machine learning & knowledge extraction

MDPI

Review

Gradient-Based Neural Architecture Search: A Comprehensive Evaluation †

Sarwat Ali * and M. Arif Wani *

Department of Computer Science, University of Kashmir, Srinagar 190006, India
* Correspondence: me.sarvat@gmail.com (S.A.); awani@uok.edu.in (M.A.W.)
† This paper is an extended version of our paper published in the Proceedings of the 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 12–14 December 2022.

Abstract: One of the challenges in deep learning involves discovering the optimal architecture for a specific task. This is effectively tackled through Neural Architecture Search (NAS). Neural Architecture Search encompasses three prominent approaches—reinforcement learning, evolutionary algorithms, and gradient descent—that have demonstrated noteworthy potential in identifying good candidate architectures. However, approaches based on reinforcement learning and evolutionary algorithms often necessitate extensive computational resources, requiring hundreds of GPU days or more. Therefore, we confine this work to a gradient-based approach due to its lower computational resource demands. Our objective encompasses identifying the optimal gradient-based NAS method and pinpointing opportunities for future enhancements. To achieve this, a comprehensive evaluation of the use of four major Gradient descent-based architecture search methods for discovering the best neural architecture for image classification tasks is provided. An overview of these gradient-based methods, i.e., DARTS, PDARTS, Fair DARTS and Att-DARTS, is presented. A theoretical comparison, based on search spaces, continuous relaxation strategy and bi-level optimization, for deriving the best neural architecture is then provided. The strong and weak features of these methods are also listed. Experimental results for comparing the error rate and computational cost of these gradient-based methods are analyzed. These experiments involved using bench marking datasets CIFAR-10, CIFAR-100 and ImageNet. The results show that PDARTS is better and faster among the examined methods, making it a potent candidate for automating Neural Architecture Search. By effectively conducting a comparative analysis, our research provides valuable insights and future research directions to address the criticism and gaps in the literature.

Keywords: Neural Architecture Search; gradient-based Neural Architecture Search; DARTS; PDARTS; Fair DARTS; Att-DARTS

Citation: Ali, S.; Wani, M.A. Gradient-Based Neural Architecture Search: A Comprehensive Evaluation. *Mach. Learn. Knowl. Extr.* **2023**, *5*, 1176–1194. https://doi.org/10.3390/make5030060

Academic Editor: Weiping Ding

Received: 27 June 2023
Revised: 9 August 2023
Accepted: 11 September 2023
Published: 14 September 2023

1. Introduction

One of the fundamental challenges in machine learning is feature engineering [1]. Deep learning's effectiveness in various problems is largely because of the automation of the process of feature engineering. This accomplishment, however, has been followed by an increase in demand for architectural engineering, in which increasingly complicated neural structures are constructed manually. The method of automating the process of architecture engineering, known as Neural Architecture Search (NAS), is, therefore, an obvious step in automating deep learning. NAS approaches have already surpassed manually developed architectures on some tasks, such as image classification [2,3]. The goal of NAS is to create a neural architecture that delivers the best possible performance while using limited computational resources in an automated manner with minimal human interference.

NASRL [2] and MetaQNN [4] are widely regarded as important initial NAS works. The neural networks created by these works employing reinforcement learning (RL) approaches have achieved good performance on tasks such as image classification. This indicates the

viability of automated neural architecture design. Following that, the work of Large-scale Evolution [5] confirmed the practicality of this notion by achieving good outcomes utilizing evolutionary learning. These solutions, however, demand hundreds of GPU days or much more computer resources. For example, Zoph et al. [6] invested 2000 GPU days in reinforcement learning (RL) to achieve the best architecture for CIFAR-10 and ImageNet datasets. On the other hand, Real et al. (2018) [7] used an evolutionary approach that took 3150 GPU days for the same purpose. As a result, improved approaches to Neural Architecture Search that ensure low processing resources, memory and power requirements are needed.

Gradient-based NAS methods substantially reduced search costs and achieved good performance through continuous relaxation [8]. These methods find better architectures with a local or global minimum to meet the constraints, such as computational burden [9].

One of the promising methods in the gradient-based NAS category is Differentiable Architecture Search (DARTS) [10], which has proven to be efficient and effective in finding good architectures. The authors concluded that its differentiable nature makes it more computationally efficient than other NAS methods. DARTS has successfully matched the performance of the state-of-the-art methods [6,7] despite utilizing significantly fewer computational resources. Specifically, DARTS required a maximum of 4 GPU days, in contrast to the extensive 2000 GPU days for NASNet [6] and the 3150 GPU days for AmoebaNet [7].

However, DARTS is not solitary in its approach. Some other popular DARTS-based methods that have been proposed in recent years include Progressive DARTS (PDARTS [11]), Fair DARTS [12] and Attention DARTS [13].

PDARTS = [11] is a progressive version of the DARTS method that starts with a simple architecture and gradually increases its complexity during the search process. This progressive approach aims to improve the quality of architectures found by the original DARTS method.

Many works [11,14,15] observed that in cells learned by DARTS, skip connections are aggregated. In Fair DARTS [12], the primary reason for the aggregation of redundant skip connections in DARTS was identified as an undue advantage. This occurs when certain operations contribute more to the competition within the architecture search process than to the overall performance of the resulting network when limited operations are allowed to compete.

Attention DARTS [13], another version of the DARTS method, uses attention mechanisms to boost the performance of the neural architectures generated by the original DARTS. Attention DARTS searches for the optimum architecture using differentiable attention modules, which allows the model to learn to focus on certain parts of the input during the search process. The search space utilized by these methods is cell-based, where each cell is composed of various convolution operations and the network architecture is obtained by stacking these cells together.

While gradient-based Neural Architecture Search (NAS) methods have shown promising results in automating architectural design, there are still some criticisms and gaps in the existing literature. The effectiveness of NAS heavily relies on the defined search space, and a poorly designed search space can hinder the algorithm's ability to discover optimal architectures [16]. The major gradient-based NAS methods have been observed to aggregate redundant skip connections [15]. This aggregation may lead to suboptimal architectures, limiting the performance gains that can be achieved through NAS. However, the current literature lacks a detailed comparative analysis of these specific methods, as reported in [17,18]. To address this limitation, this paper conducts a comprehensive evaluation of major gradient-based Neural Architecture Search (NAS) methods. By effectively conducting a comparative analysis, our research provides valuable insights and future research directions to address the criticism and gaps in the literature.

The rest of the paper is structured as follows: Section 2 presents an overview of the main DARTS-based methods; in Section 3, a theoretical comparison, based on search

spaces, continuous relaxation strategy and bi-level optimization, for deriving the best neural architecture is provided. This section also lists the strong and weak features of these methods. Section 4 analyzes the performance of gradient-based NAS methods for Image Classification. Finally, Section 5 presents future directions, and Section 6 outlines the conclusion.

2. Overview of Gradient-Based Neural Architecture Search Methods

This section provides an overview of the differentiable architecture search (DARTS) method and its variants, namely Progressive DARTS (PDARTS), Fair DARTS, and Attention DARTS (Att-DARTS). The main aim of these methods is to jointly optimize the operation strength parameters and optimal weights for these operations in order to obtain good architectures for image classification. Figure 1 shows the cell structure employed by these DARTS-based methods in which each operation is parameterized by a variable (α) that represents its strength. The details of search space and bi-level optimization are provided in Sections 3.1 and 3.2, respectively. An overview of major gradient-based methods is given below.

