D-DARTS: Distributed Differentiable Architecture Search

Alexandre Heuillet, Hedi Tabia, Hichem Arioui, Kamal Youcef-Toumi

D-DARTS: Distributed Differentiable Architecture Search

Alexandre Heuillet[a,*], Hedi Tabia[a], Hichem Arioui[a] and Kamal Youcef-Toumi[b]

[a]IBISC, Univ Evry, Université Paris-Saclay, 36 rue du Pelvoux, Evry-Courcouronnes, 91000, France
[b]MIT, 77 Massachusetts Avenue, Cambridge, 02139, MA, USA

ARTICLE INFO

Keywords:
Neural Architecture Search
Differentiable Architecture Search
Deep Learning
Computer Vision

ABSTRACT

Differentiable ARchiTecture Search (DARTS) is one of the most trending Neural Architecture Search (NAS) methods. It drastically reduces search cost by resorting to weight-sharing. However, this approach also dramatically reduces the search space, thus excluding potential promising architectures. In this article, we propose D-DARTS, a solution that addresses this problem by nesting neural networks at the cell level instead of using weight-sharing to produce more diversified and specialized architectures. Moreover, we introduce a novel algorithm that can derive deeper architectures from a few trained cells, increasing performance and saving computation time. In addition, we introduce DARTOpti, an alternative search space in which we optimize existing handcrafted architectures instead of searching from scratch. Our solution reaches competitive performance on image classification tasks.

1. Introduction

With the field of Deep Learning (DL) getting more and more attention over the past few years, a significant focus has been set on neural network architectural conception. A large number of architectures have been manually crafted to address different computer vision problems [1, 2, 3]. However, the design of these human-made architectures is mainly driven by intuition and lacks the certainty of an optimal solution. This is due to the many combinations needed to build a relevant neural architecture, making the search space difficult to browse manually. Neural Architecture Search (NAS) works [4, 5, 6, 7, 8] tried to tackle this issue by automatizing the architecture design process. In NAS, a search algorithm attempts to build a neural network architecture from a defined search space by asserting the performance of "candidate" architectures. The most trending NAS approach is currently Differentiable ARchitecTure Search (DARTS) [5], whose main advantage is a greatly reduced search cost compared to earlier approaches [7, 9] thanks to its 2-cell search space. However, searching for only two types of cells significantly restricts the search space and limits the diversity of candidate architectures. To alleviate this problem, we propose to directly search for a complete network with individualized cells. This network delegates the search process to the cell level in a distributed fashion. Moreover, we introduce a loss function based on Shapley values [10] to guide cell optimization. Consequently, this distributed structure makes it possible to encode existing architectures (e.g., ResNet50 [1]) in the search space and use them as starting points for the search process.

As presented in Section 4, **the contributions** of this article are: (i) a new way of structuring DARTS' search process by nesting small neural networks in cells to individualize them, (ii) a novel Shapley value-based loss function specially designed to take advantage of the new distributed structure, (iii) a search space augmented with additional operations to allow the encoding of handcrafted architectures. The rest of the article is structured as follows: In Section 3, we review the original concept of DARTS and discuss its issues. In Section 4, we detail the proposed method. Section 5 presents the results of a set of experiments conducted on popular computer vision datasets. Section 6 discusses the results of the experimental study. Finally, Section 7 brings a conclusion to this article while giving some insights on promising directions for future work.

2. Related Work

Here, we focus on a few recent works which attempted to improve DARTS by addressing its limitations (see Section 3). For a more comprehensive survey on differentiable NAS, we defer the reader to [11]. PC-DARTS [12] minimized the memory footprint by optimizing the search process to avoid redundancy. P-DARTS [13] greatly reduced the search time by progressively deepening the architecture when searching, leading to a better search space approximation and regularization. FairDARTS [6] used the *sigmoid* function to discretize architectures and introduced a novel loss function (see Section 3). DARTS- [14] introduced an auxiliary skip connection with a β-decay factor to mitigate the importance of this operation. β-DARTS [15] improved DARTS- with a new regularization method that can prevent the architectural parameters from saturating. This led to increased robustness and better generalization ability. C-DARTS [16] leveraged a cyclic feedback mechanism between the search (*supernet*) and evaluation networks in order to bridge the optimization gap by jointly optimizing these two networks. U-DARTS [17] redefined the relations between cells in order to browse a uniform search space. Finally, DOTS [18] proposed to decouple the operation and topology search, so that the cell topology is no longer constrained by the operation weights. Therefore, pairwise edge combinations are attributed weights that are updated

*Corresponding author
alexandre.heuillet@univ-evry.fr (A. Heuillet); hedi.tabia@univ-evry.fr (H. Tabia); hichem.arioui@univ-evry.fr (H. Arioui); youcef@mit.edu (K. Youcef-Toumi)
ORCID(s): 0000-0003-2109-7895 (A. Heuillet)

separately from the operation weights. Despite these improvements, these follow-up methods did not address one of DARTS' most critical limitations: its restricted search space. This is the issue we are focusing on in this article.

