IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 July 2024, accepted 15 July 2024, date of publication 29 July 2024, date of current version 18 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3434743

RESEARCH ARTICLE

D2NAS: Efficient Neural Architecture Search With Performance Improvement and Model Size Reduction for Diverse Tasks

JUNGEUN LEE, (Member, IEEE), SEUNGYUB HAN, (Student Member, IEEE), AND JUNGWOO LEE, (Senior Member, IEEE)

Cognitive Machine Learning Laboratory, Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea

Corresponding author: Jungwoo Lee (junglee@snu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) under Grant 2021R1A2C2014504 (30%); in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT), Institute of New Media and Communications (INMAC) under Grant 2021-0-00106 (70%); and in part by the Brain Korea 21 FOUR Program of the Education and Research Program for Future ICT Pioneers (BK21 FOUR), Seoul National University in 2024.

ABSTRACT Neural Architecture Search (NAS) has proven valuable in many applications such as computer vision. However, its true potential is unveiled when applied to less-explored domains. In this work, we study NAS for addressing problems in diverse fields where we expect to apply deep neural networks in the real world domains. We introduce D2NAS, Differential and Diverse NAS, leveraging techniques such as Differentiable ARchiTecture Search (DARTS) and Diverse-task Architecture SearcH (DASH) for architecture discovery. Our approach outperforms existing models, including Wide ResNet (WRN) and DASH, when evaluated on NAS-Bench-360 tasks, which include 10 numbers of diverse tasks for 1D, 2D, 2D Dense (tightly interconnected data) tasks. Compared to DASH, D2NAS reduces average error rates by 12.2%, while achieving an 85.1% reduction in average parameters (up to 97.3%) and a 91.3% reduction in Floating Point Operations (FLOPs, up to 99.3%). Therefore, D2NAS enables the creation of lightweight architectures that exhibit superior performance across various tasks, extending its applicability beyond computer vision to include mobile applications.

INDEX TERMS DARTS, diverse task architecture search, gradient-based NAS, mutual information, neural architecture search (NAS).

I. INTRODUCTION

In machine learning, there has been enormous demand for automating the model architecture development process, leading to the emergence of automated machine learning (AutoML). AutoML aims to streamline and democratize the intricate task of designing and fine-tuning machine learning models, enabling practitioners across various domains to harness the potential of advanced algorithms without requiring expertise in the intricacies of model architecture. Among the myriad approaches within AutoML, neural architecture search (NAS) has emerged as a pivotal tool to automate model development, aiming to streamline the design of deep neural networks. This process ensures comparable performance to handcrafted architectures while minimizing human effort spent on intricate architecture tuning. As machine learning applications become diverse, NAS has advanced significantly in search space design [1], [2], efficiency [3], and algorithms [4], [5], [6].

Among NAS methods, differentiable architecture search (DARTS) [1] has been drawing much attention lately. DARTS introduces differentiability into the architecture search process, enabling the use of gradient-based optimization methods and transforming discrete architecture search into a continuous optimization problem. The implications of DARTS extend beyond efficiency gains, offering a more scalable and versatile framework for exploring neural network architectures. However, despite the efforts to improve

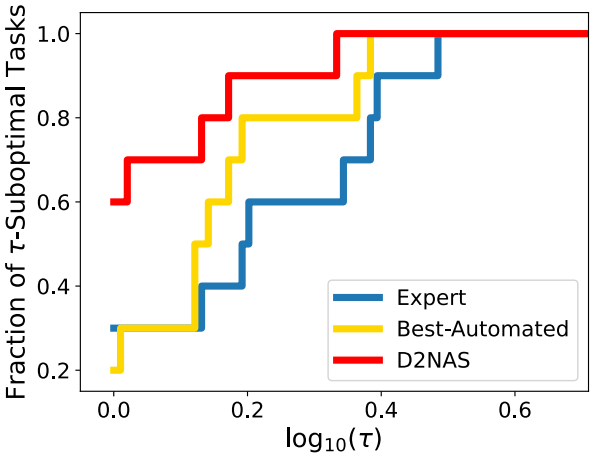
The associate editor coordinating the review of this manuscript and approving it for publication was Jolanta Mizera-Pietraszko.

search speed and achieve cutting-edge performance on vision datasets such as CIFAR and ImageNet, NAS methods, including DARTS, often exhibit suboptimal performance when applied to tasks beyond their original scope. Despite the dominance of NAS methods in computer vision benchmarks, NAS still has limited impact in under-explored or under-resourced domains where architecture design patterns are less understood. An analysis of NAS-Bench-360 [7], a recent benchmark emphasizing task diversity in NAS evaluation, revealed a significant performance gap between NAS-discovered models (e.g., DARTS and DenseNAS [8]) and hand-crafted expert architectures across various applications.

The recent success in enhancing AutoML generalizability, such as AutoML-Zero [9] and XD-operations [10], proposes relaxing inductive biases in standard search spaces. However, practical deployment challenges hinder AutoML-Zero, and XD-operations prove prohibitively expensive even for straightforward problems such as CIFAR-100. In this context, diverse-task architecture search (DASH) [11] has emerged, offering expressivity for high accuracy across domains while preserving the speed and efficiency of discrete architecture search. DASH holds the potential to address computational constraints inherent in traditional differentiable NAS methods, producing high-quality models that rival or surpass manually designed networks in various tasks. However, DASH's search efficiency and performance of the final architecture may be influenced by the choice of the backbone architecture, potentially leading to longer search times.

In this work, we introduce D2NAS, an architecture search method that outperforms DASH in diverse tasks while maintaining the expressivity of AutoML-Zero and XD-operation. Since DASH uses Wide ResNet (WRN) [12], [13] as a backbone for kernel pattern optimization, the final architecture cannot perform beyond the range that WRN can express. In order to better solve diverse problems, D2NAS uses cell-based architecture search to initially find sub-optimal architecture and then use it as a backbone for kernel pattern optimization instead of WRN.

We evaluated D2NAS on 10 datasets from NAS-Bench-360 [7], including tasks such as PDE solving, protein folding, and disease prediction, considering both accuracy and performance aspects. Across all 10 tasks, D2NAS has an error rate that is 12.2% lower than DASH, with an average reduction of 85.1% in parameters and 91.3% in FLOPs. Consequently, D2NAS, which utilizes task-specific backbones discovered through architecture search, generates models with better overall performance than both hand-crafted architectures and models obtained through AutoML, as can be seen in Fig. 1. On a task-specific level, D2NAS surpasses hand-crafted architectures in 8 out of 10 tasks and outperforms existing automated models in 9 out of 10 tasks.



(a)

Method	Top-1 Error (%)	Params (M)
MobileNetV2	31.92	2.35
DASH	24.37	2.77
D2NAS	**22.95**	**1.60**

(b)

FIGURE 1. (a) Comparison of the aggregate performance of the task-wise automated methods, expert models, and D2NAS on ten diverse tasks via performance profiles [14]. The y-axis is fraction of tasks on which a method is within $\log\tau$-factor of the best. Larger values are better. (b) Comparison of the number of parameters tested on CIFAR-100. D2NAS creates a network with fewer parameters compared to other methods or networks.

II. RELATED WORKS

A. NEURAL ARCHITECTURE SEARCH (NAS)

Neural architecture search (NAS) has emerged as a prominent approach for automating the creation of task-specific architectures. The workflow of NAS is divided into three stages: search space, search strategy, and performance estimation. Various methods are employed in the search strategy stage. Many popular search methods have adopted reinforcement learning (RL) [4], evolutionary algorithms (EA) [5], [6], or gradient descent algorithms [1], [15], [16], [17] to find the best-performing candidate architecture. Among these approaches, gradient descent-based architecture search methods consistently outperform alternative strategies in terms of efficiency, simplicity, computational cost, and validation error.