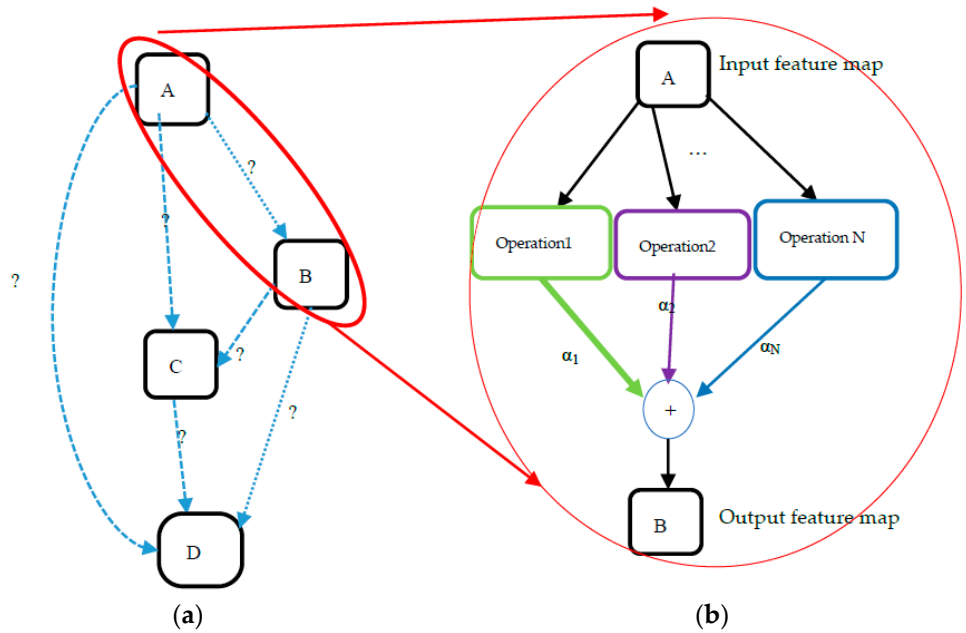

Figure 1. Continuous relaxation of the search space in DARTS cell. (**a**) DAG representing DARTS cell consisting of 4 nodes. (**b**) Mixed operation is shown. 'N' represents number of operations placed on the edges between each pair of nodes.

2.1. DARTS

Differentiable Architecture Search (DARTS) is a popular method for automating the design of neural networks. DARTS enables the search for optimal neural architectures by treating the architecture as a continuous variable and optimizing it using gradient-based methods. This approach allows the search process to be efficient and scalable, as the gradients can be computed using backpropagation. In DARTS, the search space is cell-based, and the operations included are 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling and identity. There is also a unique zero operation to indicate the absence of a connection between two nodes. These can be combined to create the basic building block of the DARTS architecture, known as a cell. A cell is a small neural network that can be stacked to form a larger network. The architecture of the cell is learned during the search process and can vary depending on the task at hand. DARTS search space is discussed in detail in Section 3.1.1.

DARTS uses a network of eight cells to learn the cell architecture. Out of these eight cells, six cells are normal cells, and two cells are reduction cells. Normal cells output an activation map with the same spatial dimension as the input, while reduction cells output an activation map with a reduced spatial dimension. By searching for the optimal cell architecture, DARTS can effectively automate the process of designing high-performance neural networks for a variety of tasks. The architecture search strategy and procedure for deriving the best architecture in DARTS are discussed in detail in Sections 3.3.1 and 3.4.1, respectively. The overview of DARTS is given in Figure 2.

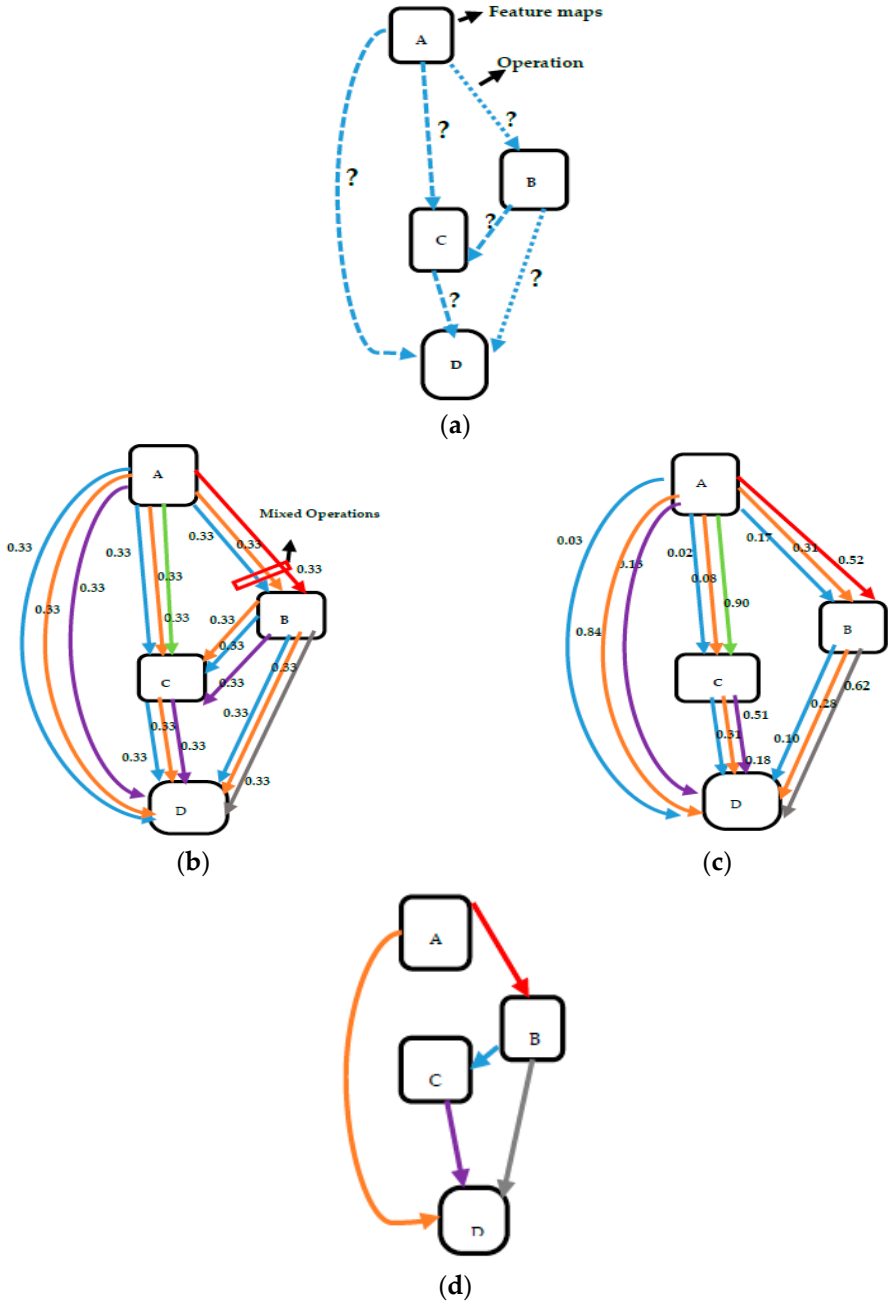

Figure 2. DARTS overview. (**a**) The operations on the edges are first unidentified. (**b**) Continuous relaxation of the search space through the placement of a mixture of candidate operations on each edge. (**c**) Solving a bi-level optimization problem to jointly improve the mixing probabilities and network weights. (**d**) Using the learned mixing probabilities to infer the final architecture.

In DARTS, the aim is to learn the operation strength probabilities (α) for various operations used to construct a cell and the optimal weights for these operations (ω). The details of bi-level optimization are given in Section 3.2.1.

The summary of the DARTS method is given below (Algorithm 1).