3. Preliminaries: DARTS

Differentiable ARchitecTure Search (DARTS) [5] is a gradient-based NAS method that searches for novel architectures through a cell-modulated search space while using a weight-sharing mechanism to speed up this process. More specifically, it searches for two different types of cells: *normal* and *reduction*. These two cells are the building blocks from which architectures of any size can be derived. Thus, most of the final network components share the same architectural weights. Each cell can be described as a direct acyclic graph of N nodes where each edge connecting two nodes is a mix of operations chosen among $|O_{i,j}| = K$ candidates, where $O_{i,j} = \{o^1_{i,j}, ..., o^K_{i,j}\}$ represents the set of all possible operations for the edge $e_{i,j}$ connecting node i to node j. DARTS browses a search space S comprising $K = 7$ operations (*skip_connect*, *max_pool_3x3*, *avg_pool_3x3*, *sep_conv_3x3*, *sep_conv_5x5*, *dil_conv_3x3* and *dil_conv_5x5*). The goal of the DARTS process is to determine which incoming operations (with a maximum of 2) must be selected for each node to maximize the validation loss L_{CE} (Cross-Entropy loss). In that objective, DARTS' search process involves learning a set of parameters, denoted by $\alpha_{i,j} = \{\alpha^1_{i,j}, ..., \alpha^K_{i,j}\}$, representing the weight of each operation from $O_{i,j}$ in the mixed output of each edge. To build the mixed output, the categorical choice of operations is done through a *softmax*:

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{K} \frac{exp(\alpha^k_{i,j})}{\sum_{k'=1}^{K} exp(\alpha^{k'}_{i,j})} o^k_{i,j}(x) \tag{1}$$

where $\overline{o}_{i,j}(x)$ is the mixed output of edge $e_{i,j}$ for input feature x and $\alpha^k_{i,j} \in \alpha_{i,j}$ is the weight associated with operation $o^k_{i,j} \in O_{i,j}$. These architectural parameters are optimized simultaneously as the global supernet weights. Thus, DARTS is practically solving a bi-level optimization problem.

However, DARTS suffers from two major issues. The first is the over-representation of *skip connections*. The second is the discretization discrepancy problem of the *softmax* operation in Eq. (1), namely a very small standard deviation of the resulting probability distribution. To mitigate these issues, the authors of FairDARTS [6] replaced the *softmax* function with the *sigmoid* function (denoted by σ) :

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{K} \sigma(\alpha^k_{i,j}) o^k_{i,j}(x) = \sum_{k=1}^{K} \frac{1}{1 + exp(-\alpha^k_{i,j})} o^k_{i,j}(x). \tag{2}$$

They also proposed a novel loss function (*zero-one loss* denoted by L_{01}) which aims to push the architectural weight values towards 0 or 1, and is defined as follows:

$$L_{01} = -\frac{1}{|\alpha|} \sum_{i=1}^{|\alpha|} (\sigma(\alpha_i) - 0.5)^2, \tag{3}$$

where $|\alpha|$ is the cell's total number of architectural weights. In fact, L_{01} corresponds to the mean square error between $\sigma(\alpha_i)$ and 0.5. L_{01} is then added to L_{CE} to form FairDARTS total loss L_F:

$$L_F = L_{CE} + w_{0-1} L_{0-1} \tag{4}$$

where w_{0-1} is a coefficient weighting L_{0-1}. Despite these solutions, DARTS and all of its evolutions [13, 6, 5, 12, 15, 18] are still limited in their capacity to create original architectures since most of their structure is rigid and human-made. The search space is also very restricted as pointed out by prior works [19, 20], with only 2 types of cells. This is the issue we are trying to address in this article. To the best of our knowledge, our method is the first to explore and implement distributed differential neural architectural search.

4. Proposed Approach

In this section, we present the main contributions of our article: a novel distributed differentiable NAS approach with a Shapley value based loss, and a method to optimize existing handcrafted architectures using differentiable NAS.

4.1. Delegating Search to Cell-Level Subnets

Our method's key idea is to increase architecture diversity by delegating the search process to subnets nested in each cell. It is itself a full neural network with its own optimizer, criterion, scheduler, input, hidden, and output layers, as shown in Fig. 1. This way, each cell that composes the global supernet is individual. Thus, instead of searching for building blocks as DARTS [5] do, we increase the number of searched cells to an arbitrary n and directly seek for a full n-layer convolutional neural network where each cell is highly specialized (contrary to generic building blocks). Thus, we trade the weight-sharing process introduced in DARTS for greater flexibility and creativity. Nonetheless, cells still belong either to the *normal* or *reduction* class, depending on their position in the network.

As explained in section 3, DARTS only searches for two types of cells and stacks them multiple times to form a network as deep as needed. This weight-sharing process has the advantage of reducing search space to a limited set of parameters (i.e., α_{normal} and α_{reduce}), thus saving time and hardware resources. However, this approach limits both the search space size and the originality of the derived architectures as all the underlying structure is human-designed. In particular, the search space size $s(K, N)$ of a single cell with K primitive operations to select from (within a maximum of 2 from different incoming edges) and N steps (i.e., intermediary nodes) can be computed as follows:

$$s(K, N) = \prod_{i=1}^{N} \frac{(i+1)i}{2} K^2 \tag{5}$$

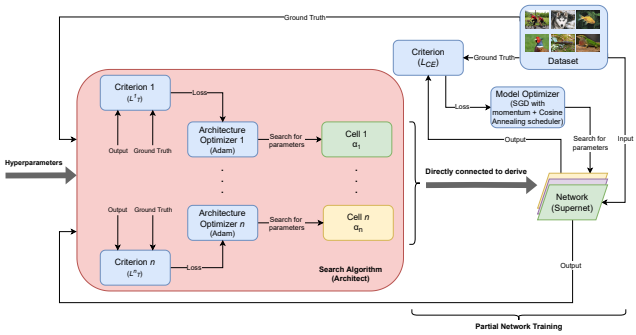

Figure 1: Layout of the search process used in D-DARTS. Each cell i is independent with its own optimizer, scheduler and criterion (our proposed ablation loss L_T). Each cell searches for its architectural parameters α_i, which makes the entire search process distributed. The searched cells are directly connected to each other to build a supernet trained to validate their performance.

