

# D-DARTS: Distributed Differentiable Architecture Search

Alexandre Heuillet  
 Université Paris-Saclay  
 alexandre.heuillet@univ-evry.fr

Hichem Arioui  
 Université Paris-Saclay  
 hichem.arioui@univ-evry.fr

Hedi Tabia  
 Université Paris-Saclay  
 hedi.tabia@univ-evry.fr

Kamal Youcef-Toumi  
 MIT  
 youcef@mit.edu

August 2021

## Abstract

Differentiable ARchitecture Search (DARTS) is one of the most trending Neural Architecture Search (NAS) methods, drastically reducing search cost by resorting to Stochastic Gradient Descent (SGD) and weight-sharing. However, it also greatly reduces the search space, thus excluding potential promising architectures from being discovered. In this paper, we propose D-DARTS, a novel solution that addresses this problem by nesting several neural networks at cell-level instead of using weight-sharing to produce more diversified and specialized architectures. Moreover, we introduce a novel algorithm which can derive deeper architectures from a few trained cells, increasing performance and saving computation time. Our solution is able to provide state-of-the-art results on CIFAR-10, CIFAR-100 and ImageNet while using significantly less parameters than previous baselines, resulting in more hardware-efficient neural networks.

## 1 Introduction

With the field of Deep Learning (DL) getting more and more attention over the past few years, an important focus has been set on neural network architectural conception. A large number of architectures have been manually crafted to address different computer vision problems [7, 9, 10, 19]. However, the design of these human-made architectures is mainly driven by intuition and lacks

the certainty of an optimal solution. This is mainly due to the vast number of possible combinations needed to build a relevant neural architecture, making the manual search space very difficult. On the other side, there have recently been a number of works [15] that tried to automate the architecture design process. These works are referred to as Neural Architecture Search (NAS). In NAS, a search algorithm attempts to build a neural network architecture from a defined search space by asserting the performance of “candidate” architectures. Early works such as [23] are based on Reinforcement Learning (RL), but they are very costly with hundreds or thousands of GPU days needed to obtain competitive architectures. Since then, new alternatives have been proposed such as gradient-based methods [3, 5, 11, 20, 21, 22]. In these methods, the search space is relaxed to be continuous so that the architecture can be optimized by a gradient descent algorithm e.g. Stochastic Gradient Descent (SGD). They operate on a small component, i.e. cell, as the building block of the designed architecture. The obtained cell could either be stacked up multiple times to build a convolutional network or recursively connected to build a recurrent network [3, 11, 20]. The main advantage of gradient-based methods over RL-based ones is the greatly reduced search cost, by several orders of magnitude (see Section 5). However, despite the progress brought by Differentiable ARchitecture Search (DARTS) [11], existing methods greatly reduce the search space of relevant neural architecture as searching for building blocks limits the

algorithm’s creativity. To alleviate this problem, we propose to directly search for a super network which includes a set of learnable cells. Moreover, to further optimize the super network architecture, we leverage Game theory. We introduce a novel loss function based on Shapley value [18] concept.

As presented in Section 4, the contributions of this paper are:

- A new way of structuring the DARTS search process by nesting small neural networks at cell level.
- A novel loss specially designed to take advantage of the new distributed structure.
- A mechanism to derive larger architectures from a few well-trained highly specialized cells.

The rest of the paper is structured as follows: In Section 2, we conduct a short survey on recent differentiable NAS works, in Section 3, we review the original concept of DARTS and discuss its issues, 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 and Section 7 finally brings a conclusion to this article while giving some insights on promising directions of future work.

## 2 Related Work

Few other works already attempted to improve on DARTS by addressing these limitations, such as PC-DARTS [22], P-DARTS [4] and FairDARTS [5]. In PC-DARTS, authors tried to minimize the memory footprint of DARTS by optimizing the search process to avoid redundancy. P-DARTS (Progressive DARTS) was able to greatly reduce search time by progressively deepening the architecture during search, leading to better search space approximation and regularization. FairDARTS tried to solve two important problems that occurred in DARTS, the over-representation of *skip* connections and the uncertainty in the probability distribution of operations, leading to a “fairer” system. Both of these works obtained state-of-the-art results on popular datasets [6, 10].