Algorithm 1 DARTS Algorithm

1. *Create a stacked network with eight cells (3rd and 6th cells in the network are reduction cells, and the remaining cells are normal cells), and all eight operations present in operation space are placed on each edge of the cell;*
2. *Initialize operation strength parameters (α) with zeros and weight parameters (ω) with random values;*
3. *Set loss function as cross entropy;*
4. *While not converged, do:*

   I *Update weights ω by descending the gradient of the training loss function $\xi \nabla \omega \mathcal{L}_{train}$ (ω, α) [ξ represents the learning rate parameter]*

   $$i.e.,\ \omega_i = \omega_i - \xi.(\partial \mathcal{L}_{train} / \partial \omega i)$$

   where $\mathcal{L}_{train}$ denotes cross entropy loss and y_i denotes labels in training data and y_i' denotes predicted labels. ξ represents the learning rate parameter;

   II *Update α until convergence by descending the gradient of the validation loss($\mathcal{L}_{val}$) function, i.e.,*
   $\alpha * [i][j] = \alpha[i][j] - \xi.(\partial\ \mathcal{L}_{val} / \partial\ \alpha[i][j])$
   where $\mathcal{L}_{val} = -\sum y_j log y_j$' which denotes cross entropy loss and y_j denotes labels in validation data and y_j' denotes predicted labels. ξ represents the learning rate parameter;

5. *Retain only one operation (o) at each edge, with maximum α value to obtain the final model structure, using argmax function*
   $o^{(i,j)} = argmax_{o \in O}\ \alpha_o^{(i,j)}$
   where α={ α(i,j) }.

The resulting cell is the final cell with its operation strength parameter and weight parameters as the optimal parameters.

2.2. PDARTS

Progressive Differentiable Architecture Search (PDARTS), a variant of DARTS, is another approach for automating the process of designing neural networks. PDARTS uses gradient-based optimization to learn the operation strength probabilities (α) for various operation strengths and the optimal weights (ω) for the operations. The details of the search space of PDARTS and its bi-level optimization are given in Sections 3.1.2 and 3.2.2, respectively. PDARTS begins with a base network and progressively grows it by adding new modules to it in stages. The details about the configuration of the base network used in PDARTS are provided in Section 3.3.2. At each step, the network is evaluated on a validation set, and the architecture is updated using gradient descent to minimize the loss function. The overview of the PDARTS method is given in Figure 3. The summary of the PDARTS algorithm is given below (Algorithm 2).

Algorithm 2 PDARTS Algorithm

1. *Construct a stacked network with five cells (three normal and two reduction cells), and all eight operations present in the operation space are placed on each edge of the cell. Set the dropout rate for skip connections to 0.0;*
   Obtain_networkparameters (); *#defined below*
2. *Add six normal layers to the network constructed in the previous step (the network now has nine normal and two reduction cells). This network contains only five good performing operations at each edge of the cell that were retained in the previous stage. Set the dropout rate for skip connections to 0.3;*
   Obtain_networkparameters (); *#defined below*
3. *Add six normal layers to the network constructed in the previous step (the network now has fifteen normal and two reduction cells). This network contains only three good performing operations at each edge of the cell that were retained in the previous stage. Set the dropout rate for skip connections to 0.6;*
   Obtain_networkparameters (); *#defined below*
   The architecture obtained at this stage is the final architecture. The cell obtained is the final cell with its operation strength parameter and weight parameters as the optimal parameters.

Procedure Obtain_networkparameters ();

1. *Initialize operation strength parameters (α) with zeroes;*
2. *Set loss function as cross entropy;*
3. *For the first two epochs, train the network weights;*
   Updated weights: $\omega_i = \omega i - \xi.(\partial\ \mathcal{L}_{train}/\partial\omega i)$ *(*ξ *represents the learning rate parameter);*
4. *Until convergence, train both the weights and operation strength parameters. The operation strength parameters are updated as:*
   $\alpha * [i][j] = \alpha[i][j] - \xi.(\partial\ \mathcal{L}_{val}/\partial\ \alpha[i][j])$*;*
5. *Retain the top performing operations at each edge, with maximum α value to obtain the final model structure (for stage 1, retain five out of eight operations; for stage 2, retain three out of five operations; and for stage 3, retain only top one operation).*

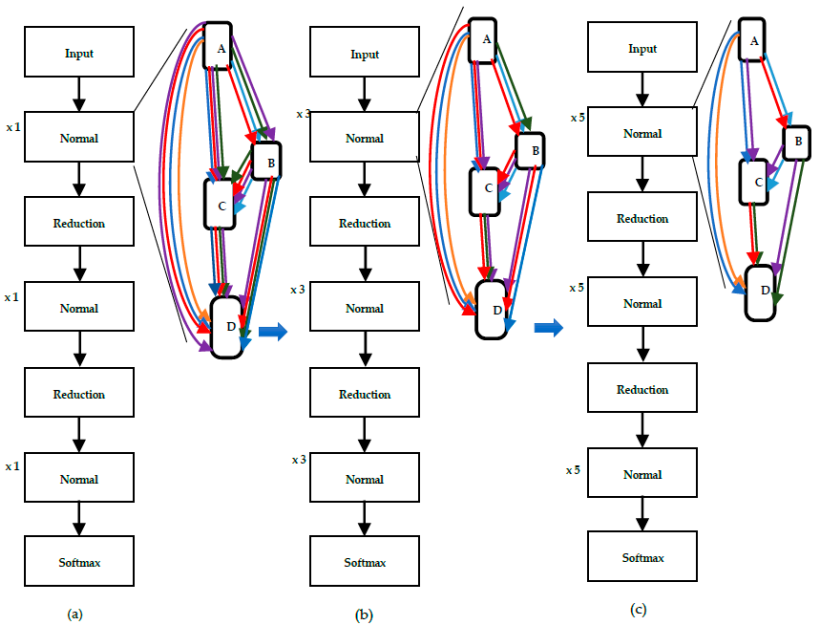

Figure 3. This figure is adapted from [11] with a few modifications. The overall pipeline of PDARTS. (**a**) Depicts the initial stage with a shallow architecture and each cell containing all operations. (**b**) Represents the intermediate stage with a deeper architecture and each cell containing a reduced set of operations. (**c**) Illustrates the final stage with an even deeper architecture and each cell incorporating the fewest operations. (here, ×1 implies 1 cell, ×3 implies 3 cells and ×5 implies 5 cells).

2.3. Fair DARTS

Fair DARTS is an extension of the Differentiable Architecture Search (DARTS) method that incorporates a fairness constraint into the architecture search process. The search space used is the same as that in the case of DARTS and is described in detail in Section 3.1.3. The authors of Fair DARTS found that when skip connections become prominent, DARTS suffers from considerable performance deterioration. According to the authors, excessive skip connections are caused by unfair competition among diverse operations. During training in DARTS, the weight of a skip connection grows faster than that of other operations due to the softmax operation, which leads to exclusive competition among different operations. As a result, the optimization process becomes biased towards skip connections, and they dominate the final model. So, the authors suggested that while skip connections are useful for enhancing information flow, other operations should also be given equal importance to prevent being unfairly overshadowed by skip connections. To achieve this, they introduced the use of sigmoid activation (σ) for each operation strength weight, $\alpha_{i,j}$, which lets operations be switched on and off separately without being interrupted. Details are provided in Section 3.2.3. The architecture search strategy and the techniques for deriving the best architecture in Fair DARTS are discussed in detail in Sections 3.3.3 and 3.4.2, respectively. The summary of the Fair DARTS algorithm is given below (Algorithm 3).