Following Eq. (5), using DARTS default parameters ($K = 7$ and $N = 4$), the search space of a single cell comprises around 10^9 possible configurations. Thus, as both *normal* and *reduction* cells share the same K and N, the total search space size of DARTS is around 10^{18} possibilities. This number is comparable to other differentiable NAS works [21, 20], but far lower than those of Reinforcement Learning based NAS methods [22, 7] that describe architecture topologies using sequential layer-wise operations, which are also far less efficient. Our approach, dubbed *D-DARTS*, effectively expands the search space by a factor of $10^{(c-2)*9}$ (according to Eq. (5)) where c is the total number of searched cells. Thus, the total size of D-DARTS search reaches around 10^{72} when considering $c = 8$. In Section 5, we show that smaller (e.g., 4 or 8 layers) D-DARTS architectures can achieve similar or higher performance than larger (e.g., 14 or 20 layers) architectures on popular datasets.

4.2. Adding a New Cell-Specific Loss

In addition to the new network structure introduced in Section 4.1, we designed a novel cell-specific loss function that we dubbed ablation loss (denoted L_{AB}). Indeed, as we increased the number of searched cells, the learning challenge became greater with a large amount of additional parameters to consider. Thus, the global loss functions used in DARTS [5] and FairDARTS [6] cannot accurately assess the performance of each cell and instead only take into account the global performance of the supernet. In contrast, our new loss function is specific to each cell. It is an additive loss, based on the global loss function L_F introduced in [6] that proved to be a significant improvement over the original one [5]. The main idea behind this ablation loss function is to perform a limited ablation study on the cell level. This is done by measuring the performance of the network with/without each cell. Ablation studies have long proven to hold a key role in asserting the effectiveness of neural network architectures [23]. This way, by computing the difference in the supernet loss L_{CE} with and without each individual cell activated, we can obtain a measure of their respective contributions that we call their marginal contributions, labeled $M_C = \{M_C^1, ..., M_C^n\}$ for an n-cell network. This method is based on Shapley values [10], a game theory technique widely used in Explainable Artificial Intelligence to assess the contributions of model features to the final output [24, 25]. Thus, cell C_i marginal contribution M_C^i is computed as follows:

$$M_C^i = L_{CE}^{(C)} - L_{CE}^{(C \setminus \{C_i\})}, \tag{6}$$

where C is the set containing all cells such as $C = \{C_1, ..., C_n\}$. Once we obtained all the marginal contributions M_C, we apply the following formula to compute the ablation loss L_{AB}^i of cell C_i:

$$L_{AB}^i = \begin{cases} \frac{M_C^i - mean(M_C)}{mean(M_C)} & \text{if } mean(M_C) \neq 0 \\ 0 & \text{else} \end{cases} \tag{7}$$

L_{AB}^i expresses how important the marginal contribution of cell i is w.r.t. the mean of all the marginal contributions. Finally, L_{AB}^i is then added to FairDARTS global loss L_F (see Eq.(4)) to form the total loss L_T^i, weighted by the hyperparameter w_{AB}:

$$L_T^i = L_F + w_{AB} L_{AB}^i \tag{8}$$

In Section 5, we show that L_T^i can warm start the search process and substantially increase performance.

4.3. Building Larger Networks from a Few Highly Specialized Cells

As presented in Section 4.1, we directly search for a "full" network of multiple individual cells instead of searching for building block cells as in DARTS [5]. But, the downside of this method is that searching for many cells is memory-hungry, as each cell must possess its own optimizer, criterion, and parameters. Thus, using a larger number of cells without spending additional search time may be useful, especially when dealing with highly complex datasets such as ImageNet [26]. To this end, we developed a new algorithm to derive larger architectures from an already searched smaller one, inspired by what is done in DARTS [5]. The key idea behind this concept is to keep the global layout of the smaller architecture with the reduction cells positioned at the 1/3 and 2/3 of the network, similarly as in DARTS and FairDARTS [6], and repeat the searched structure of "normal" cells in the intervals between the reduction cells until we obtain the desired number of cells. This way, we can obtain a larger architecture without launching a new search (i.e., without any overhead). This process is summarized in Algorithm 1.

4.4. Encoding Handcrafted Architectures in DARTS (DARTOpti)

Unlike the original DARTS [5] or one of its derivatives [13, 15, 18], D-DARTS individualizes each cell and makes

Algorithm 1 Algorithm describing the larger architecture derivation process for D-DARTS