1) DIFFERENTIABLE ARCHITECTURE SEARCH (DARTS)

DARTS [1] is one of the most representative differentiable NAS methods. It employs the technique of continuous relaxation to transform the discrete selection of neural operations into a well-defined optimization task, all within a predefined candidate operation set. This process is made end-to-end differentiable through the use of a softmax function, allowing for optimization with standard gradient-based methods.

The first NAS paradigm we consider is cell-based NAS [4]. In this context, a "cell" is formally defined as a

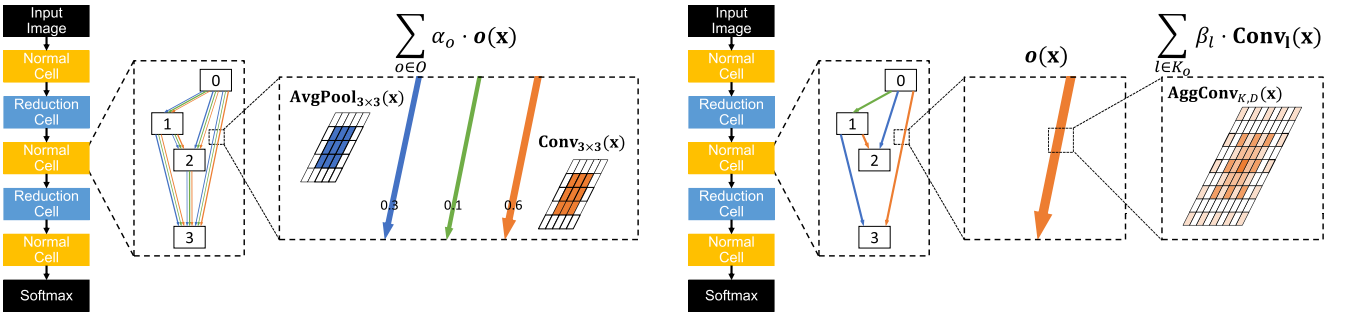


FIGURE 2. The 2-step search process of the proposed hierarchical architecture search. Since the operations at each node are fixed in the first step and the convolutional operations are replaced by the summed convolution in the second step, the search space is not large, allowing for efficient architecture discovery.

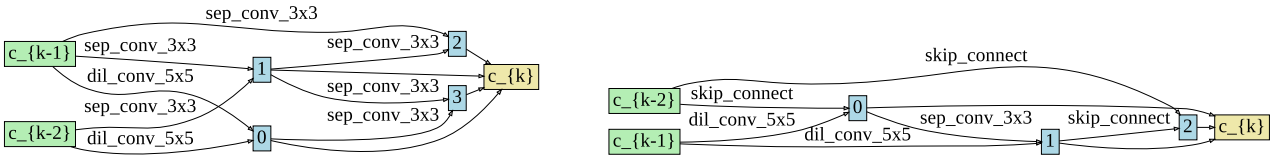


FIGURE 3. (a) Example of a normal cell with 4 nodes learned on 2D tasks. (b) Example of a normal cell with 3 nodes learned on 1D tasks.

directed acyclic graph (DAG) comprising several nodes. The fundamental objective of these methods is to search for a genotype, essentially a cell that encapsulates neural operations. Subsequently, during the evaluation phase, an architecture is constructed by replicating the searched cell and stacking them together.

Among the various search spaces available, DARTS search space stands as the most prevalent choice. It involves the assignment of one of eight operations to six edges in two distinct types of cells. The ''normal'' cells primarily preserve the input's shape, whereas ''reduction'' cells serve to downsample it. P-DARTS [16] enhances both the generalizability and accuracy of DARTS through a progressive approach that incrementally deepens the network and refines the set of candidate operations using a mixed operation weight.

B. ARCHITECTURE SEARCH FOR DIVERSE TASKS

While NAS methods have traditionally targeted computer vision applications, recent research has explored their applicability in broader domains such as automated machine learning (AutoML). Three notable approaches in this direction are AutoML-Zero [9], XD-operations [10], and DASH [11].

1) XD-OPERATIONS

XD-operations introduce a unique approach by utilizing convolutional theorems in architecture discovery. They expand the search space by expressing convolutions as formulas involving discrete Fourier transforms and generating kaleidoscope matrices [18] during the search process. However, XD-operations are computationally intensive and may produce less efficient architectures due to the absence of a discretization step.

2) DASH

DASH employs various techniques, including simplifications of the discrete Fourier transform (DFT) similar to those used in XD-operations, to efficiently search for convolutional patterns and integrate them into existing backbone architectures. DASH is promising in overcoming computational constraints while achieving competitive accuracy levels. However, the efficiency of DASH's search process and the performance of the final architecture can be influenced by the choice of the backbone architecture.

3) NAS-BENCH-360

Benchmark datasets and evaluation protocols are crucial for NAS research. Common benchmarks [19], [20] such as CIFAR-10 and ImageNet are primarily focused on computer vision, limiting their suitability for evaluating diverse tasks. To address this limitation, NAS-Bench-360 [7] was introduced as a comprehensive benchmark that explores the potential of NAS in various domains beyond computer vision. In our experiments, we utilize NAS-Bench-360 as a benchmark and compare its performance to a state-of-the-art AutoML baseline (details are in Table 1).

III. UNIFYING SEARCH SPACES OF CELL AND OPERATOR

In this section, we first explain the problem of a large search space required to simultaneously find two learnable optimal

TABLE 1. Information about tasks in NAS-Bench-360. NAS-Bench-360 is a benchmark consisting of a setup consisting of 10 suitable datasets representing different application areas, dataset sizes, problem dimensions, and learning objectives.

Type	Task	# Data	# Class	Metric	Expert arch.	Explaination	License
2D	CIFAR100	60K	100	error(%)	DenseNet-BC [21]	Standard Image Classification. Classify natural images.	CC BY 4.0
2D	Spherical	60K	100	error(%)	S2CN [22]	Classifying Spherically Projected CIFAR-100 Images	CC BY-SA
2D	NinaPro	3956	18	error(%)	Attention Model [23]	Classifying Electromyography Signals. Classify sEMG signals corresponding to hand gestures	CC BY-ND
2D	FSD50K	51K	200 (multi-label)	1 - mAP	VGG [24]	Labeling Sound Events. Classify sound events in log-mel spectrograms.	CC BY 4.0
2D Dense	Darcy Flow	1100	-	Relative l_2	FNO [25]	Solving Partial Differential Equations (PDEs). Predict the final state of a fluid from its initial conditions.	MIT
2D Dense	PSICOV	3606	-	MAE_8	DEEPCON [26]	Protein Distance Predict. Predict pairwise distances between residuals from 2D protein sequence features.	GPL
2D Dense	Cosmic	5250	-	1 - AUROC	deepCR-mask [27]	Identifying Cosmic Ray Contamination. Predict probabilistic maps to identify cosmic rays in telescope images	Open License
1D	ECG	330K	4	1 - F1	ResNet-1D [28]	Detecting Heart Disease. Detect atrial cardiac disease from an ECG recording	ODC-BY 1.0
1D	Satellite	1M	24	error(%)	ROCKET [29]	Satellite Image Time Series Analysis. Classify satellite image pixels' time series into 24 land cover types	GPL 3.0
1D	DeepSEA	250K	36 (multi-label)	1 - AUROC	DeepSEA [30]	Predicting Functional Effects From Genetic Sequences. Predict chromatin states and binding states of RNA sequences.	CC BY 4.0

hyperparameters: cell and kernel. We then discuss how a hierarchical search aids in finding architectures for diverse tasks. By analyzing the information bottleneck property of each architecture, we propose a search hierarchy. This begins with learning promising local approximate cell architectures, and then finding the optimal kernel of edges for each task.