Motivated by the success of DARTS, this present work explores for the first time a gradient-based distributed

search procedure for deep convolutional neural network architectures.

## 3 Preliminaries: DARTS

DARTS (Differentiable ARchitecTure Search) [11] 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 (i.e. that composes most of the architecture) and reduction (i.e. that performs dimension reduction). During the search process, a small size proxy network (supernet) with only a few of these cells (e.g. 8 as suggested in [11]) is trained. Once inferred, these two cells are the building blocks from which architectures of any size can be derived, similarly as residual blocks in ResNet [7] and thus most of the final network components share the same architectural weights. In particular, they are stacked multiple times to form a network of the desired size (e.g. 14 or 20 layers as in [11]), determined empirically depending on if the emphasis is put on raw performance or hardware efficiency. 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  $O_{i,j} = \{o_{i,j}^1, \dots, o_{i,j}^k\}$  represents the set of all possible operations for the edge  $e_{i,j}$  connecting node  $i$  to node  $j$ . The goal here is to determine which operations (with a maximum of 2) must be selected for each edge in order to maximize the validation loss  $L_{val}$ . Such loss corresponds here to the Cross Entropy loss  $L_{CE}$ , a widely used categorical loss function derived from C.E. Shannon’s Theory of Information [17] and defined as follows:

$$L_{CE}(x, t_c) = -\log \left( \frac{\exp(x[t_c])}{\sum_{j=0}^{|x|} \exp(x[j])} \right) \quad (1)$$

where  $x$  is the output of the final linear classifier (i.e. a probability distribution) and  $t_c$  is the target class for this output.

In that objective, DARTS search process involves a set of parameters, denoted by  $\alpha^{(i,j)} = \{\alpha_1^{(i,j)}, \dots, \alpha_k^{(i,j)}\}$  which represents the weight of each operation from  $O_{i,j}$  in the mixed output of each edge. To build the mixed out-

put, the categorical choice of operations is done through a *softmax*:

$$\bar{o}_{i,j}(x) = \sum_{o \in O_{i,j}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in O_{i,j}} \exp(\alpha_{o'}^{(i,j)})} o(x) \quad (2)$$

where  $\bar{o}_{i,j}(x)$  is the mixed output of edge  $e_{i,j}$  for input feature  $x$  and  $\alpha_o^{(i,j)} \in \alpha^{(i,j)}$  is the weight associated with operation  $o \in O_{i,j}$ .

These parameters are optimized using a gradient descent algorithm while the global supernet (from which the cells are part of) is trained on a given dataset. Thus, DARTS is practically solving a bi-level optimization problem. DARTS search process is summarized in Fig. 1.

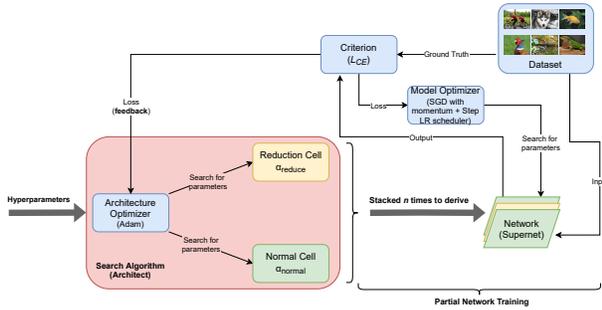


Figure 1: Layout of the search process at work in DARTS [11]. A global optimizer searches for two sets of parameters (i.e.  $\alpha_{normal}$  and  $\alpha_{reduce}$ ) that define the architecture of cells. The two types of searched cells are stacked multiple times to build a proxy network (supernet) which is trained in order to validate the performance of these cells.