Require: List: C, list of searched cells
Require: Integer: n, desired number of cells
Ensure: List: C_f, list of cells that compose the derived architecture
$C_f \leftarrow$ empty_list()
$m \leftarrow$ euclidean_division($|C|, 3$)
$m2 \leftarrow$ euclidean_division($2 \times |C|, 3$)
for i in $[0, n]$ **do**
  if $n > |C|$ **then**
    if $i <$ euclidean_division($n, 3$) **then**
      $c \leftarrow$ modulo(i, m)
    else if $i =$ euclidean_division($n, 3$) **then**
      $c \leftarrow m$
    else if $i >$ euclidean_division($n, 3$) **and** $i <$ euclidean_division($2 \times n, 3$) **then**
      $c \leftarrow$ modulo($i, m2 - 1 - m$) $+ m + 1$
    else if $i =$ euclidean_division($2 \times n, 3$) **then**
      $c \leftarrow m2$
    else
      $c \leftarrow$ modulo($i, |C| - 1 - m2$) $+ m2 + 1$
    end if
  else
    $c \leftarrow i$
  end if
  append(c, C_f)
end for

it possible to encode large handcrafted architectures [27, 1], which typically possess multiple types of layers (e.g., 13 for Xception [27]). The primary motivation behind this is to use existing architectures as initial points for the optimization process. Since these handcrafted architectures have been optimized carefully, they might be considered local minima of the search space. This process involves a few key procedures. These procedures are (i) encoding the handcrafted architecture in D-DARTS' cell system, (ii) a new weight-sharing mechanism, (iii) warm-starting the model, and (iv) designing a new search space S_o.

(i) The architecture is manually encoded as a D-DARTS-compatible *genotype* (i.e., a data structure that describes each cell design, the location of reduction cells in the architecture, and the maximum number of steps in a cell). Then, when searching from this architecture or training it, the corresponding *genotype* is automatically loaded and deserialized into α weights (see Section 3), which can be optimized by D-DARTS' search process.

(ii) A weight-sharing mechanism is introduced to reduce the search cost and redundancy in cells (especially when starting from architectures with many layers). Every identical cell in the baseline architecture will share the same weights. For instance, Xception [27] is composed of 13 cells but only 5 of those are different. Thus, in this case, the number of optimizers will be reduced from 13 to 5.

(iii) The supernet is warm-started for 5 epochs. This happens before the actual search starts to let the performance of the baseline architecture be assessed and taken into account by the search algorithm.

(iv) 5 new operations are added to DARTS' original search space S: (1) `conv_3x1_1x3`, (2) `conv_7x1_1x7`, (3) `simple_conv_1x1`, (4) `simple_conv_3x3`, and (5) `bottleneck_1x3x1`. This brings the total number of operations to 12, and unlocks new possibilities but at the cost of further increasing D-DARTS' already large search space. This new search space is denoted S_o.

We show in Section 5 that this process, denoted DARTOpti, can successfully optimize handcrafted architectures.

5. Experiments

We conducted image classification experiments on various datasets including CIFAR-10 and CIFAR-100 [28], ImageNet [26], MS-COCO [29], and Cityscapes [30]. These datasets are well-known and widely used in the computer vision and pattern recognition community.

5.1. Experimental Settings

Image classification tasks were evaluated on CIFAR-10, CIFAR-100 [28] and ImageNet [26] datasets. All experiments were conducted using Nvidia GeForce RTX 3090 and Tesla V100 GPUs. We mostly used the same data processing, hyperparameters, and training tricks as in FairDARTS [6] and DARTS [5]. We searched for 8-cell networks and used Algorithm 1 to derive 14-cell networks. We set the batch size to 128 when searching on CIFAR-10/100 and to 96 when searching on ImageNet. When starting from an existing architecture, the number of initial channels is increased to 64 when training to match the designs of the baseline architectures. However, this makes the average number of parameters of DARTOpti architectures significantly more important than D-DARTS' ones (see Tables 1, 2 and 3). Hence, we reduce the number of channels to 32 while searching to save memory. Naturally, we searched for networks whose number of cells matches the number of different layers in the original architecture (e.g., 4 in ResNet50 [1]). Finally, we select the architectural operations using FairDARTS *edge* (i.e., 2 operations maximum per edge) or *sparse* (i.e., 1 operation maximum per edge) method with a threshold of 0.85. We chose $w_{01} = 8$ and $w_{abl} = 0.5$ for the hyperparameters of total loss L_T (see Eq. (8)) as discussed in Section 5.2. DARTS' parsing method is referred to as *darts* in Table 1 and Table 3. Our implementation is based on PyTorch 1.10.2 and is derived from [5, 6]. Code and pretrained models can be accessed at `https://github.com/aheuillet/D-DARTS`.

5.2. Analysis of the Ablation Loss L_{AB}

5.2.1. *Hyperparameter Choice*

We made the hyperparameter weights w_{abl} and w_{01} from Eq. (8) vary to choose their optimal value w.r.t. the global loss. Thus, in Fig. 5.2.1 we made w_{abl} vary from 0 to 2 while keeping L_{01} deactivated (i.e., $w_{01} = 0$) in order to analyze its impact. We can observe that an optimal value seems to be attained around $w_{abl} = 0.5$, with the global loss mainly increasing when w_{abl} reaches higher or lower values.

Moreover, we made the value of w_{01} vary during search on CIFAR-100, with $w_{abl} = 0.5$ fixed, and reported the number of dominant operations (i.e., operations with $\sigma(\alpha) > 0.9$). This experiment was conducted to select a relevant value for w_{01} since the one used in FairDARTS [6] ($w_{01} =$

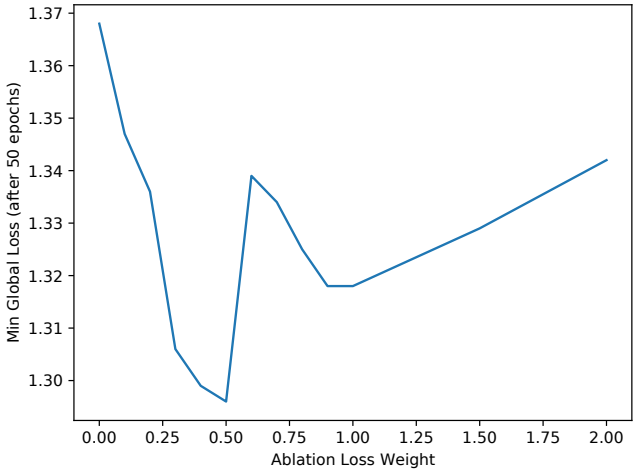

Figure 2: Line plot of the minimal global loss obtained by searching for a model on CIFAR-100 [28] for 50 epochs w.r.t. the sensitivity weight w_{abl} used for the ablation loss L_{AB}. We deactivated L_{01} (i.e., $w_{01} = 0$) to prevent interference from occurring.