A. PROBLEM OF LARGE SEARCH SPACE

The potential search space for cells and operation patterns (kernel patterns) for multiple diverse tasks, aimed at finding the optimal architecture from scratch, is much larger than the search space for DARTS and the multi-scale convolution cell search space with a fixed backbone for DASH. DARTS starts with the cell-based architecture search, as shown in Fig. 2-1. As described in Fig. 3, we initialize the numbers of nodes N and edge E as $|N| = 7$, $|E| = 14$ for 2D tasks as originally used in DARTS and $|N| = 5$ and $|E| = 9$ for 1D tasks. Since DARTS does not demonstrate the result of 1D tasks, we design the above numbers for 1D task by empirically searching on these unknown hyperparameters. The possible choice of operation is given as 8 kernels, $\mathcal{S}_{DARTS} = (\texttt{Conv}_3, \texttt{Conv}_5, \texttt{DilatedConv}_{3,2}, \texttt{DilatedConv}_{5,2}, \texttt{AvgPool}, \texttt{MaxPool}, \texttt{identity}, \texttt{zero})$. In the case of DASH, the fixed backbone is based on Wide ResNet, we can only modify kernel pattern $S_{AggConv_{K,D}}$ which denotes the aggregated convolution filter set that is defined in DASH [11], and covers $|K||D|$ convolutions, such as $\texttt{Conv}_K$, $\texttt{DilatedConv}_{K,D}$.

We now define a cell as a directed acyclic graph, which consists of nodes for latent representations and edges for an energy-based mixed-operation $\bar{o}^{i,j}$ from node i to node j:

$$\bar{o}^{i,j}(x) = \sum_{o \in \mathcal{S}} \frac{\exp(\alpha_o^{i,j})}{\sum_{o' \in \mathcal{S}} \exp(\alpha_{o'}^{i,j})} o(x),$$

where α_o is the weight of a given operator o [1], [11].

To search the architecture of diverse tasks from scratch, we define the search space $\mathcal{S}$ as the union of four different operation sets: $(S_{AggConv_{K,D}}, \texttt{AvgPool}, \texttt{MaxPool}\ \texttt{identity}, \texttt{zero})$. As the proposed search space is proportional to $O(|E|(|K||D| + |T|))$, where $|E|$, $|K|$, $|D|$, and $|T|$ are the desired set of the number of edges in a cell, kernel sizes, dilation rate, and the remaining operations, respectively, searching on our extended operation search space is computationally expensive with the differential architecture search method. For example, the number of possible combinations of 2-layered cell architectures, which consist of one normal cell and one reduction cell for 2D tasks (Fig. 3-(a)) can be computed as $2|E|(|K||D| + |T|) = 2 \cdot 14(4 \cdot 4 + 4) = 560$, which is 2.5 times larger than DARTS ($2|E||\mathcal{S}_{DARTS}| = 2 \cdot 14 \cdot 8 = 224$) and 2.7 times larger than DASH ($|E||K||D| = 13 \cdot 4 \cdot 4 = 208$). To improve upon the vanilla differential architecture search, we propose the following techniques which build up to the extension of DARTS and DASH.

B. HIERARCHICAL STRUCTURE FOR OPERATIONS

Since the extended set of possible operations is too large, we describe a hierarchical structure for candidate operations to reduce search wall-clock time. As the differential architecture search for building block is based on a Boltzmann mixed operation, which is also called *softmax* version of mixed operators, we can generate a probabilistic graphical model for our interested operator space $\mathcal{S}$.

To define the proposed hierarchical search space rigorously, we introduce two sets: the operation type set $\mathcal{T}$=$\{\texttt{zero}, \texttt{identity}, \texttt{AvgPool}, S_{AggConv_{K,D}}\}$ and the child set $\mathcal{C}$ of the kernel pattern $S_{AggConv_{K,D}}$, $\mathcal{K} \subset K \times D = \{(k, d) | k \in K \text{ and } d \in D\}$. Note that we can exclude $\texttt{MaxPool}$ for the smaller search space by observing that $\texttt{MaxPool}$ is not empirically critical. Now, we formally define our extended search space $\mathcal{S}_{\text{D2NAS}}$ as $\mathcal{T} \times \mathcal{K} = \{(t, s_t) | t \in \mathcal{T}, s_t \in \mathcal{K}\}$. Note that the child kernel pattern sets for the operation types $\{\texttt{zero}, \texttt{identity}, \texttt{AvgPool}\}$ are $\{\texttt{default}\}$, which have no specific shape (only provide default setting). Then, we can rewrite the probability of the operations $o \in \mathcal{S}_{\text{D2NAS}}$ for

edge (i, j) as

$$P^{(i,j)}(\texttt{type} = t, \texttt{pattern} = s) = P(t|(i,j)) \cdot P(s|t,(i,j)).$$

By applying the *soft*-version of mixed-operations, we provide the mixture version of operation $\bar{o}^{i,j}$ as

$$\sum_{t\in\mathcal{O}} \frac{\exp(\alpha_t^{i,j})}{\sum_{t'\in\mathcal{O}}\exp(\alpha_{t'}^{i,j})} \cdot \left(\sum_{s\in\mathcal{K}_o} \frac{\exp(\beta_s^{i,j})}{\sum_{s'\in\mathcal{K}_o}\exp(\beta_{s'}^{i,j})} \right) o^{i,j}, \quad (1)$$

where α_t is the weight of a given operation type t and β_s is the weight of a given kernel pattern $s = (k, d)$. In terms of gradient based bi-level optimization problem for differentiable architecture search, we can write our proposed problem as

$$\begin{aligned} &\min_{\alpha,\beta} \mathcal{L}_{val}(w^*(\alpha,\beta),\alpha,\beta) \\ &\text{s. t.} \quad w^*(\alpha,\beta) = \arg\min_{w} \mathcal{L}_{train}(w,\alpha,\beta). \end{aligned} \quad (2)$$

As noted earlier, to find the global optimal architecture in the extended search space $\mathcal{S}_{\text{D2NAS}} = \cup_{t\in\mathcal{T}}\{(t, s_t)\}$ is challenging due to the high dimensional problems with a large sample size.

C. INFORMATION BOTTLENECK ANALYSIS ON CELL AND KERNEL

In this section, we discuss the information bottleneck properties among input data X, the lth hidden layer H_l and prediction logit $\hat{Y}$ with given operations $\mathcal{S}$. The information bottleneck method for each hidden layer l introduces the following problem with the lagrange multiplier λ as

$$\max_{w^*(\alpha,\beta),\alpha,\beta} I(H_l, \hat{Y}; w^*(\alpha,\beta)) - \lambda I(H_l, X; w^*(\alpha,\beta)) \quad (3)$$

to learn the optimal hidden layer H_l which is both maximally expressive about $\hat{Y}$ and maximally compressive about X. Recent studies [31], [32] have shown that the feature compression effect by lower $I(H_l, X; w^*(\alpha,\beta))$ has high correlation with better generalization performance.