Algorithm 3 Fair DARTS Algorithm

1. *Create a stacked network with eight cells (3rd and 6th cells in the network are reduction cells, and the remaining cells are normal cells), and all eight operations present in operation space are placed on each edge of the cell;*
2. *Initialize operation strength parameters (α) with zeroes and weight parameters (ω) with random values;*
3. *Set training loss to the sum of cross entropy loss and mean square loss;*
4. *Do the following steps until convergence:*
   Update weights ω by descending the gradient of the training loss function $\xi \nabla \omega \mathcal{L}_{train}$ *(ω, α), i.e.,* $\omega_i = \omega_i - \xi.(\partial\ \mathcal{L}_{train}/\partial\omega_i)$ *(ξ represents the learning rate parameter);*
5. *Retain only one operation at each edge, with maximum sigmoid weighted α values by comparing* $\sigma(\alpha[i][j])$ *with* $\sigma_{threshold}$*(zero-one loss).*
   If $(\sigma\left(\alpha^{(i,j)}\right) > \sigma_{threshold}\Big)$
   $\alpha^{(i,j)} = 1$
   Else
   $\alpha\alpha^{(i,j)} = 0$
   where $\sigma_{threshold} = 0.85$
   The resulting cell is the final cell with its operation strength parameter and weight parameters as the optimal parameters.

2.4. Attention DARTS

Attention Darts is a Neural Architecture Search method that builds upon the DARTS (Differentiable Architecture Search). The main innovation of Attention Darts is the introduction of a new attention mechanism that allows the method to focus on the most important operations in the network during the search process. The details of the search space are given in Section 3.1.4. During the exploration phase, Att-DARTS examines the cell, which includes both attention modules and operations. Att-DARTS assumes that an edge in the cell is formed by an operation preceding an attention module. Att-DARTS recognizes an operation (from the operation space) and an attention module (from the attention module space) during the search stage. The details about bi-level optimization and architecture search procedure in Fair DARTS are given in Sections 3.2.4 and 3.3.4, respectively. Thus, the aim of Att-DARTS is to learn operation strength parameters (α), attention module strength parameters (β) and weight parameters (ω). Details about discretization are provided in Section 3.4.1. The overview of Att-DARTS is given in Figure 4. The summary of the Att-DARTS algorithm is given below (Algorithm 4).

Algorithm 4 Att-DARTS Algorithm

1. *Create a stacked network with eight cells (3rd and 6th cells in the network are reduction cells, and the remaining cells are normal cells), and all eight operations present in the operation space and six attention modules present in the attention module space are placed on each edge of the cell;*
2. *Initialize operation strength parameters (α) and attention strength parameters ($_{\beta}$) with zeros and weight parameters (ω) with random values;*
3. *Set loss function as cross entropy;*
4. *While not converged, do*
   I. Update weights ω by descending the gradient of the training loss function $\xi\nabla\omega\mathcal{L}_{train}$ (ω, α, $_{\beta}$), i.e.,
   $\omega i = \omega i - \xi.(\partial\ \mathcal{L}_{train}/\partial\omega i)$
   where $\mathcal{L}_{train}$ denotes cross entropy loss and y_i denotes labels in training data and y_i denotes predicted labels. ξ represents the learning rate parameter;
   II. Update α and $_{\beta}$ until convergence by descending the gradient of the validation loss function, i.e.,
   $\alpha * [i][j] = \alpha\ [i][j] - \xi.(\partial\ \mathcal{L}_{val}/\partial\ \alpha\ [i][j])$
   $_{\beta}{}^{*}\ [i][j] = _{\beta}\ [i][j] - \xi.(\partial\ \mathcal{L}_{val}/\partial\ _{\beta}\ [i][j])$
   which denotes cross entropy loss and y_i denotes labels in validation data and denotes predicted labels (ξ represents the learning rate parameter);
5. *Retain only one operation (o) with maximum α value followed by one attention module (a) with maximum $_{\beta}$ value to obtain the final model structure, using the argmax function.*

$$o^{(i,j)} = argmax_{o\in O}\ \alpha_o^{(i,j)}$$
$$a^{(i,j)} = argmax_{a\in A}\ \beta_a^{(i,j)}$$

where $\alpha = \{\alpha^{(i,j)}\}$ and $_{\beta} = \{_{\beta}{}^{(i,j)}\}$

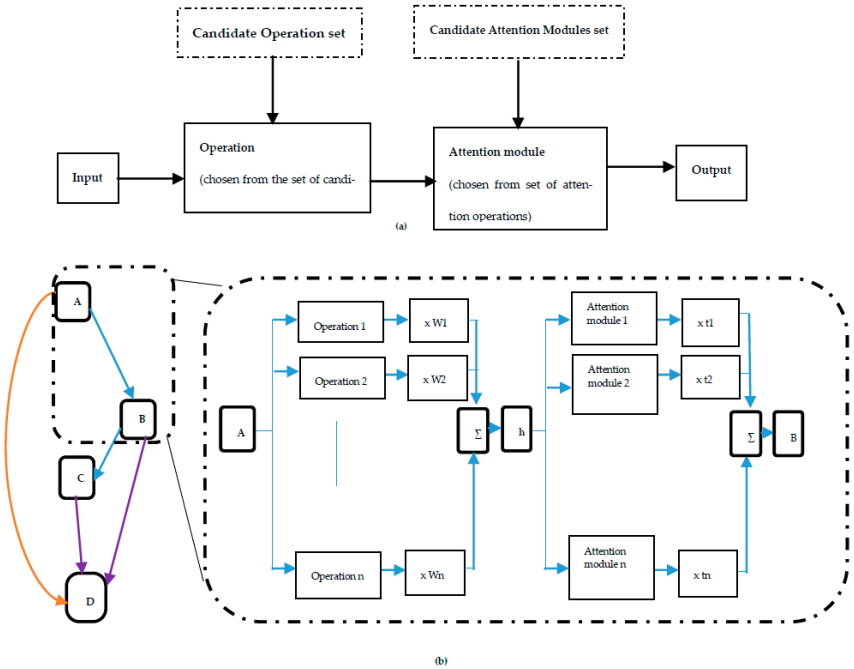

Figure 4. This figure is adapted from [13] with a few modifications. (**a**) Overview of Att-Darts. Both the operation and attention module are chosen from the candidate space. (**b**) Illustration of how operation, as well as attention module, is inserted between the nodes.

The resulting cell is the final cell with its operation strength parameter, attention module strength parameters and weight parameters as the optimal parameters.

3. Theoretical Comparison of Gradient-Based NAS Methods

In this section, various components of major DARTS-based methods are explored. These methods are compared based on several key factors, including their search space, con-

tinuous relaxation, bi-level optimization, architecture search strategies, and the procedures for deriving the best architectures.

3.1. Comparison Based on Search Spaces

The neural architectures that can be defined are identified by the search space. The search space is more clearly defined by the neural network's selected operation set and hyperparameters. These options govern which neural architectures NAS may search for [19].

The two broad categories of search spaces are Global search space and cell-based search space. The global search space combines various operations in all possible ways, resulting in a huge search space [4]. The produced combinations are hierarchical networks with a lot of freedom in terms of how operations are grouped. Cell-based strategies have been developed to "modularize" the search space, as seen in previous works [20–22]. That is, distinct layers are combined in the form of modules. The choice of search space affects the quality of the search [23].

The search space employed by the major gradient-based NAS methods is cell-based. A cell is a special block in which layers are assembled much like any other model. These cells use a variety of operations to generate feature maps/activation maps that may be handed on to other cells. A complete model is created by stacking these cells in succession. The major DARTS-based methods (DARTS, PDARTS, Fair DARTS and Attention DARTS) employ two kinds of cells:

Normal cell: These cells output an activation map with the same spatial dimension as the input. Convolutions and pooling's in this block have a stride of 1.
Reduction cell: These cells output an activation map with a reduced spatial dimension. This block's convolutions and pooling's have a stride equal to 2. The job of the reduction cell is to down sample the activation maps.