10) is no longer valid as we altered the search process. Fig. 5.2.1 shows that the proportion of dominant operations steadily increases from $w_{01} = 0$ to $w_{01} = 5$ where it reaches a plateau and stabilizes. It is worth noting that for $w_{01} = 5$ and higher, nearly all sigmoid values $\sigma(\alpha)$ are either greater than 0.9 or inferior to 0.1. Finally, we chose $w_{01} = 7$ as it offers both a high number of dominant operations and an equilibrium between operations with $\sigma(\alpha) > 0.9$ and those with $\sigma(\alpha) < 0.1$.

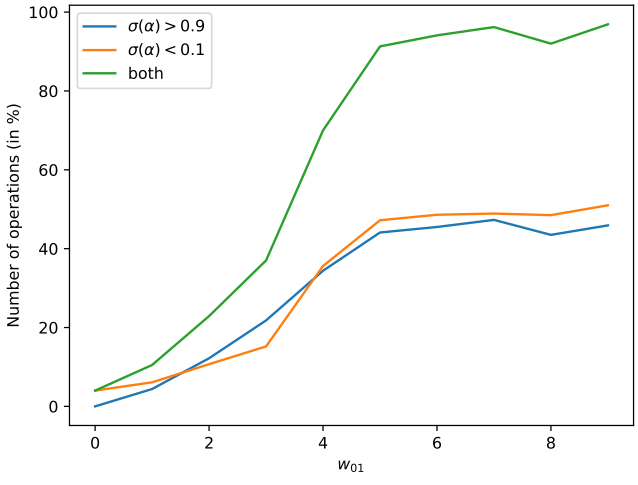

Figure 3: Line plot showing the percentage of dominant operations obtained in the final architecture α while searching on CIFAR-100 w.r.t. the sensitivity weight w_{01} used for L_{01}. We can see that the proportion of both types of operations stabilizes after $w_{01} = 5$ and reaches an equilibrium at $w_{01} = 7$.

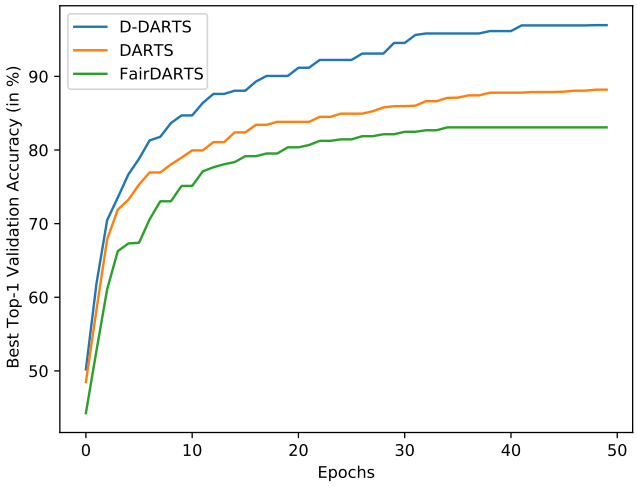

Figure 4: Line plot showing the best validation top-1 accuracy while searching on CIFAR-10[28] w.r.t. the current epoch. D-DARTS clearly outperforms both DARTS and FairDARTS by a large margin.

5.2.2. Ablation Study

We conducted an ablation study on our proposed ablation loss L_T of Eq. (8). In particular, we compared the performance of architectures with similar characteristics searched either with L_T or L_F (see Eq. (4)). Tables 1 and 2 show that L_T-searched architectures (DD-1, DD-3, DD-4, DD-5) outperform their L_F-searched counterparts by an average of 0.6 % across all datasets, confirming the advantage procured by this new loss function. Concretely, when considering the 14-cell *edge* parsed models evaluated on CIFAR-10, DD-3 reached a top-1 accuracy of 97.58 %, thus outperforming L_F-searched DD-2 by 0.48 %. Moreover, it is worth noting that L_T-searched DD-1 (8-cell model) reached a similar score as DD-2 (around 97.1 %), despite featuring significantly fewer parameters (1.7M vs. 3.3M). One additional point is that L_T seems to provide a larger increase in performance for CIFAR-100. For example, the gain in performance is around 1 % between the 8-cell versions of DD-6 and DD-4. However, when considering models that leveraged Algorithm 1 to increase their number of cells (e.g., DD-4 and DD-6), the performance gain is limited (e.g., around 0.1 % on CIFAR-100). This could be due to Algorithm 1 that may have a leveling effect by disturbing the cell sequence and increasing the number of model parameters.

5.2.3. Convergence Speed

We conducted an experiment on the search process convergence speed of our method D-DARTS compared to previous baselines [5, 6]. Fig. 4 shows a plot of the best validation top-1 accuracy w.r.t. the number of epochs. One can notice that D-DARTS converges very quickly and faster than the others. D-DARTS outperforms both DARTS and FairDARTS respectively by 9% and 14%.