We leverage the above result to measure the feature compression property evolving through the hidden layers and compare the performance of each architecture. Further, we empirically demonstrate that the operation type $\mathcal{T}$ is more principal component of varying $I(H_l, X; w^*(\alpha,\beta))$. In Fig. 5, we present the mutual information estimates of the first four layers of D2NAS by varing cell architecture and kernel pattern, respectively. As we note in Fig. 4, the mutual information estimates diminishes over time, so we provide the values of the first four layers for better visualization. We note that the differential architecture search updates the values of operation weight and parameters simultaneously, making the proposed problem of Eq. (2) is a bi-level optimization problem. For better efficiency, we need to optimize the high varying variables first. Therefore, our search strategy is to make the cell search priority higher and search the kernel pattern based on the cell architecture obtained from the earlier stage.

Algorithm 1 Differentiable and Diverse Neural Architecture Search (D2NAS)

1: Initialize: architecture parameters α_1, model weights θ_1.
2: Split dataset into training and validation sets.
3: **while** not converged **do**
4: Sample a mini-batch of training data.
5: Compute the gradient $\nabla_{\alpha_1}\mathcal{L}_{\text{val}}(\theta_1,\alpha_1)$.
6: Update architecture parameters $\alpha_1 \leftarrow \alpha_1 - \eta\nabla_{\alpha_1}\mathcal{L}_{\text{val}}(\theta_1,\alpha_1)$.
7: Compute the gradient $\nabla_{\theta_1}\mathcal{L}_{\text{train}}(\theta_1,\alpha_1)$.
8: Update model weights $\theta_1 \leftarrow \theta_1 - \eta\nabla_{\theta_1}\mathcal{L}_{\text{train}}(\theta_1,\alpha_1)$.
9: **end while**
10: Derive the semifinal architecture by selecting $\arg\max_{o\in O}\alpha_{1o}^{(i,j)}$ for each edge (i, j).
11: Prepare the set of kernel sizes K and the set of dilation rate D.
12: Replace each **Conv** layer in the semifinal architecture with the mixed operation $\textbf{AggConv}_{K,D}$
13: Initialize: architecture parameters α_2, model weights θ_2.
14: **while** not converged **do**
15: Sample a mini-batch of training data.
16: Compute the gradient $\nabla_{\alpha_2}\mathcal{L}_{\text{train}}(\theta_2,\alpha_2)$ and $\nabla_{\theta_2}\mathcal{L}_{\text{train}}(\theta_2,\alpha_2)$
17: Update architecture parameters $\alpha_2 \leftarrow \alpha_2 - \eta\nabla_{\alpha_2}\mathcal{L}_{\text{train}}(\theta_2,\alpha_2)$.
18: Update model weights $\theta_2 \leftarrow \theta_2 - \eta\nabla_{\theta_2}\mathcal{L}_{\text{train}}(\theta_2,\alpha_2)$.
19: **end while**
20: Derive the final architecture by selecting $\arg\max_{k\in K,d\in D}\alpha_{2k,d}$ for each **AggConv** layer.
21: Tune hyperparameters on a subset of the training data.
22: Retrain the discretized model with all training data.

Fig. 4 shows the important relationship between the generalization performance and information bottleneck. The architectures with higher prediction performance in Table 2 compress the information of the last hidden layer feature better. The mutual information $I(X, H_l)$ decreases more as the network evolves the deeper layer whether the networks are trained or not. It implies that the hidden layers compress the information in the input features to help the final fully-connected layer to predict values more easily. All layer outputs of learned cells or blocks at the front have relatively large mutual information with the input data compared to the internal layers, because the front layers try to capture the input feature to predict result successfully by maximizing $I(H_l, \hat{Y}; w^*(\alpha,\beta))$ which increases the correlation among X, H_l, and $\hat{Y}$ in Eq. (3). Although the proposed method D2NAS has fewer parameters than baselines, the repetitive proposed cell architecture effectively compress the mutual information $I(X, H_l)$. It implies that the backbone operator type is more critical than the kernel pattern when we consider the comparison between Wide ResNet and DASH in CIFAR100.

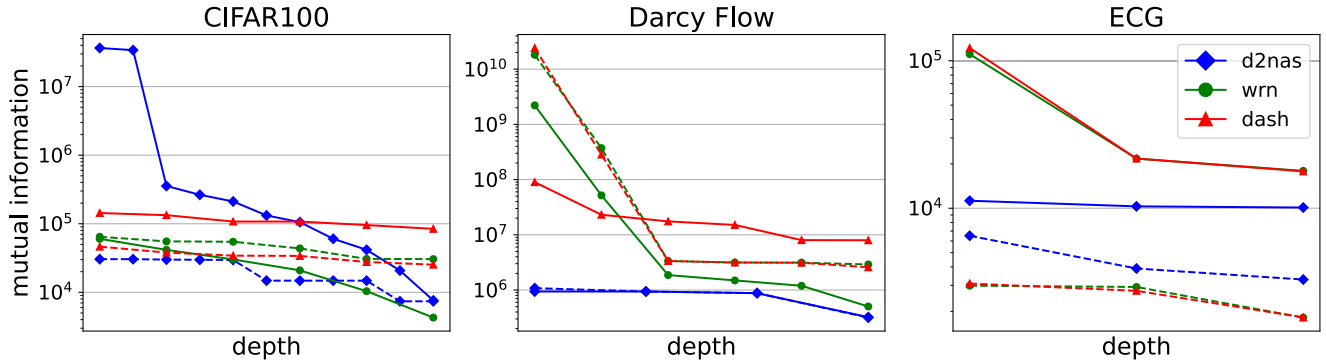


FIGURE 4. Results of $I(X, H_l)$ for three types of benchmarks, CIFAR100, DarcyFlow, ECG. Dashed line denotes the initialized model parameters for all architecture search algorithms. The X axis is normalized to represent the relative depth for each network, and the dots indicate the result of each hidden layer.

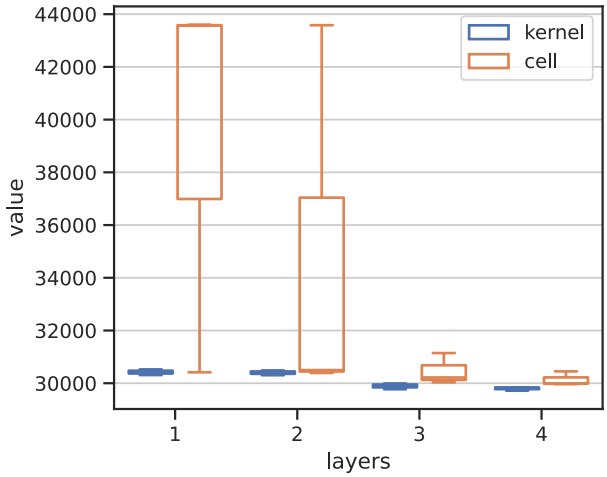


FIGURE 5. Variances of mutual information estimates of the first four layers of D2NAS for CIFAR100 for 10 random choices of operation type $\mathcal{T}$ and kernel pattern $\mathcal{K}$.

IV. PROPOSED METHOD

In this section, we describe the details of D2NAS in Algorithm 1, a fast and stable search algorithm designed to generate a lightweight and high-performance network architecture for various domains.

A. HIERARCHICAL ARCHITECTURE SEARCH: D2NAS

As discussed in Section III-C, the weight parameters α and β have different amounts of effect on feature compression performance, we propose a novel hierarchical differentiable architecture search algorithm that consists of the two-level hierarchy (see Fig. 2). This approach is significant for the architecture search across diverse tasks because the full combination of operator types and shapes is too large for the existing NAS methods, where select a small subset heuristically to reduce the search time. To tackle this problem, we first optimize on the operator type set $\mathcal{T}$, which has the lower varying feature compression performance, using initial choices of kernel pattern for $S_{AggConv_{K,D}}$. Note that these initial choices are only used for finding the operator types of each node (i, j) on the first stage. After the first stage for the semi-final architecture with searched operator type t in line 10 in Algorithm 1 by discretizing cell architecture as explained in the *retrain* part in Section IV-B, we then optimize on the kernel pattern $s_t \in \mathcal{K}$ for the cell architecture with the above searched operator types t. The detailed pseudo-code is provided in Algorithm 1.