In these methods, Directed Acyclic Graphs (DAGs) are used to represent the cells. A DAG is made up of N nodes ($x^{(i)}$) that represent a set of feature mappings and edges (i,j) that represent operations ($o^{(i,j)}$) on the feature maps. The number of hidden nodes in a convolution neural network (CNN) cell is set to four. There are two inputs and one output for each cell. The outputs of the previous two cells are used as two inputs. The output of the cell is obtained by conducting a reduction operation (for example, concatenation) on all of the cell's intermediary nodes. An intermediate node is calculated using the information from all of its predecessor nodes:

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)}) \tag{1}$$

3.1.1. DARTS

To explore a cell in a convolution neural network, DARTS employs seven candidate operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling and identity. There is also a unique zero operation to indicate the absence of a connection between two nodes. Initially, all these operations, with their strengths initialized to zeroes, are placed between each pair of nodes. As the search progresses, their optimal strengths, as well as weights, are learned. This enables DARTS to automatically identify the most effective operations for constructing the neural architecture. The illustration is provided in Figure 2.

3.1.2. PDARTS

PDARTS, an extension of DARTS, adopts a multi-stage search process with specific adjustments in each stage. In the first stage, PDARTS utilizes a similar search space and network configuration as DARTS but sets the number of cells to five for faster exploration. The operation space remains complete, consisting of eight candidate operations. Moving to the second stage, the number of cells is increased to 11, and the operation space size is

reduced to five by eliminating three less effective operations. Finally, in the third stage, the number of cells is further increased to 17, while the operation space size is reduced to three. This dynamic change in the operation space across stages is known as "search space approximation", a technique that is thoroughly explained in Section 3.3.2 of the paper. By employing this approach, PDARTS aims to strike a balance between exploration efficiency and performance to discover more optimized architectures for image classification tasks. The search process of PDARTS is visually depicted in Figure 3.

3.1.3. Fair DARTS

In Fair DARTS, the authors have utilized two different search spaces for the PDARTS method. The first search space is the same as the one used in DARTS, which includes seven candidate operations: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, and identity operation.

In addition to the DARTS search space, the authors have also explored the ProxylessNAS search space [23]. The ProxylessNAS search space is a larger search space that includes a more extensive set of candidate operations, allowing for a more comprehensive exploration of architectural possibilities.

3.1.4. Attention DARTS

In the DARTS, PDARTS, and Fair DARTS methods, the focus was primarily on exploring the search space of convolutional neural networks (CNNs), which involve using convolutions to analyze both spatial and channel-based features of images. However, it was observed that this approach might also consider irrelevant characteristics, leading to suboptimal architectures. To address this limitation, the authors of Att-DARTS introduced attention modules to the search space.

In Att-DARTS, the search space includes not only the traditional convolutional and pooling operations but also prospective attention modules. Attention modules enable the model to filter out unimportant elements and prioritize essential features, allowing for more effective information processing. The operation space in Att-DARTS remains the same as in the other methods, including 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, and identity operation. However, the attention space (A) includes additional attention modules such as Identity, Squeeze-and-Excite, Gather-Excite, Bottleneck Attention Module, Double Attention Block, and Convolution Block Attention Module.

CNNs equipped with attention modules have shown better performance compared to those without attention modules, even when both models have the same number of parameters. This suggests that attention modules are a promising option to be incorporated into neural networks within the search space. By considering attention modules in the search process, Att-DARTS aims to discover architectures that are not only efficient in their convolutional operations but also leverage attention mechanisms to improve their overall performance in handling image classification tasks.

3.1.5. Strong and Weak Features

Strong features:

1. All these methods have used cell-based search space, which allows for fine-grained control over neural architecture design. Candidate operations can be combined in various ways to form cell architecture, leading to more flexibility in design;
2. Cells can be stacked to form the overall network architecture, allowing for customization and scalability;
3. Attention modules used in Att-DARTS focus on relevant aspects of images.

Weak features:

1. Determining the optimal structure of a cell is a challenging task;

2. Convolution operations used in DARTS, PDARTS, and Fair DARTS may also be considered irrelevant characteristics. Without attention modules, unimportant elements may not be filtered out, leading to inefficiencies and potential performance issues.

3.2. Comparison Based on Continuous Relaxation and Bi-Level Optimization

3.2.1. DARTS

In DARTS-based methods, mixed operations are placed on the edges between the nodes in the cell. Every edge O(i,j) of the DAG is a linear combination of all available operations. Each operation has some weight such that the sum of these weights (represented by α) is 1. Thus, an edge represents the weighted combination of all the available operations. This continuous variable (α) can then be optimized using Gradient Descent. The viable cell topologies are first found by learning a collection of these continuous variables (α) that parameterize the operation mixing weights for a given set of nodes. The categorical decision of operations is relaxed to Softmax over all feasible operations in DARTS, as given in Equation (2).

$$\bar{O}(i,j)(x) = \sum_{o \in O} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(i,j)}\right)} o(x) \tag{2}$$

where $\bar{O}(i,j)(x)$ represents mixed operations placed on the edge.

Figure 1 depicts the continuous relaxation of the search space in a cell. In DARTS-based methods, the challenge of searching a neural architecture is then reduced to learning a collection of variables, $\alpha = \{\alpha^{(i,j)}\}$ that represent the strength of various operations placed on edges between each pair of nodes in the cell, as shown in Figure 2. When the search is finished, the discrete architecture is generated by substituting mixed operations between nodes with the most probable operation.

After the relaxation of the search space, the next step is bi-level optimization that aims to jointly learn both the architecture parameters (α) and the weights (ω) within all the operations in the network. The training loss ($\mathcal{L}_{train}$) and validation loss ($\mathcal{L}_{val}$) are affected by both the network structure and the weight of the network (ω). The goal of the architecture search is to find α^* that minimizes the validation loss $\mathcal{L}_{val}(\omega^*,\alpha^*)$, where the weights ω^* associated with the architecture are obtained by minimizing the training loss ω^* = argmin $\omega\mathcal{L}_{train}(\omega, \alpha*)$. The problem can be expressed as follows in a bi-level optimization form (Equations (3) and (4) below):

$$\min_{\alpha} \mathcal{L}_{val}(\omega^*(\alpha), \alpha) \tag{3}$$

$$\text{s.t. } \omega^*(\alpha) = argmin_{\omega}\ \mathcal{L}_{train}(\omega, \alpha) \tag{4}$$

The optimization problem has thus been posed as to finding α's so that validation loss is minimized, given that we have weights that are already optimized on the training set. However, Equation (3) contains a computational challenge. To determine the best weights, the network must be trained by minimizing the training loss, which necessitates repeatedly modifying the weights as the network design (α) changes. This repetition makes training impractical and necessitates a large amount of computational resources and time. To address the problem posed by the inner optimization step, a basic approximation method was proposed by authors to compute the gradient, which consists of:

$$\nabla\alpha\mathcal{L}_{\text{val}}\left(\omega(\alpha),\ \alpha\right) \tag{5}$$

$$\approx \nabla\alpha\mathcal{L}_{\text{val}}(\omega - \xi\nabla\text{w}\mathcal{L}_{\text{train}}\left(\omega,\ \alpha\right),\ \alpha) \tag{6}$$

where ω represents the existing weights and ξ represents the inner optimization step's learning rate. Instead of entirely solving the inner optimization (Equation (3)) by training until convergence, the goal is to approximate $\omega^*(\alpha)$ by modifying ω in a single training iteration.