Table 1
Comparison of models on CIFAR-10 [28]. Each reported Top-1 accuracy is the best of 4 independent runs. For previous baselines, results are the official numbers from their respective articles. The search cost is expressed in GPU days. All models have been searched on CIFAR-10 except for ⋄ which have been searched on CIFAR-100.

Models		Params (M)	Loss	Top-1 (%)	Layers	Cost
DARTS[5]		3.3	L_{CE}	97.00	20	1.5
PC-DARTS[12]		3.6	L_{CE}	97.43	20	3.8
P-DARTS[13]		3.4	L_{CE}	97.50	20	**0.3**
FairDARTS-a[6]		2.8	L_F	97.46	20	0.4
C-DARTS[16]		3.9	L_{CE}	97.52	20	**0.3**
U-DARTS[17]		3.3	L_{CE}	97.41	20	N.D.
DOTS[18]		3.5	L_{CE}	97.51	20	**0.3**
DARTS-[14]		3.5	L_{CE}	97.41	20	0.4
β-DARTS[15]		3.75	L_{CE}	97.47	20	0.4
Ours	DD-1	**1.7**	L_T	97.02	8	0.5
	DD-2	3.3	L_F	97.10	**8**	0.5
	DD-3	6.55	L_T	97.58	14	0.5
	DD-4⋄	3.9	L_T	97.48	8	0.5
	DD-4⋄	7.6	L_T	**97.75**	14	0.5

Table 2
Comparison of models on CIFAR-100 [28]. Each reported Top-1 accuracy is the best of 4 independent runs. For previous baselines, results are the official numbers from their respective articles. The search cost is expressed in GPU days. All models have been searched on CIFAR-100 except for ⋄ which have been searched on CIFAR-10.

Models		Params (M)	Loss	Top-1 (%)	Layers	Cost
DARTS[5]		3.3	L_{CE}	82.34	20	1.5
P-DARTS[13]		3.6	L_{CE}	84.08	20	**0.3**
FairDARTS[6]		3.5	L_F	83.80	20	0.4
DOTS[18]		4.1	L_{CE}	83.52	20	**0.3**
DARTS-[14]⋄		3.4	L_{CE}	82.49	20	0.4
β-DARTS[15]⋄		3.83	L_{CE}	83.48	20	0.4
Ours	DD-1⋄	**1.7**	L_T	81.10	8	0.5
	DD-4	3.9	L_T	83.86	8	0.5
	DD-4	7.6	L_T	**84.15**	14	0.5
	DD-5	**1.7**	L_T	81.92	8	0.5
	DD-6	3.3	L_F	82.90	**8**	0.5
	DD-6	6.1	L_F	84.06	14	0.5

5.3. Searching Architectures on CIFAR

On CIFAR-10/100 [28], we searched for several 8-layer models arbitrarily dubbed DD-1, DD-2, DD-3, DD-4, DD-5 and DD-6. These models were searched using varying hyperparameters such as using loss L_T or L_F, or using the *sparse* or *threshold* parsing method (presented in Section 5.1). We used Algorithm 1 to increase the depth to 14 layers to test how smaller architectures compete with larger ones. All results are presented in Tables 1 and 2. Overall, D-DARTS models reach competitive results in both datasets. The smaller ones, such as DD-1 or DD-5, can match the performance of previous baselines despite possessing fewer parameters (e.g., 1.7M against 2.8M for the smallest model of [6]), although the largest achieve better results (e.g., 84.15% top-1 accuracy for DD-4 on CIFAR-100). Moreover, Algorithm 1 effectively provides a performance gain in both datasets (e.g., around 0.3% for DD-4 when using 14 cells instead of 8), thus asserting its usefulness. This impact is more important on CIFAR-100 (around 2%). Hence, deeper architectures might be less relevant on simpler datasets such as CIFAR-10, where *sparse* models already achieve very high top-1 scores (greater or equal to 97%), than with more challenging datasets like CIFAR-100 or ImageNet [26]. We also compared the performance of models using FairDARTS [6] loss function L_F with ones using our new ablation-based loss function L_T to assert its effectiveness (see Section 5.2 for details).

5.4. Searching and Transferring to ImageNet

To test our approaches on a more challenging dataset, we transferred our best models searched on CIFAR-100 [28] to ImageNet [26]. We also searched directly on ImageNet using the same training tricks and hyperparameters as the authors of DARTS [5]. Table 3 shows that model DD-7 (searched directly on ImageNet) reached a top-1 accuracy of 75.5 %, outperforming PC-DARTS [12] and P-DARTS [13] by 0.6 %. It is important to note that all of these approaches (and ours) use DARTS search space $\mathcal{S}$ while FairDARTS [6] uses its own custom search space (mainly composed of inverted bottlenecks) as they argue that $\mathcal{S}$ is too limited for ImageNet. Nevertheless, DD-7 still reached a near-identical score as FairDARTS-D despite using this simpler search space. In addition, the DARTOpti versions of ResNet18 and ResNet50 reached a competitive top-1 accuracy of respectively 77.0 % and 76.3 %. They critically improve on the original architectures, increasing top-1 accuracy by an average of 5.1 %. Notably, DO-2-ResNet50 achieves the same score as FBNetV2 [20] while requiring nearly a hundred times less search cost (i.e., 0.3 GPU days versus 25 GPU days) and not even having been searched directly on ImageNet but instead transferred from CIFAR-100. Finally, a notable gap (around 0.5 %) between DD-4 (transferred from CIFAR-100) and DD-7 shows that searching directly on ImageNet significantly impacts performance.