B. FULL PIPELINE

For the proposed method, D2NAS, we evaluate the possible combinations for searching as follows. In the first stage, we consider $2|E_1||T| = 2 \cdot 14 \cdot 5 = 140$ combinations and apply the intermediate results with the reduced numbers of edge in the second stage. Then, we have the possible combinations for the second stage as $2|E_2||K||D|P_{conv} = 2 \cdot 8 \cdot 4 \cdot 4 \cdot P_{conv} = 256 \cdot P_{conv}$, where E_2 and P_{conv} denote the reduced number of edges and the selected proportion of $\texttt{Conv}_3$ and $\texttt{DilatedConv}_{5,2}$, which are the convolution operators with arbitrary patterns for selecting operator type in the first stage. The range of combinations for the second stage lies between 0 and 256 considering P_{conv} varies within $[0, 1]$. For the total search complexity, we aggregate the complexities of the two stages: $O(2|E_1||T|) + O(2|E_2||K||D|P_{conv})$. This yields a simplified range of $140 + 0 = 140$ to $140 + 256 = 396$. We note that the achievable complexity is remarkably lower than 560 in Section III-A even if we have 396 for the number of choices.

1) CELL-BASED ARCHITECTURE SEARCH

Before initiating cell-based architecture search, it is necessary to define the cell, including the number of nodes and edges, and specifying the types of operations. Subsequently, we determine the number of layers, denoted as L, to ensure an appropriate combination of normal cells and reduction cells. For 2D dense tasks (details are in Table 1), the architecture exclusively comprises normal cells, as these tasks necessitate the preservation of the input data's shape.

TABLE 2. Comparison of error rates (Lower is better) on NAS-Bench-360 tasks. D2NAS(common) utilized the same hyperparameter set for each group, categorized by dataset type such as 2D, 2D Dense and 1D, while D2NAS(custom) employed a distinct hyperparameter set for each individual dataset. Scores of D2NAS are averaged over three trials. Scores of the baselines are from Tu et al. [7] and Shen et al. [11]. D2NAS surpasses expert architectures in 8 out of 10 tasks and outperforms automated models in 9 out of 10 tasks.

Method	2D				2D Dense			1D		
	CIFAR100	Spherical	NinaPro	FSD50K	Darcy Flow	PSICOV	Cosmic	ECG	Satellite	DeepSEA
	error(%)	error(%)	error(%)	1 - mAP	Relative l_2	MAE_8	1 - AUROC	1 - F1	error(%)	1 - AUROC
Expert	**19.39±0.2**	67.41±0.76	8.73±0.9	0.62±0.004	0.008±0.001	3.35±0.14	**0.13±0.001**	0.28±0	19.8±0	0.3±0.24
WRN	23.35±0.05	85.77±0.71	6.78±0.26	0.92±0.001	0.073±0.001	3.84±0.053	0.24±0.015	0.43±0.01	15.49±0.03	0.4±0.001
TCN	-	-	-	-	-	-	-	0.57±0.005	16.21±0.05	0.44±0.001
Perceiver IO	70.04±0.44	82.57±0.19	22.22±0.62	0.72±0.002	0.24±0.01	8.06±0.06	0.48±0.01	0.66±0.01	15.93±0.08	0.38±0.004
WRN-ASHA	23.39±0.01	75.46±0.4	7.34±0.76	0.91±0.03	0.066±0	3.84±0.05	0.25±0.021	0.43±0.01	15.84±0.52	0.41±0.002
DARTS-GAEA	24.02±1.92	**48.23±2.87**	17.67±1.39	0.94±0.02	0.026±0.001	2.94±0.13	0.22±0.035	0.34±0.01	12.51±0.24	0.36±0.02
DenseNAS	25.98±0.38	72.99±0.95	10.17±1.31	0.64±0.002	0.1±0.01	3.84±0.15	0.38±0.038	0.4±0.01	13.81±0.24	0.4±0.02
Auto-DL	-	-	-	-	0.049±0.005	6.73±0.73	0.49±0.004	-	-	-
AMBER	-	-	-	-	-	-	-	0.67±0.015	12.97±0.07	0.68±0.01
DASH	24.37±0.81	71.28±0.68	6.6±0.33	0.6±0.008	0.0079±0.002	3.3±0.16	0.19±0.02	0.32±0.007	12.28±0.5	0.28±0.013
D2NAS(common)	25.41±0.57	57.05±2.95	6.21±0.15	0.55±0.012	**0.0055±0.0006**	**2.83±0.12**	0.24±0.023	**0.28±0.001**	**12.21±0.1**	**0.25±0.005**
D2NAS(custom)	22.95±0.17	54.73±4.07	**5.89±0.18**	**0.51±0.009**	-	-	0.18±0.013	0.29±0.007	-	-

TABLE 3. Comparison of the number of parameters and FLOPs. D2NAS(custom) is superior to the original architecture in terms of the computational cost. D2NAS significantly reduces the number of parameters and FLOPs on average by 85.1% and 91.3%, respectively.

Method	CIFAR100		Spherical		NinaPro		FSD50K		Darcy Flow	
	Params(M)	FLOPs(G)	Params(M)	FLOPs(G)	Params(M)	FLOPs(G)	Params(M)	FLOPs(T)	Params(M)	FLOPs(G)
DASH	2.77	32.54	10.35	509.49	9.78	179.39	23.58	5.34	3.71	268.25
D2NAS(common)	0.38	1.58	0.35	21.51	0.24	7.40	0.48	0.04	0.14	10.13
D2NAS(custom)	1.60	3.98	0.39	23.09	0.27	8.17	0.72	0.05	-	-

Method	PSICOV		Cosmic		ECG		Satellite		DeepSEA	
	Params(M)	FLOPs(G)	Params(M)	FLOPs(G)	Params(M)	FLOPs(G)	Params(M)	FLOPs(T)	Params(M)	FLOPs(G)
DASH	4.22	553.17	3.10	203.13	3.16	3.32	1.40	28.14	1.59	476.25
D2NAS(common)	0.22	29.29	0.10	6.70	205.08	127.14	0.30	2.09	0.23	34.30
D2NAS(custom)	-	-	0.07	4.67	2.46	1.62	-	-	-	-

However, in all other scenarios, we identify the layer with an index i corresponding to the element in the following set, which then becomes a reduction cell: $i \in \left[\frac{L}{3}, \max\left(3, \frac{2 \cdot L}{3}\right)\right]$.

In this work, for the 2D task, a single cell is composed of 2 inputs and 4 nodes, following the approach established in DARTS. However, in the case of the 1D task, we reduced the number of nodes to 3 (see Fig. 3). While the edge connection method remains consistent with DARTS, we modified the set of 8 operations to either 5 or 4 operations based on the necessity of including pooling operations to reduce the time required for cell searching.

2) KERNEL PATTERNS SEARCH

In this phase, the network obtained from the preceding cell-based architecture search step serves as the backbone. During this stage, DASH conducts joint optimization of both model weights and architecture parameters through direct gradient descent, guided by the loss function specific to the target task to find improved kernel patterns.

3) HYPERPARAMETER TUNING

Incorporating a hyperparameter tuning stage before retraining allows us to methodically identify the optimal values for critical parameters, including learning rate, momentum, weight decay, and dropout rate. This is achieved through a grid search process.