DARTS suffers from two majors issues. The first is the over-representation of *skip connections*. The second is the discretization discrepancy problem of the *softmax* operation, namely a very small standard deviation of the resulting probability distribution. To mitigate these issues, the authors of FairDARTS [5] replaced the *softmax* function by the *sigmoid* function (denoted by  $\sigma$ ) and 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:

$$L_{01} = -\frac{1}{N}(\sigma(\alpha) - 0.5)^2 \quad (3)$$

In fact,  $L_{01}$  corresponds to the mean square error between  $\sigma(\alpha)$  and 0.5.  $L_{01}$  is then added to  $L_{CE}$  to form FairDARTS total loss  $L_F$ :

$$L_F = L_{CE} + w_{01} * L_{01} \quad (4)$$

Despite these solutions, DARTS and all of its evolutions [4, 5, 11, 22] are still limited in their capacity to create original architectures since most of their structure is rigid and human-made (e.g. the search space modulation or the number of cells searched) and the search space is very restricted as pointed out by prior works [14, 20].

This is the issue we are trying to address in this paper. Contrary to other approaches, such as FBNetV2 [20] where the authors used a masking mechanism for feature map reuse to increase search speed, and included a wider range of spatial and channel dimensions. This paper is the first to explore and implement distributed differential architectural search for the first time.

## 4 Proposed Approach

### 4.1 Delegating search to cell-level subnets

As explained in section 3, DARTS [11] only searches for two types of cells (“normal” and “reduce”) and stacks them multiple times to form a network as large as needed. This weight-sharing process has the advantage of reducing search space to a limited set of parameters (i.e.  $\alpha$ ) 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  $s(n_o, n_e)$  of a single cell with  $n_o$  primitive operations to select from (within a maximum of 2) and  $n_e$  edges can be computed as follows:

$$s(n_o, n_e) = \binom{n_o}{2}^{n_e} = \left( \frac{n_o!}{2!(n_o - 2)!} \right)^{n_e} \quad (5)$$

Following Eq. 5, using DARTS default parameters ( $n_o = 7$  and  $n_e = 14$ ), the search space of a single cell comprises  $10^{18}$  possible configurations. Thus, as both *normal* and *reduce* cells share the same  $n_o$  and  $n_e$ , the total search space size of DARTS is  $10^{36}$  possibilities. This number is comparable to other differentiable NAS works [3, 20],

but far lower than those of Reinforcement Learning based NAS methods [2, 23] that describe architecture topologies using sequential layer-wise operations, which are also far less efficient.

The key idea behind our method is to increase diversity in the architecture by delegating the search process to sub-nets nested in each cell. This way each cell that composes the global supernet is individual and is itself a full neural network with its own optimizer, criterion, scheduler and its input, hidden and output layers, as shown in Fig. 2. In addition, instead of searching for building blocks as DARTS do, we increase the number of searched cells to an arbitrary  $n$  (e.g. 8) and directly seek for a full  $n$ -layer convolutional neural network. Thus, we trade the weight sharing process introduced by DARTS for greater flexibility and creativity. Nonetheless, cells still belong either to the *normal* or *reduction* class, depending on their position in the network (reductions cells are positioned at the  $\frac{1}{3}$  and  $\frac{2}{3}$  of the network).

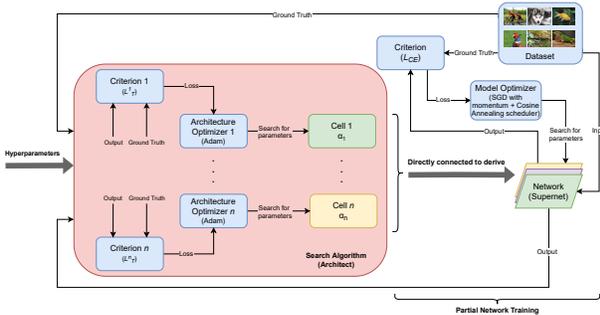


Figure 2: Layout of the search process used in D-DARTS. Cells are still divided in two types (*normal*, *reduction*) but each cell  $i$  is independent with its own optimizer, scheduler and criterion (based on our novel ablation loss  $L_T$ ) which search for its architectural parameters  $\alpha_i$ , thus making the entire search process distributed. The searched cells are directly connected to one another to form a proxy network (supernet) which is trained to validate their performance.

