural architecture search via proxy data. *arXiv preprint arXiv:2106.04784*, 2021.

Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems*, 34:12265–12277, 2021.

Yameng Peng, Andy Song, Haytham Fayek, Vic Ciesielski, and Xiaojun Chang. Swap-nas: Sample-wise activation patterns for ultra-fast nas. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 31100–31125, 2024. URL `https://proceedings.iclr.cc/paper_files/paper/2024/file/840abfadd04c967feaa2a49aba94a32d-Paper-Conference.pdf`.

Vishak Prasad, Colin White, Sibasis Nayak, Paarth Jain, Aziz Shameem, Prateek Garg, and Ganesh Ramakrishnan. Speeding up nas with adaptive subset selection. In *AutoML 2024 Methods Track*, 2024.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10428–10436, 2020.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning*, pp. 2902–2911. PMLR, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Manas Sahni, Shreya Varshini, Alind Khare, and Alexey Tumanov. Compofa: Compound once-for-all networks for faster multi-platform deployment. *arXiv preprint arXiv:2104.12642*, 2021.

Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. *Advances in neural information processing systems*, 29, 2016.

Jae-hun Shim, Kyeongbo Kong, and Suk-Ju Kang. Core-set sampling for efficient neural architecture search. *arXiv preprint arXiv:2107.06869*, 2021.

Shahid Siddiqui, Christos Kyrkou, and Theocharis Theocharides. True rank guided efficient neural architecture search for end to end low-complexity network discovery. In *International Conference on Computer Analysis of Images and Patterns*, pp. 25–34. Springer, 2023.

Shahid Siddiqui, Christos Kyrkou, and Theocharis Theocharides. Efficient global neural architecture search. *SN Computer Science*, 6(3):282, 2025.

Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference*, pp. 497–504, 2017.

Zihao Sun, Yu Hu, Shun Lu, Longxing Yang, Jilin Mei, Yinhe Han, and Xiaowei Li. Agnas: Attention-guided micro and macro-architecture search. In *International conference on machine learning*, pp. 20777–20789. PMLR, 2022.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

David Towers, Linus Ericsson, Amir Atapour-Abarghouei, Elliot J. Crowley, and Andrew Stephen McGough. NAS Unseen-Data Challenge 2025, 2025a. URL `https://www.nascompetition.com/`. Online benchmark and competition.

David Towers, Linus Ericsson, Amir Atapour-Abarghouei, Elliot J Crowley, and Andrew Stephen McGough. GameOfLife Prediction Dataset. 9 2025b. doi: 10.25405/data.ncl.30000835.v1. URL `https://data.ncl.ac.uk/articles/dataset/GameOfLife_Prediction_Dataset/30000835`.

Xingchen Wan, Binxin Ru, Pedro M Esperança, and Zhenguo Li. On redundancy and diversity in cell-based neural architecture search. *arXiv preprint arXiv:2203.08887*, 2022.

Wenna Wang, Xiuwei Zhang, Hengfei Cui, Hanlin Yin, and Yannnig Zhang. Fp-darts: Fast parallel differentiable neural architecture search for image classification. *Pattern Recognition*, 136:109193, 2023.

Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European conference on computer vision*, pp. 660–676. Springer, 2020.

Colin White, Sam Nolen, and Yash Savani. Exploring the loss landscape in neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 654–664. PMLR, 2021a.

Colin White, Arber Zela, Robin Ru, Yang Liu, and Frank Hutter. How powerful are performance predictors in neural architecture search? *Advances in neural information processing systems*, 34:28454–28469, 2021b.

Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*, 2023.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Shen Yan, Colin White, Yash Savani, and Frank Hutter. Nas-bench-x11 and the power of learning curves. *Advances in Neural information processing systems*, 34:22534–22549, 2021.

Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. Nas evaluation is frustratingly hard. *arXiv preprint arXiv:1912.12522*, 2019.

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

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv preprint arXiv:1807.06906*, 2018.

Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *arXiv preprint arXiv:2001.10422*, 2020.

Xuanyang Zhang, Yonggang Li, Xiangyu Zhang, Yongtao Wang, and Jian Sun. Differentiable architecture search with random features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16060–16069, 2023.

Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020.

Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2423–2432, 2018.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.