4) RETRAIN

In the last step, we retrain the model, which has undergone two stages of discretization, using the optimal hyperparameters along with the complete training dataset. Discretization, defined in DARTS, is the process to select the most probable operator from the soft-version of mixed operations in Eq. (1). The resulting model demonstrates enhanced efficiency when compared to the original supernet.

V. EVALUATION

In this work, we conducted experiments using 10 datasets from NAS-Bench-360 [7]. Instead of employing conventional backbone networks such as Wide ResNet (WRN), we used networks discovered by the progressive architecture discovery method used in P-DARTS [16] as the backbone. Subsequently, we evaluated the performance and GPU latency of the architecture after enhancing the kernel pattern with DASH [11] and reported the aggregate performance results via performance profiles [14] to compare performance of D2NAS and other methods across multiple tasks.

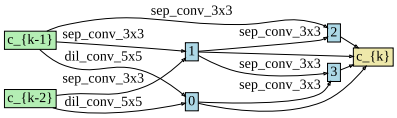
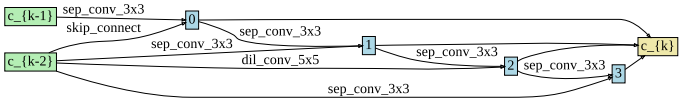
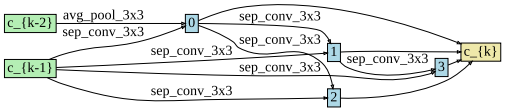
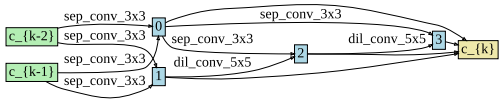
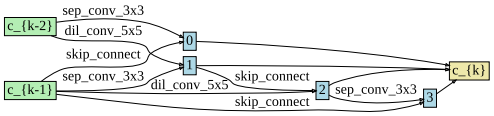
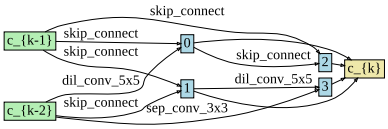
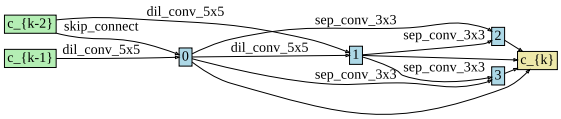
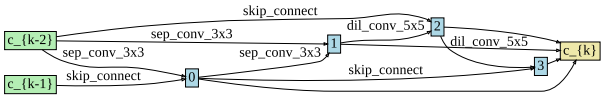
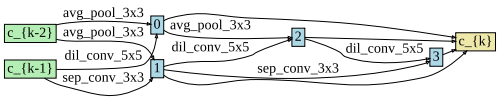
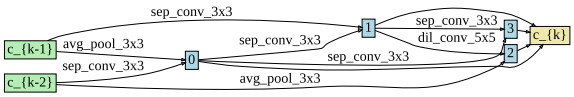
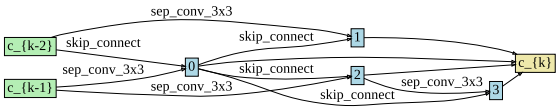
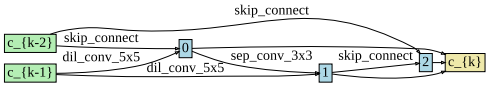
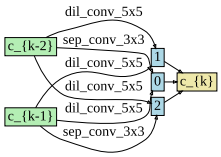
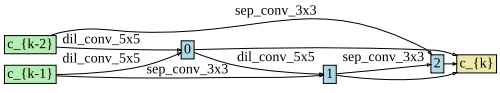
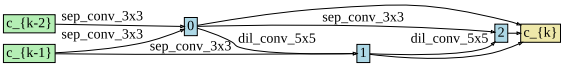
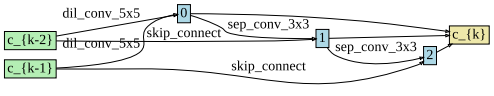
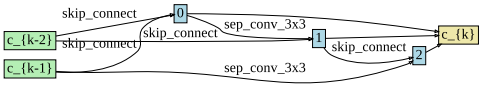
Task	Normal cell	Reduction cell
CIFAR100		
Spherical		
NinaPro		
FSD50K		
Darcy Flow		N/A
PSICOV		N/A
Cosmic		N/A
ECG		
Satellite		
DeepSEA		

FIGURE 6. Discovered cells in cell-based architecture search stage.

TABLE 4. Task-specific Hyperparameters.

Type	Task	Batch size	Input size	Kernel sizes (K)	Dilations (D)	Loss function
2D	CIFAR100	64	(32, 32)	3, 5, 7, 9	1, 3, 7, 15	Cross Entropy
2D	Spherical	64	(60, 60)	3, 5, 7, 9	1, 3, 7, 15	Cross Entropy
2D	NinaPro	128	(16, 52)	3, 5, 7, 9	1, 3, 7, 15	Focal
2D	FSD50K	128	(96, 101)	3, 5, 7, 9	1, 3, 7, 15	BCE w. Logits
2D Dense	Darcy Flow	10	(85, 85)	3, 5, 7, 9	1, 3, 7, 15	L2
2D Dense	PSICOV	8	(128, 128)	3, 5, 7, 9	1, 3, 7, 15	MSE
2D Dense	Cosmic	4	(128, 128)	3, 5, 7, 9	1, 3, 7, 15	BCE w. Logits
1D	ECG	1024	1000	3, 7, 11, 15, 19	1, 3, 7, 15	Cross Entropy
1D	Satellite	256	46	3, 7, 11, 15, 19	1, 3, 7, 15	Cross Entropy
1D	DeepSEA	256	1000	3, 7, 11, 15, 19	1, 3, 7, 15	BCE w. Logits

TABLE 5. Full-pipeline runtime in GPU hours for NAS-Bench-360 tasks evaluated on a NVIDIA A5000. In 8 out of 10 tasks, runtimes are similar or shorter than DASH. While D2NAS is inferior to ECG and satellite in runtime, this is not a serious disadvantage as it can create a very lightweight architecture with improved performance.

	Task	DASH	D2NAS (common)	D2NAS (custom)
2D	CIFAR100	3.9	4.0	18.2
	Spherical	7.1	7.1	7.7
	NinaPro	0.6	0.4	0.4
	FSD50K	39.2	27.5	35.2
2D Dense	Darcy Flow	1.2	1.0	-
	PSICOV	17.4	16.7	-
	Cosmic	5.5	4.4	5.9
1D	ECG	1.5	11.0	2.9
	Satellite	5.2	13.3	-
	DeepSEA	5.7	5.8	-

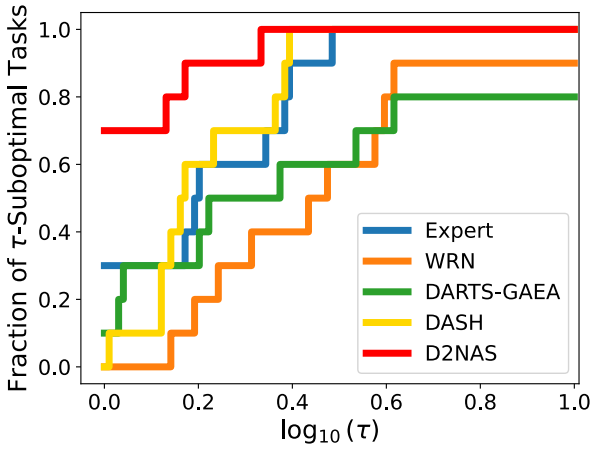


FIGURE 7. Comparison of the aggregate performance of D2NAS and other competing method on ten diverse tasks via performance profiles [14]. The y-axis is the fraction of tasks on which a method is within logτ-factor of the best. The larger the value is, the better.