The update rule for optimizing the weights (ω) using the training loss as a guide is represented in the equation below:

$$\omega^* = \omega - \xi \nabla \omega \mathcal{L}_{\text{train}}\ (\omega,\ \alpha) \tag{7}$$

Moreover, the update rule for optimizing the operation strength parameter (α) by using the validation loss as a guide is represented in the equation below:

$$\alpha* = \alpha - \nabla \alpha \mathcal{L}_{\text{val}}(\omega - \xi \nabla \omega \mathcal{L}_{\text{train}}\ (\omega,\ \alpha),\ \alpha) \tag{8}$$

where $\nabla \omega \mathcal{L}_{\text{train}}$ is the gradient of the training loss with respect to the weights ω and $\nabla_{\alpha} \mathcal{L}_{\text{val}}$ is the gradient of the validation loss with respect to the hyperparameters α. Learning rate, ξ, is a hyperparameter that determines the step size of the update.

3.2.2. PDARTS

In PDARTS, the continuous relaxation technique used is the same as the one employed in DARTS. Continuous relaxation involves representing the architectural parameters as continuous variables, which enables gradient-based optimization methods to be applied during the search process.

By treating the architectural parameters as continuous variables, the search space becomes differentiable, enabling the use of gradient descent methods to optimize the architecture. This continuous relaxation allows for the efficient and scalable exploration of the vast search space of possible architectures, making it computationally feasible to find high-performing neural architectures.

3.2.3. Fair DARTS

In contrast to DARTS and PDARTS, the authors of Fair DARTS found that when skip connections become prominent, DARTS suffers from considerable performance deterioration. According to the authors, excessive skip connections are caused by unfair competition among diverse operations. The skip connection in DARTS, PDARTS and Attention DARTS is softmax-weighted and appended to the output, similar to a simple residual module present in ResNet [24]. While this helps in training, the weight of a skip connection grows faster than that of other operations due to the softmax operation, which leads to exclusive competition among different operations. As a result, the optimization process becomes biased towards skip connections, and they dominate the final model. This leads to a reduction in the effectiveness of convolutional operations and an overall reduction in performance. To address the downfalls of skip connections, the authors proposed a collaborative approach to address the issue of unfair advantage caused by excessive use of skip connections. They suggested that while skip connections are useful for enhancing information flow, other operations should also be given equal importance to prevent being unfairly overshadowed by skip connections. To achieve this, they introduced the use of sigmoid activation (σ) for each $\alpha_{i,j}$, which lets operations be switched on and off separately without being interrupted. As a result, Equation (2) in DARTS is replaced with a new Equation (9) that incorporates the sigmoid activation.

$$\bar{O}(i,j)(x) = \sum_{o \in O} \sigma(\alpha_o^{(i,j)}) o(x) \tag{9}$$

3.2.4. Attention DARTS

The Att-DARTS method identifies an operation from the set of operations and an attention module from the set of attention modules. Att-DARTS incorporates attention modules alongside candidate operations, and the relative weights of these modules are determined by applying the softmax operation over all the possible parameters of the candidate attention modules (in a similar manner as in Equation (2)).

3.2.5. Strong and Weak Features

Strong features:

1. Single-step inner optimization used in these methods reduces computational time and simplifies the optimization process.
2. The sigmoid function used in Fair DARTS allows for independent switching of operations, providing more flexibility in the design of neural architectures.

Weak features:

1. Using single-step inner optimization in these methods to learn optimal operation strength values may not lead to the most optimal results, potentially sacrificing performance for computational efficiency.
2. Deciding the optimal threshold for the zero loss function used in Fair DARTS can be difficult and may not result in the most optimal network architecture.
3. Adding attention modules to the search space in methods such as Att-DARTS adds an additional computational burden in finding the optimal attentional module strengths, potentially increasing the computational resources needed.

3.3. Comparison Based on Architecture Search Strategies

In this section, the architectures of networks used during architecture search and the differences in architecture search strategies in DARTS, PDARTS, FairDARTS and Att-DARTS are highlighted.

3.3.1. DARTS

DARTS utilizes shallow networks consisting of eight cells during the architecture search process. Among these, there are two reduction cells located at the 3rd and 6th positions in the network, while the remaining cells are normal cells. Although the architecture search is performed on this shallow network, the evaluation of the identified structure is conducted on a deeper network comprising at least 20 cells. The behaviors of deep and shallow networks exhibit significant differences, indicating that the structures chosen during the search phase might not always be optimal for evaluation. This discrepancy between the depth of the search and evaluation networks is commonly referred to as the "depth gap" in the context of Neural Architecture Search. Figure 5 illustrates the depth gap problem.

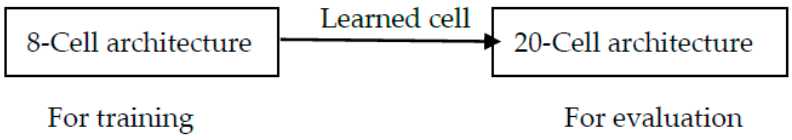

Figure 5. Depth Gap between Search and Evaluation in DARTS.

3.3.2. PDARTS

In contrast, the authors of PDARTS argue that when examining the architecture space, depth continues to be the key component. So, they try to reduce the gap in depth between search and evaluation scenarios by using a progressive variation of a differentiable architectural search method, as illustrated in Figure 3. The fundamental concept is to incrementally deepen candidate architectures throughout the search process. Two useful strategies to regularize and approximate the search process, respectively, have been developed by the authors to address the problems of computational overhead and instability.

The authors carried out multiple iterations of the DARTS search procedure and found that the normal cells in the resulting architectures tend to have shallow connections rather than deep ones. This results from the fact that neural networks that have relatively few layers of nodes frequently experience faster gradient falls during the search process, which goes against the conventional wisdom that deeper networks typically outperform shallow networks. Thus, in order to narrow the depth gap, a method was suggested that gradually

increases the depth of the network throughout the search stage until the depth is adequately similar to the evaluation setting. However, this approach raises two problems: increased processing overheads and decreased search stability. The number of structures grows exponentially with depth, causing problems with both time and memory. Specifically, GPU memory utilization in DARTS is dependent on the depth of explored networks.

PDARTS used a search space approximation strategy to address this issue, which progressively reduces the number of potential operations at the conclusion of each stage, with the preceding stage's scores serving as the selection criterion. Additionally, while conducting a search on a deeper network architecture, the differentiable methods have a tendency to favor the skip connect operation. This operation speeds up forward and backward propagation and often results in the quickest gradient descent path. However, as it has no parameters, its ability to learn visual representations is limited. For that reason, another approach called search space regularization is suggested, which is discussed in later sections, which includes operation-level dropout [25] to prevent the network architecture from 'over-fitting' and limit the number of retained skip connects for increased stability.

3.3.3. Fair DARTS

Like DARTS, Fair DARTS employs a shallow network of eight cells for architecture search. In this network, two reduction cells form the 3rd and 6th cells, while the rest are normal cells. However, the identified structure is evaluated in a deeper network with at least 20 cells. The main difference between the two methods is that Fair DARTS uses sigmoid weighted operation strength parameters instead of softmax weighted operation strengths, which prevents the strengths of one operation from suppressing other operations. This approach provides more stability during the optimization process.

3.3.4. Attention DARTS

On the other hand, Att-DARTS uses attention modules to seek a neural architecture, as shown in Figure 4. The entire network is made up of recurring cells, and Att-DARTS searches for the good ones. During the exploration phase, Att-DARTS examines the cell, which includes both attention modules and operations. While original attention research suggested various positions for incorporating attention modules, the most commonly preferred location is directly after a convolution operation. Att-DARTS follows this approach and assumes that an edge in the cell is formed by an operation preceding an attention module.

Att-DARTS recognizes an operation (from the operation space) and an attention module (from the attention module space) during the search stage. When the emphasis is on the edge from one node x_i to another node x_j, the softmax operation over the operation parameters of candidate operations determines the relative weight of each candidate operation. Similarly, the softmax operation over attention module parameters existing in the search space determines the relative weight of each candidate attention module. On the edge from node x_i to x_j, an additional node $h_{i,j}$ is created in such a way that the flow of information from node xi to node $h_{i,j}$ may be estimated as the weighted total of all candidate operations. The information flow between $h_{i,j}$ and x_j is given by the weighted total of all candidate attention modules.