The architectural parameters ( $\alpha$ ) are computed using logits. In this context, logits represents the probability distribution generated by the linear classifier placed at the end of the supernet. At each training step, they are computed using the global supernet and are passed down to

cells which use them to compute their individual losses and gradients in order to update their architectural weights ( $\alpha$ ). This way, we directly build a full and complete neural network where each cell is highly specialized (w.r.t. its position in the supernet, contrary to generic building blocks), thus effectively expanding the search space by a factor of  $10^{(n-2)*18}$  (according to Eq. 5) where  $n$  is the total number of searched cells (e.g. a factor a  $10^{118}$  when considering 8 cells). To the best of our knowledge, this is the largest search space ever explored by a differentiable NAS method. In section 5, we show that smaller (e.g. 7 or 8 layers) NestedDARTS architectures can achieve similar or higher performance than large (e.g. 14 or 20 layers) DARTS architectures on common datasets.

## 4.2 Adding a new cell-specific loss

In addition to the new network structure introduced in subsection 4.1, we designed a novel cell-specific loss function that we dubbed ablation loss. Indeed, as we increased the number of searched cells, the learning challenge became greater with a large amount of additional parameters to take into account. Thus, the global loss functions used in DARTS [11] and FairDARTS [5] 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 assigned to and is an additive loss, based on the global loss function introduced in [5] that proved to be a significant improvement over the original one [11].

The main idea behind this ablation loss function is to perform a limited ablation study on the cell level. Indeed, ablation studies have long proven to hold a key role in asserting the effectiveness of neural network architectures [13]. This way, by computing the difference in the supernet loss  $L_{CE}$  (i.e. the Cross Entropy loss, see Eq. 1) with and without each individual cell activated, we can obtain a measure of their respective contributions that we call their marginal contributions, noted  $MC = \{MC_1, \dots, MC_n\}$ . This method is inspired by Shapley values [18], a game theory technique widely used in Explainable Artificial Intelligence to assess the contributions of model features to the final output [1, 12] or the contributions of agents to the common reward in a cooperative multi-agent Reinforcement Learning context [8]. Thus, cell  $C_i$  marginal contribution  $MC_i$  is computed as

follows:

$$MC_i = L_{CE}(C, \alpha, w) - L_{CE}(C \setminus \{C_i\}, \alpha, w) \quad (6)$$

Where  $C$  is the set containing all cells such as  $C = C_1, \dots, C_n$ ,  $\alpha$  is the encoding of the architecture (see section 3) and  $w$  are the weights associated with the architecture. Once we obtained all the marginal contributions  $MC$ , we apply the following formula to compute the ablation loss of cell  $C_i$ :

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

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

$$L_T^i = L_F + w_{abl} * L_{AB}^i \quad (8)$$

Finally, when expanding Eq. 8, we obtain:

$$L_T^i = L_{CE} + w_{01} * L_{01} + w_{abl} * L_{AB}^i \quad (9)$$

where  $L_{CE}$  is the CrossEntropy loss (see Eq. 1),  $L_{01}$  is the Zero-One loss introduced in FairDARTS [5] (see Eq. 3) and  $w_{01}$  is an hyperparameter weighting  $L_{01}$ .

In section 5, we show that  $L_T^i$  can warm start the search process and provide a substantial increase in performance. However, it also increases GPU memory usage significantly, as shown in Section 5.2.