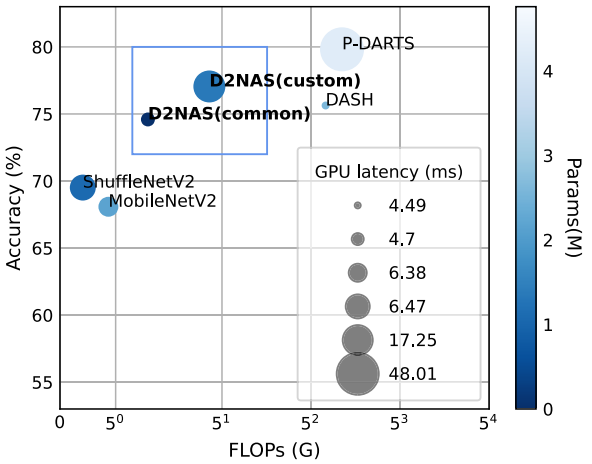


FIGURE 8. Comparison of architectures including lightweight architectures tested on CIFAR-100. D2NAS is the most advantageous in terms of number of parameters and creates an architecture that is not particularly inferior in terms of performance, FLOPs, and GPU latency.

A. EXPERIMENTAL SETUP

1) SEARCH SPACE

In our cell-based architecture search, we defined a fundamental search space consisting of five operations: separable convolution with a kernel size of 3, dilated separable convolution with a kernel size of 5, average pooling with a kernel size of 3, identity, and zero. It is worth noting that the 'average pooling' operation can be excluded when working with datasets that do not require pooling operations. This omission helps reduce the search space and, consequently, the search time.

During the kernel search stage, we defined an aggregated convolution search space with the following parameters: For 2D tasks, we considered kernel sizes ranging from $K = \{3, 5, 7, 9\}$ and dilations from $D = \{1, 3, 7, 15\}$. For 1D tasks, our search space encompassed kernel sizes from $K = \{3, 7, 11, 15, 19\}$ and dilations from $D = \{1, 3, 7, 15\}$.

As part of the DASH method, we replaced each convolution layer in the cell-based architecture with the aggregated convolution, while we left convolutions with a kernel size of 1 unchanged, as these kernels serve specific purposes. The details of all configurations are in Table 4.

2) ARCHITECTURE PARAMETERS

To initiate experiments without prior knowledge of the datasets, we defined a common set of hyperparameters. We categorized the 10 datasets in NAS-Bench-360 into three groups based on their types: 2D, 2D Dense, and

TABLE 6. Common hyperparameters for D2NAS pipeline.

		cell search		kernel search	hyperparam tuning	retrain
		stage 1	stage2			
grad clip	2D	5	5	5	-1	-1
	2D Dense	1	1	1	5	5
	1D	1	1	1	-1	-1
layers	2D	2	2	2	2	2
	2D Dense	3	4	4	4	4
	1D	2	3	3	3	3
init channels	2D	8	16	16	48	48
	2D Dense	8	16	16	48	48
	1D	8	16	16	48	48

TABLE 7. Custom hyperparameters for D2NAS pipeline.

		cell search		kernel search	hyperparam tuning	retrain
		stage 1	stage2			
grad clip	CIFAR100	5	5	5	-1	-1
	Spherical	1	1	1	-1	-1
	NinaPro	1	1	1	-1	-1
	FSD50K	1	1	1	-1	-1
	Cosmic	1	1	1	5	5
	ECG	1	1	1	-1	-1
layers	CIFAR100	5	11	11	11	11
	Spherical	2	2	2	2	2
	NinaPro	2	2	2	2	2
	FSD50K	2	2	2	2	2
	Cosmic	3	4	4	4	4
	ECG	2	3	3	3	3
init channels	CIFAR100	8	16	16	32	32
	Spherical	8	16	16	48	48
	NinaPro	8	16	16	48	48
	FSD50K	8	16	16	64	64
	Cosmic	8	16	16	32	32
	ECG	4	4	4	4	4

1D (see Table 1). Within each group, we applied common configurations for the number of cell layers and channels.

Given that we utilized P-DARTS for our cell-based architecture search, we structured the entire search process into two stages. The size of the operation space was set to 5 or 4 in stage 1 and reduced to 3 in stage 2, respectively. We imposed a minimum requirement of 2 layers and 4 input channels. In the common configuration, the number of layers L was constrained to the range $L = \{l|2 \leq l \leq 4\}$ at each P-DARTS stage, while the input channels were set to 8 and 16 for each stage.

During the search for kernel patterns, we kept the same number of layers and channels as those used in the last stage of the P-DARTS process. To enhance architectural capacity, we increased the number of channels several times during hyperparameter tuning and retraining. In the common configuration, the number of input channels was initially set to 16 during the kernel pattern search and later increased to 48 after completing the kernel pattern search.

Following the evaluation, we identified several tasks of the 10 tasks that needed improvement and reran the full pipeline with custom configurations. The details of all architecture parameters are in Tables 6 and 7.

B. RESULT AND DISCUSSION

We first study the overall performance of NAS algorithms over diverse types of tasks in NAS-Bench-360. Table 2 provides the error rate results for all tasks of NAS-Bench-360. D2NAS outperforms expert architectures in 8 out of 10 tasks and surpasses other automated baseline models in 9 out of 10 tasks. For more details, we conduct the aggregate performance [14] visualization in Fig. 7. As demonstrated in Table 2, D2NAS has the largest area under curve, which means that all tasks can achieve less than 0.4-suboptimality. The $\tau-$optimality implies that a given algorithm achieves a worse error rate by the value of τ than the best performance among all algorithms for comparison, and the fraction of $\tau-$suboptimal tasks means that the algorithm performs better

than τ-suboptimality in the given fraction of tasks. Compared to other baselines, it is remarkable that D2NAS shows 0-suboptimality in 60% of NAS-Bench-360 tasks and also reaches 1.0 at the lowest value of τ. For example, while DASH achieves the second-lowest τ value of 1.0, it does not achieve 0-suboptimality in any tasks.

For other considerations for the NAS algorithm, Table 5 shows that D2NAS has a shorter full-pipeline runtime in GPU hours for searching and training than DASH. In detail, D2NAS outperforms on 5 out of 10 tasks and achieves similar efficiency to DASH on 3 out of 10 tasks. We also report the test-time considerations in Table 3. We note that D2NAS significantly reduces the number of parameters by an average of 85.1%, with reductions up to 97.3%, and reduces FLOPs by an average of 91.3%, with reductions up to 99.3%. This is notable given that D2NAS reduces average error rates by 12.2% compared to DASH. Despite D2NAS having larger runtimes in ECG and Satellite, it is not a significant drawback because we can create a substantially lighter architecture with improved performance as demonstrated in Fig. 8, Furthermore, D2NAS has fewer parameters and excels in terms of GPU latency at test time. Therefore, it can be considered a viable option for mobile applications as well, and we report the detailed architectures for all tasks in Fig. 6.

The experimental results demonstrate that our approach facilitates the creation of more lightweight architectures compared to the conventional DASH approach, without incurring significant runtime penalties, while maintaining generally acceptable performance. Through the definition of a 'custom' hyperparameter set for datasets requiring further enhancement, we achieved superior performance or even more lightweight architectures than the common settings, without experiencing a significant drop in performance.