3.3.5. Strong and Weak Features

Strong features:

1. PDARTS addresses the depth gap issue that DARTS suffers from by conducting the search in stages and gradually increasing the depth of architecture;
2. The search space approximation in PDARTS reduces the number of operations in the search space as the search progresses, which reduces computational complexity.

Weak features:

1. DARTS suffers from a depth gap issue between the search and the evaluation phase;

2. The regularization of the search space in PDARTS reduces the number of skip connections, but finding the optimal dropout rate can be challenging.

3.4. Comparison of Procedures for Deriving Best Neural Architecture

Once the architecture search is completed, the last step is to derive the best cell architecture by retaining only those operations whose strengths (probabilities) are optimal, i.e., topmost operations.

3.4.1. DARTS, PDARTS and Attention DARTS

DARTS, PDARTS and Att-DARTS face the problem of the difference between continuous representation and discrete encoding because it searches for a continuous relaxation of the architecture search space, which is then discretized to obtain a final architecture. During the search process, these methods learn the continuous weights of operations, which are then used to construct a discrete architecture. However, the resulting architecture is not guaranteed to be a good approximation of the continuous weights due to the discreteness of the encoding. This can lead to a performance gap between the continuous relaxation and the final discrete architecture. To address this issue, these employ various techniques to ensure that the final architecture is a good approximation of the continuous relaxation, such as a regularization term to encourage the weights to be close to a discrete value. In these methods, relaxing continuous choices from discrete categorical ones should result in a near approximation during the searching phase.

3.4.2. Fair DARTS

Fair DARTS addresses the problem of the difference between continuous representation and discrete encoding faced by methods such as DARTS, PDARTS, and Att-DARTS. These methods search for a continuous relaxation of the architecture search space, which is then discretized to obtain a final architecture. However, the resulting architecture may not be a good approximation of the continuous weights due to the discreteness of the encoding, leading to a performance gap between the continuous relaxation and the final discrete architecture. Fair DARTS tackles this issue by forcing the sigmoid value of architectural strength towards 0 or 1 using an additional loss known as zero-one loss. This helps to improve the approximation of the final architecture to the continuous relaxation during the searching phase.

3.4.3. Strong and Weak Features

Strong features:

1. The regularization techniques used in these methods ensure that the final discrete architecture is a good approximation of the continuous relaxation;
2. The use of an additional loss function in Fair DARTS helps to address the problem of the difference between continuous representation and discrete encoding and ensures that the resulting architecture is a good approximation of the continuous relaxation.

Weak features:

1. The discreteness of the encoding used in DARTS, PDARTS, and Att-DARTS methods can lead to a performance gap between the continuous relaxation and the final discrete architecture;
2. The use of an additional loss function in Fair DARTS may increase the difficulty of finding the optimal threshold for the sigmoid value of architectural strength.

4. Experimental Comparison of Gradient-Based NAS Methods

In this section, a comparison of already discussed gradient-based NAS works on various image classification datasets such as CIFAR-10, CIFAR-100 and ImageNet is presented. The evaluation metrics include error rate (in percentage), the number of parameters (in millions), and computation cost (measured in GPU days/hours). The summarized results of these methods on the CIFAR-10 dataset are presented in Table 1.

Table 1. Performance comparison of gradient-based NAS methods on CIFAR-10 dataset.

Works	Error Rate (%)	Parameters (M)	GPU Days/Hours
[10] DARTS + Cutout	2.76 ± 0.09	3.3 M	4 Days
[11] PDARTS	**2.25**	10.5 M	**0.3 Days**
[12] Fair DARTS	2.54	2.8 M	-
[13] Att-DARTS + cutout	2.54 ± 0.10	3.2 M	-

The differentiable architecture search method is a promising approach because it is the first to use gradient optimization to address the problem of NAS. This method differs from traditional NAS methods, which involve a discrete search space and require a time-consuming process of training and evaluating numerous architectures. Instead, a differentiable architecture search directly optimizes the architecture using a gradient-based approach. This method achieved an error rate of 2.76% ± 0.09, which is comparable to state-of-the-art results, in just 4 days, which is significantly less computational time compared to previous works as reported by [26]. DARTS initially learned a cell on a small network consisting of eight cells, but when it was tested on a larger network with 20 cells, its performance decreased.

To address this issue, PDARTS was developed as a follow-up to DARTS, which utilized a progressive approach to learn cells that can perform better even in deeper architectures. The use of search space regularization and approximation strategies in PDARTS enabled it to achieve better results than DARTS while using less computation time. PDARTS achieved an error rate of 2.25% on the CIFAR-10 dataset in just 7 h, which is far less computational time than DARTS.

Fair DARTS, on the other hand, focused on reducing the dominance of skip connections that were observed in cells learned by DARTS. The authors used the sigmoid activation function to allow independent on-off switching of operations. This prevented skip connections from suppressing other operations. The Error rate (%) achieved by Fair DARTS on CIFAR-10 is 2.54%.

Another work, Att-DARTS, incorporated attention modules in search space to enable the exclusion of unnecessary information and concentrate solely on pertinent aspects, which improved network performance as compared to DARTS. This work achieved an error rate of 2.54% on the CIFAR-10 dataset while having a comparatively smaller number of parameters (3.2 M) than all other methods. Table 2 shows the summarized outcomes of these methods applied to the CIFAR-100 dataset. Remarkably, PDARTS has achieved a remarkable error rate of 14.64%, surpassing all other methods by a significant margin despite having more parameters than the other methods. Att-DARTS obtained an error rate of 16.54% ± 0.40, whereas DARTS obtained an error rate of 2.76% ± 0.09 on CIFAR-100. However, Fair DARTS showed a decreased performance with an Error rate (%) of only 22.8% on the same dataset.

Table 2. Performance comparison of gradient-based NAS methods on CIFAR-100 dataset.

Works	Error Rate (%)	Parameters (M)	GPU Days/Hours
[10] DARTS + Cutout	17.54	3.3 M	4 Days
[11] PDARTS	**14.64**	11.0 M	**0.3 Days**
[12] Fair DARTS	22.8	5.3 M	3 Days
[13] Att-DARTS + cutout	16.54 ± 0.40	3.2 M	-

Evaluation results and comparison of DARTS and PDARTS when transferred on the ImageNet dataset are reported in Table 3. The results indicate that PDARTS discovered architectures on CIFAR-10 and CIFAR-100 that significantly outperformed those discovered

by DARTS, highlighting the ability of the discovered architectures to transfer to other datasets, such as ImageNet.

Table 3. Performance of gradient-based NAS methods when transferred on various datasets.

Works	Searched on	Transferred to	Error Rate (%)	Parameters (M)	GPU Days/Hours
[10] DARTS + Cutout	CIFAR-10	ImageNet	26.7	4.7 M	4 Days
[11] PDARTS (large) + cutout	CIFAR-10	CIFAR-100	15.27	10.5 M	0.3 Days
[11] PDARTS	CIFAR-10	ImageNet	24.4	-	0.3 Days
[11] PDARTS (large) + cutout	CIFAR-100	CIFAR-10	**2.43**	11.0 M	0.3 Days
[11] PDARTS	CIFAR-100	ImageNet	24.7	5.1 M	0.3 Days

Our comprehensive evaluation showcases the remarkable effectiveness and efficiency of gradient-based NAS methods for image classification tasks. PDARTS stands out as a powerful and transferable approach, achieving exceptional performance with significantly reduced computational time. The outstanding performance of PDARTS, especially on the CIFAR-100 dataset, highlights its potential as a highly efficient NAS solution. Its ability to achieve superior results in a fraction of the computation time (0.3 GPU days) compared to other methods positions PDARTS as a good choice for time-critical applications.