5.5. Detecting objects on MS-COCO and Instance Segmentation on Cityscapes

We transferred our best model trained on ImageNet (DO-2-ResNet18) to MS-COCO [29] in order to test our approach on tasks other than image classification. We used our model as the backbone of RetinaNet [31] and fine-tuned for 12 epochs, similarly to FairDARTS [6]. Table 4 shows that our approach reached a box AP score of 34.2 % hence outperforming DARTS- [14], FairDARTS [6] and the original ResNet18 [1]. We also performed instance segmentation on Cityscapes [30], a dataset that focuses on semantic understanding of street scenes. We used DO-2-ResNet18 as the backbone of Mask R-CNN [32] and compared it against other baselines (DARTS, FairDARTS, ResNet18). Table 5 features results similar to MS-COCO, with D-DARTS outperforming previous approaches. This way, the performance advantage of D-DARTS is confirmed when transferring to computer vision tasks other than image classification.

Table 3
Comparison of models on ImageNet [26]. For previous baselines, the results are the official numbers from their respective articles. The search cost is expressed in GPU days. †: Our implementation in search space S_o, results might vary from the official one.

Models		Params (M)	+× (M)	Parsing Method	Loss	Top-1 (%)	Layers	Cost	Search Space	Searched On
FBNetV2-L1[20]		8.49	326	N.A.	N.A.	**77.0**	N.A.	25	custom	ImageNet
DARTS[5]		4.7	574	*darts*	L_{CE}	73.3	14	4	S	CIFAR-100
PC-DARTS[12]		5.3	586	*darts*	L_{CE}	75.8	14	3.8	S	ImageNet
P-DARTS[13]		5.1	577	*darts*	L_{CE}	75.9	14	**0.3**	S	ImageNet
FairDARTS-D[6]		4.3	440	*sparse*	L_F	75.6	20	3	custom	ImageNet
DOTS[18]		5.3	596	*sparse*	L_{CE}	76.0	20	1.3	S	ImageNet
DARTS-[14]		4.9	467	*sparse*	L_{CE}	76.2	20	4.5	S	ImageNet
C-DARTS[16]		6.1	701	*darts*	L_{CE}	76.3	14	1.7	S	ImageNet
U-DARTS[17]		4.9	N.D.	*darts*	L_{CE}	73.78	14	N.D.	S	ImageNet
β-DARTS[15]		5.4	597	*sparse*	L_{CE}	75.8	20	0.4	S	CIFAR-100
ResNet18[1]†		14.17	2720	N.A.	N.A.	69.2	4	N.A.	N.A.	N.A.
ResNet50[1]†		24.36	4715	N.A.	N.A.	73.9	4	N.A.	N.A.	N.A.
Xception[27]†		14.7	31865	N.A.	N.A.	74.1	13	N.A.	N.A.	N.A.
Ours	DD-4	7.6	617	*sparse*	L_T	75.0	14	0.5	S	CIFAR-100
	DD-7	6.4	828	*edge*	L_T	75.5	8	3	S	ImageNet
	DO-2-ResNet18	53.4	8619	*sparse*	L_T	**77.0**	**4**	**0.3**	S_o	CIFAR-100
	DO-2-ResNet50	73.23	10029	*sparse*	L_T	76.3	**4**	**0.3**	S_o	CIFAR-100

Table 4
Comparison of backbone models for RetinaNet [31] on MS-COCO [29].

Models	AP (%)	AP_{50} (%)	AP_{75}(%)	AP_s(%)	AP_m(%)	AP_l(%)
FairDARTS[6]	31.9	51.9	33.0	17.4	35.3	43.0
DARTS-[14]	32.5	**52.8**	34.1	18.0	36.1	43.4
ResNet18[1]	31.7	49.6	33.4	16.2	34.2	43.0
DO-2-ResNet18 (Ours)	**34.2**	52.1	**36.6**	**19.1**	**38.3**	**45.3**

6. Discussion

In Section 5, we showed that our proposed concepts perform well but are not exempt from limitations. Increasing the search space size to such an extent (e.g., 10^{72} with 8 cells) makes the optimization process significantly more challenging. However, if we consider Fig. 4, we can observe that D-DARTS outperforms both DARTS and FairDARTS during the search phase. Hence, expanding the search space provides benefits that far outweigh the increase in optimization difficulty. In addition, our novel ablation-based loss L_{AB} (see Eq. 8) aims to enhance the optimization process by obtaining finer information (i.e., cell-specific) than the global loss L_{CE}. Algorithm 1 also helps to reduce this optimization issue by allowing us to search on a small proxy network before expanding it to a larger one in the training phase. Nevertheless, this optimization issue could potentially be further relieved by, for instance, enhancing cell optimizers. This could be the subject of future work. Moreover, we show that despite using a data-dependent search process, following the practice of all DARTS-based previous works [5, 6, 14, 15], that should decrease robustness w.r.t. other datasets our D-DARTS models are still generalizable to other tasks. In fact, transfer learning experiments show that models searched on the small CIFAR-100 still achieve competitive performance on the large-scale ImageNet dataset (see Table 3). Furthermore, we proved that DARTOpti could successfully optimize top-performing handcrafted architectures such as ResNet50 [1] with a significant gain in performance (i.e., a 3.9 % average increase in top-1 accuracy across all datasets). However, this approach also has its limitations as the optimization process becomes more challenging when the architecture has many cells (e.g., Xception [27] with 13 cells). This translates to an increased search cost and limited performance gains, as the standard 50 search epochs may not be enough for architectures of that size. Nonetheless, combining the *sparse* parsing method with our distributed design

Table 5
Comparison of backbone models for Mask R-CNN [32] on Cityscapes [30].