### 4.3 Building larger networks from a few highly specialized cells

In previous works [5, 11, 22], the final network architecture was derived from the two searched cells (i.e. *normal* and *reduce*) which were stacked as much time as needed to build a network with the desired number of layers (e.g. 10, 15 or 20 layers).

However, as presented in subsection 4.1, in D-DARTS, we directly search for a “full” network of multiple individual cells instead of searching for building block cells as in DARTS [11]. But, the downside of this method is

that searching for a high number of cells is memory hungry (see Section 5.3), as each cell must possess its own optimizer, criterion and parameters. This is not a critical issue as we show in section 5 that a few (e.g. 8) of these highly specialized cells can outperform a large number of base cells.

Nonetheless, it may be useful to use a larger number of cells without spending additional search time, especially when dealing with highly complex datasets such as ImageNet [6]. Thus, to save computation time and still profit from a high number of layers, we developed a new algorithm to derive larger architectures from an already searched smaller one, inspired by what is done in DARTS [11]. 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 [5], and repeat the searched structure of “normal” cells in the intervals between the reduction cells until we obtain the desired number of cells. This process is summarized in Algorithm 1.

---

#### 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 layers

**Ensure:** List:  $C_f$ , list of cells that compose the derived architecture

```

 $C_f \leftarrow \text{empty\_list}()$ 
 $m \leftarrow \text{euclidean\_division}(|C|, 3)$ 
 $m2 \leftarrow \text{euclidean\_division}(2 * |C|, 3)$ 
for  $i$  in  $[0, n]$  do
  if  $n > |C|$  then
    if  $i < \text{euclidean\_division}(n, 3)$  then
       $c \leftarrow \text{modulo}(i, m)$ 
    else if  $i = \text{euclidean\_division}(n, 3)$  then
       $c \leftarrow m$ 
    else if  $i > \text{euclidean\_division}(n, 3)$  and  $i < \text{euclidean\_division}(2 * n, 3)$  then
       $c \leftarrow \text{modulo}(i, m2 - 1 - m) + m + 1$ 
    else if  $i = \text{euclidean\_division}(2 * n, 3)$  then
       $c \leftarrow m2$ 
    else
       $c \leftarrow \text{modulo}(i, |C| - 1 - m2) + m2 + 1$ 
    end if
  end if
  else
     $c \leftarrow i$ 
  end if
   $\text{append}(c, C_f)$ 
end for

```

---

Thus, Algorithm 1 allows us to obtain a larger architecture without the need to launch a new search (i.e. without

any overhead). In section 5, we show that doubling the number of layers this way can result up to a 1% increase of top1 accuracy when evaluating on CIFAR-100 [10]. However, the gain is more limited (around 0.15 %) for simpler datasets such as CIFAR-10 [10] where the base model already performs very well.

## 5 Experiments

We focused on searching for convolutional network architectures and evaluating them on image classification tasks.

### 5.1 Experimental Settings

Experiments were conducted on CIFAR-10, CIFAR-100 [10] (search, evaluation) and ImageNet [16] (transfer) using Nvidia GeForce RTX 3090 and Tesla V100 GPUs. We mostly use the same data processing, hyperparameters and training tricks than FairDARTS [5] and DARTS [11]. However, due to the increase in memory consumption that comes with the new search algorithm and loss (see Section 5.3), we lowered the search batch size from 96 to 72, which had the negative effect of increasing the search time. Moreover, we used PyTorch Automatic Mixed Precision to speed up floating point operations, mainly when fully training models. 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  $\sigma = 0.8$ , meaning that the maximum number of operations per edge is limited to 2 as in [5, 11]. 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, on the other hand, systematically selects the two operations with the highest *softmax* weights for each edge (this parsing method is referred as *darts* in Table 1 and Table 3). Our implementation is based on PyTorch 1.7.1 and derived from the one of [5].