5. Future Directions

This study offers valuable insights into how gradient-based NAS methods perform and their efficiency for image classification. Based on the strengths and weaknesses discussed earlier, researchers can make informed decisions when selecting suitable NAS approaches for specific image classification tasks.

In future research endeavors, addressing the issue of aggregated redundant skip connections observed in gradient-based NAS methods emerges as a crucial focus. By formulating strategies to mitigate this aggregation and facilitate the discovery of more optimal architectural configurations, the potential for enhanced performance gains through NAS can be unlocked.

Moreover, it is observed that all the examined methods use four nodes in their cell structure. While it is a commonly used configuration, we did not encounter any explicit claims in the literature supporting its optimality. Therefore, future research can explore alternative cell structures. By carefully defining the operations and hyperparameters in the search space, NAS algorithms can be guided to explore architectures with desirable properties. PDARTS holds the potential for finding better-performing and transferable architectures. Exploring PDARTS further, the integration of attention modules in its search space can be considered to enhance its capabilities. Another challenge lies in overcoming the limitation of single-step inner optimization in gradient-based NAS methods. Although this approach improves computational efficiency, it negatively affects performance. To address this challenge, the implementation of enhanced inner optimization strategies that find a harmonious balance between efficiency and performance can be pursued, thereby enhancing the effectiveness of methods such as DARTS.

The implications of this research extend into the realm of advancing automated architectural design within deep learning. The understanding gained from various NAS methods equips researchers to craft more efficient and effective neural architectures for a wide range of applications.

6. Conclusions

This study comprehensively evaluated four major gradient-based Neural Architecture Search (NAS) methods for image classification, i.e., DARTS, PDARTS, Fair DARTS, and

Att-DARTS. The analysis encompassed a detailed comparison with methods, focusing on critical aspects such as search space, continuous relaxation strategies, and bi-level optimization. Alongside highlighting each method's strengths and weaknesses, the study extensively compared error rates, parameter counts, and computation times across datasets, including CIFAR-10, CIFAR-100, and ImageNet. Additionally, the study analyzed the cross dataset transferability capability of architectures generated by these methods by comparing their performance on various datasets such as ImageNet. The results showed that PDARTS is better and faster among the examined methods, making it a potent candidate for automating Neural Architecture Search. Future research directions stemming from the performance analysis were outlined. The benefits and implications of the study were also presented in the future directions section.

Author Contributions: Conceptualization, S.A. and M.A.W.; methodology, S.A. and M.A.W.; software, M.A.W.; validation, S.A. and M.A.W.; formal analysis, S.A. and M.A.W.; investigation, M.A.W.; resources, M.A.W.; writing—original draft preparation, S.A.; writing—review and editing S.A. and M.A.W.; visualization, S.A. and M.A.W.; supervision, M.A.W.; project administration, M.A.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: This work has been supported by Rashtriya Uchchatar Shiksha Abhiyan (RUSA) 2.0, a centrally sponsored scheme of Dept. of Higher Education, Govt. of India.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pfeifer, B.; Holzinger, A.; Schimek, M.G. Robust random forest-based all-relevant feature ranks for trustworthy ai. *Stud. Health Technol. Inform.* **2022**, *294*, 137–138. [CrossRef] [PubMed]
2. Barret, Z.; Quoc, V.L. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578. Available online: http://arxiv.org/abs/1611.01578 (accessed on 20 May 2022). [CrossRef]
3. Masanori, S.; Shinichi, S.; Tomoharu, N. A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. In Proceedings of the GECCO'17: Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017; pp. 5369–5373. [CrossRef]
4. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing Neural Network Architectures using Reinforcement Learning. *arXiv* **2016**, arXiv:1611.02167.
5. Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Tan, J.; Le, Q.V.; Kurakin, A. Large-scale evolution of image classifiers. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 2902–2911.
6. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q. Learning Transferable Architectures for Scalable Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710. [CrossRef]
7. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized Evolution for Image Classifier Architecture Search. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4780–4789. [CrossRef]
8. Mun, J.; Ha, S.; Lee, J. DE-DARTS: Neural architecture search with dynamic exploration. *ICT Express* **2023**, *9*, 379–384. [CrossRef]
9. Santra, S.; Hsieh, J.-W.; Lin, C.-F. Gradient Descent Effects on Differential Neural Architecture Search: A Survey. *IEEE Access* **2021**, *9*, 89602–89618. [CrossRef]
10. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable architecture search. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. Available online: https://openreview.net/ (accessed on 20 May 2022).
11. Chen, X.; Xie, L.; Wu, J.; Tian, Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1294–1303.
12. Chu, X.; Zhou, T.; Zhang, B.; Li, J. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. In *European Conference on Computer Vision*; Springer International Publishing: Cham, Switzerland, 2020. [CrossRef]
13. Nakai, K.; Matsubara, T.; Uehara, K. Att-DARTS: Differentiable Neural Architecture Search for Attention. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]

14. Liang, H.; Zhang, S.; Sun, J.; He, X.; Huang, W.; Zhuang, K.; Li, Z. DARTS+: Improved Differentiable Architecture Search with Early Stopping. *arXiv* **2019**, arXiv:1909.06035.
15. Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; Hutter, F. Understanding and robustifying differentiable architecture search. *arXiv* **2019**, arXiv:1909.09656. Available online: https://openreview.net/forum?id=H1gDNyrKDS (accessed on 1 June 2023).
16. Gabriel, B.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; Quoc, V.L. Understanding and Simplifying One-Shot Architecture Search. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 550–559.
17. Baymurzina, D.; Golikov, E.; Burtsev, M. A review of neural architecture search. *Neurocomputing* **2022**, *274*, 82–93. Available online: https://www.sciencedirect.com/science/article/pii/S0925231221018439 (accessed on 5 December 2022). [CrossRef]
18. Chitty-Venkata, K.T.; Emani, M.; Vishwanath, V.; Somani, A.K. Neural Architecture Search Benchmarks: Insights and Survey. *IEEE Access* **2023**, *11*, 25217–25236. [CrossRef]
19. Ren, P.; Yun, X.; Xiaojun, C.; Po-Yao, H.; Zhihui, L.; Xiaojiang, C.; Xin, W. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Comput. Surv.* **2021**, *54*, 1–34. [CrossRef]
20. Mobeen, A.; Muhammad, A.; Dongil, H. A Novel Encoding Scheme for Complex Neural Architecture Search. In Proceedings of the 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Jeju Island, Republic of Korea, 23–26 June 2019; pp. 1–4. [CrossRef]
21. Elsken, T.; Metzen, J.H.; Hutter, F. Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. *arXiv* **2018**, arXiv:1804.09081.
22. Zhong, Z.; Yang, Z.; Deng, B.; Yan, J.; Wu, W.; Shao, J.; Liu, C.-L. BlockQNN: Efficient Block-Wise Neural Network Architecture Generation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 2314–2328. [CrossRef]
23. Jaafra, Y.; Laurent, J.L.; Deruyver, A.; Naceur, M.S. Reinforcement learning for neural architecture search: A review. *Image Vis. Comput.* **2019**, *89*, 57–66. Available online: https://www.sciencedirect.com/science/article/pii/S0262885619300885 (accessed on 17 May 2022). [CrossRef]
24. Kaiming, H.; Xiangyu, Z.; Shaoqing, R.; Jian, S. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778. [CrossRef]
25. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
26. Ali, S.; Wani, M.A. Recent Trends in Neural Architecture Search Systems. In Proceedings of the 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), Nassau, Bahamas, 12–14 December 2022; pp. 1783–1790. [CrossRef]