Models	AP(%)	AP_{50}(%)	AP_{75}(%)	AP_s(%)	AP_m(%)	AP_l(%)
FairDARTS[6]	41.1	69.3	N.A.	18.9	42.4	61.8
DARTS-[14]	41.7	**70.4**	N.A.	19.5	43.4	62.3
ResNet18[1]	40.9	66.2	N.A.	17.6	41.1	61.8
DO-2-ResNet18 (Ours)	**44.0**	69.6	N.A.	**20.2**	**46.0**	**65.1**

allowed us to discover unprecedentedly small architectures (around 1.7 M for the tiniest) that can still yield competitive results. In addition, while using the *edge* parsing method, it is possible to search for larger-size models that reach state-of-the-art results. This demonstrates the flexibility and usefulness of our novel approach.

7. Conclusion and Future Work

In this article, we proposed a novel paradigm for DARTS [5] with individualized cells. We showed that it effectively achieves competitive results on popular image classification datasets. We also demonstrated that this approach could successfully improve the design of existing top-performing handcrafted deep neural network architectures [1, 27]. In addition, our novel architecture derivation algorithm allows us to build larger architectures from only a small number of cells without further training. Finally, many ideas around this approach have not been explored yet, such as the automatic selection of search hyperparameters or enhancing cell optimizers. These could be the subject of future work.

Acknowledgements

This work was performed using HPC resources from GENCI-IDRIS (Grant 20XX-AD011012644).

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[3] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: ICLR, 2015.

[4] A. Heuillet, H. Tabia, H. Arioui, Nasiam: Efficient representation learning using neural architecture search for siamese networks, in: The first INNS DLIA Workshop, IJCNN, 2023.

[5] H. Liu, K. Simonyan, Y. Yang, DARTS: Differentiable architecture search, in: ICLR, 2019.

[6] X. Chu, T. Zhou, B. Zhang, J. Li, Fair darts: Eliminating unfair advantages in differentiable architecture search, in: ECCV, Springer, 2020, pp. 465–480.

[7] B. Zoph, Q. V. Le, Neural architecture search with reinforcement learning, in: ICLR, 2017.

[8] S. Doveh, E. Schwartz, C. Xue, R. Feris, A. Bronstein, R. Giryes, L. Karlinsky, Metadapt: meta-learned task-adaptive architecture for few-shot classification, Pattern Recognition Letters 149 (2021) 130–136.

[9] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning transferable architectures for scalable image recognition, in: CVPR, 2018, pp. 8697–8710.

[10] L. Shapley, A Value for N-Person Games, Contributions to the Theory of Games (1953) 307–317.

[11] A. Heuillet, A. Nasser, H. Arioui, H. Tabia, Efficient automation of neural network design: A survey on differentiable neural architecture search, arXiv preprint arXiv:2304.05405 (2023).

[12] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, H. Xiong, Pc-darts: Partial channel connections for memory-efficient architecture search, in: ICLR, 2020.

[13] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive darts: Bridging the optimization gap for nas in the wild, International Journal of Computer Vision 129 (2021) 638–655.

[14] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, J. Yan, Darts-: Robustly stepping out of performance collapse without indicators, in: ICLR, 2021.

[15] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, W. Ouyang, β-darts: Beta-decay regularization for differentiable architecture search, in: CVPR, 2022.

[16] H. Yu, H. Peng, Y. Huang, J. Fu, H. Du, L. Wang, H. Ling, Cyclic differentiable architecture search, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (2022) 211–228.

[17] L. Huang, S. Sun, J. Zeng, W. Wang, W. Pang, K. Wang, U-darts: Uniform-space differentiable architecture search, Information Sciences 628 (2023) 339–349.

[18] Y.-C. Gu, L.-J. Wang, Y. Liu, Y. Yang, Y.-H. Wu, S.-P. Lu, M.-M. Cheng, Dots: Decoupling operation and topology in differentiable architecture search, in: CVPR, 2021.

[19] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, P. Dollár, Designing network design spaces, in: CVPR, 2020, pp. 10428–10436.

[20] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, et al., Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions, in: CVPR, 2020, pp. 12965–12974.

[21] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, in: ICLR, 2018.

[22] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning, in: ICLR, 2017.

[23] R. Meyes, M. Lu, C. W. de Puiseau, T. Meisen, Ablation studies to uncover structure of learned representations in artificial neural networks, in: Proceedings on the International Conference on Artificial Intelligence, The Steering Committee of The World Congress in Computer Science, Computer . . . , 2019, pp. 185–191.

[24] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: NeurIPS, 2017, pp. 4768–4777.

[25] A. Heuillet, F. Couthouis, N. Díaz-Rodríguez, Collective explainable ai: Explaining cooperative strategies and agent contribution in multiagent reinforcement learning with shapley values, IEEE Computational Intelligence Magazine 17 (2022) 59–71.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR, 2009.

[27] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: CVPR, 2017, pp. 1251–1258.

[28] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: ECCV, Springer, 2014, pp. 740–755.

[30] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[31] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[32] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.