The advantages of our approach can be attributed to the remarkable capabilities of hierarchical architecture search, which enables the transcending of limitations inherent in the WRN. This approach allows for the identification and enhancement of sub-optimal architectures that are well-suited to serve as a robust backbone.

An in-depth study of cell structures shown in Fig. 6 and convolution patterns offers valuable insights into dataset characteristics. In our future work, we plan to analyze these patterns further. This study holds the potential to unveil innovative architectural enhancements, providing a deeper understanding of how these structures adapt to diverse datasets. Such insights can guide the design of more adaptable neural networks.

As mentioned in SectionV-A2, the first stage of our method for searching cell architecture is based on P-DARTS. For future work, it may be beneficial to extend the first stage beyond this framework by studying other variations of DARTS or alternative methods, as the optimization strategies of cell architecture are crucial, as shown in Fig. 4. This approach could allow various optimization algorithms to reveal more efficient and effective network configurations, fostering continuous improvement in the field.

VI. CONCLUSION

In this work, We have addressed the challenge of finding models tailored to underexplored domains through neural architecture search (NAS). Our proposed method, D2NAS, not only generates new architectures, but also enhances their performance. We have demonstrated that D2NAS achieves comparable or even superior accuracy while significantly reducing the number of parameters and floating-point operations per second (FLOPs) compared to existing methods that do not generate architectures.

One of the noteworthy aspects of D2NAS is its versatility as well as its practicality. This method can be effectively employed in mobile applications, where resource constraints are a crucial consideration. By achieving a balance between model complexity and performance, D2NAS appears to be promising for enhancing the efficiency and effectiveness of deep learning in mobile contexts. In summary, D2NAS addresses the challenges of custom architecture generation, model enhancement, and resource efficiency simultaneously. For novel domains and applications of deep learning, D2NAS may be effective in tailoring models to specific needs and constraints.

REFERENCES

[1] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.
[2] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.
[3] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.
[4] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.
[5] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 4780–4789.
[6] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017, *arXiv:1711.00436*.
[7] R. Tu, N. Roberts, M. Khodak, J. Shen, F. Sala, and A. Talwalkar, "NAS-Bench-360: Benchmarking neural architecture search on diverse tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 12380–12394.
[8] J. Fang, Y. Sun, Q. Zhang, Y. Li, W. Liu, and X. Wang, "Densely connected search space for more flexible neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10625–10634.
[9] E. Real, C. Liang, D. So, and Q. Le, "AutoML-zero: Evolving machine learning algorithms from scratch," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8007–8019.
[10] N. Roberts, M. Khodak, T. Dao, L. Li, C. Ré, and A. Talwalkar, "Rethinking neural operations for diverse tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 15855–15869.
[11] J. Shen, M. Khodak, and A. Talwalkar, "Efficient architecture search for diverse tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 16151–16164.
[12] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
[13] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Müller, and F. Petitjean, "InceptionTime: Finding AlexNet for time series classification," *Data Mining Knowl. Discovery*, vol. 34, no. 6, pp. 1936–1962, Nov. 2020.
[14] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, Jan. 2002.

[15] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient architecture search," 2019, *arXiv:1907.05737*.
[16] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1294–1303.
[17] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*.
[18] T. Dao, N. S. Sohoni, A. Gu, M. Eichhorn, A. Blonder, M. Leszczynski, A. Rudra, and C. Ré, "Kaleidoscope: An efficient, learnable representation for all structured linear maps," 2020, *arXiv:2012.14966*.
[19] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.
[20] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," 2020, *arXiv:2001.00326*.
[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[22] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling, "Spherical CNNs," 2018, *arXiv:1801.10130*.
[23] D. Josephs, C. Drake, A. Heroy, and J. Santerre, "sEMG gesture recognition with a simple model of attention," in *Proc. Mach. Learn. Health*, 2020, pp. 126–138.
[24] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: An open dataset of human-labeled sound events," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 30, pp. 829–852, 2022.
[25] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," 2020, *arXiv:2010.08895*.
[26] B. Adhikari, "DEEPCON: Protein contact prediction using dilated convolutional neural networks with dropout," *Bioinformatics*, vol. 36, no. 2, pp. 470–477, Jan. 2020.
[27] K. Zhang and J. Bloom, "DeepCR: Cosmic ray rejection with deep learning," *J. Open Source Softw.*, vol. 4, no. 41, p. 1651, Sep. 2019.
[28] S. Hong, Y. Xu, A. Khare, S. Priambada, K. Maher, A. Aljiffry, J. Sun, and A. Tumanov, "HOLMES: Health OnLine model ensemble serving for deep learning models in intensive care units," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1614–1624.
[29] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining Knowl. Discovery*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020.
[30] J. Zhou and O. G. Troyanskaya, "Predicting effects of noncoding variants with deep learning–based sequence model," *Nature Methods*, vol. 12, no. 10, pp. 931–934, Oct. 2015.
[31] M. Gabrié, A. Manoel, C. Luneau, N. Macris, F. Krzakala, and L. Zdeborová, "Entropy and mutual information in models of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1826–1836.
[32] K. Kawaguchi, Z. Deng, X. Ji, and J. Huang, "How does information bottleneck help deep learning?" 2023, *arXiv:2305.18887*.

JUNGEUN LEE (Member, IEEE) received the B.S. degree in electronics and communication engineering from Hanyang University, Seoul, South Korea, in 2013. She is currently pursuing the M.S. degree with the Department of Electrical and Computer Engineering, Seoul National University, Seoul. Additionally, she has been a Staff Engineer with the Modem Development Team, System LSI Division, Samsung Electronics Company Ltd., since 2013. Her research interests include neural architecture search and machine learning.

SEUNGYUB HAN (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from Seoul National University, South Korea, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include reinforcement learning and deep learning. He is currently a Reviewer of NeurIPS, ICML, and ICLR.

JUNGWOO LEE (Senior Member, IEEE) received the B.S. degree in electronics engineering from Seoul National University, Seoul, South Korea, in 1988, and the M.S.E. and Ph.D. degrees in electrical engineering from Princeton University, in 1990 and 1994, respectively. He is currently a Professor with the Department of Electrical and Computer Engineering, Seoul National University. He was a member of Technical Staff working on multimedia signal processing with SRI (Sarnoff), from 1994 to 1999, where he was the Team Leader (PI) of an $18M NIST ATP Program. He has been with the Wireless Advanced Technology Laboratory, Lucent Technologies Bell Laboratories, since 1999, and worked on W-CDMA base station algorithm development as the Team Leader, for which he received two Bell Laboratories technical achievement awards. He holds 21 U.S. patents. His research interests include wireless communications, information theory, distributed storage, and machine learning. He is also a member of the National Academy of Engineering of Korea. He received the Qualcomm Dr. Irwin Jacobs Award, in 2014, for his contributions in wireless communications. He was a co-recipient of the 2020 IEEE Communications Society Fred W. Ellersick Prize. He has also been the General Chair of JCCI 2019 and the Track Chair of IEEE ICC SPC (2016–2017) and was a TPC/OC Member of ICC 2005, ISITA 2005, PIMRC 2008, ISIT 2009, ICC 2015, ITW 2015, and VTC 2015s. He was an Editor of IEEE Wireless Communications Letters (WCL), from 2017 to 2021. He was an Associate Editor of IEEE Transactions on Vehicular Technology (2008–2011) and *Journal of Communications and Networks* (JCN) (2012–2016). He has also been a Chief Editor of KICS journal and an Executive Editor for *ICT Express* (Elsevier-KICS), since 2015.