### 5.2 Analysis of Ablation Loss: What impact does it have?

#### 5.2.1 Hyperparameter Choice

We made the hyperparameter weights of the ablation loss  $w_{abl}$  and the zero-one loss  $w_{01}$  from Eq. 8 vary in order to choose their optimal value according to the global loss. Thus, in Fig. 3 we made  $w_{abl}$  vary from 0 to 2 while keeping zero-one loss deactivated (i.e.  $w_{01} = 0$ ) in order to analyze its impact. We can see that an optimal value seems to be attained around 0.5 with the global loss mainly increasing when  $w_{abl}$  reaches higher or lower values.

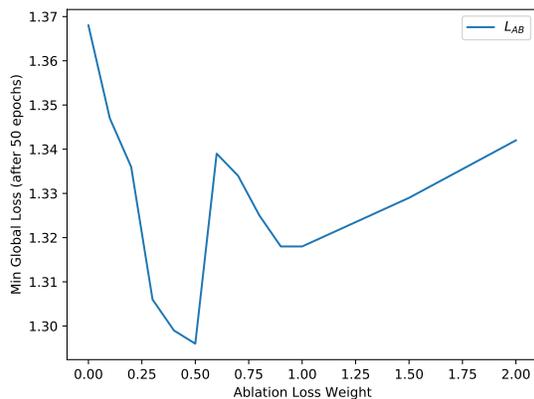


Figure 3: Line plot of the minimal global loss obtained by searching for a model on CIFAR-100 [10] for 50 epochs w.r.t. the sensitivity weight  $w_{abl}$  used for the ablation loss  $L_{AB}$ . We deactivated  $L_{01}$  (i.e. we set  $w_{01} = 0$ ) to prevent interference from occurring.

Moreover, we made the value of the sensitivity weight  $w_{01}$  (used for  $L_{01}$ , see Section 3) vary during search on CIFAR-100, with  $w_{abl} = 0.5$  fixed, and reported the number of dominant operations (i.e. operations whose softmax weight value  $\sigma(\alpha)$  is greater than 0.9). This experiment was conducted in order to select a relevant value for  $w_{01}$  since the value chosen by the authors of FairDARTS [5] ( $w_{01} = 10$ ) is no longer valid as the search process has been altered. Fig. 4 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 operations softmax values are either greater than 0.9 or inferior to 0.1. Finally, we chose 7 as the optimal value for  $w_{01}$ , as it offers both a high number of dominant operations and an equilibrium between operations whose softmax values is greater than 0.9 and those whose softmax value is inferior to 0.1.

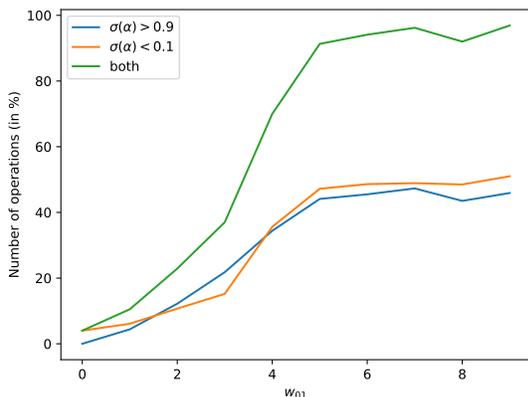


Figure 4: Line plot showing the percentage of dominant operations obtained in the final architecture  $\alpha$  while searching on CIFAR-100 w.r.t. the sensitivity weight  $w_{01}$  used for  $L_{01}$ . We can see that the proportion of both types of operations stabilizes after  $w_{01} = 5$  and reaches an equilibrium at  $w_{01} = 7$ .

### 5.2.2 Ablation Study

We conducted an ablation study on our proposed ablation loss  $L_T$  (see Eq. 8). In particular, we compared the performance of architectures with similar characteristics searched either with  $L_T$  or FairDARTS [5] loss  $L_F$  (see Eq. 4). Tables 1, 2 and 3 show that  $L_T$ -searched architectures (DD-3, DD-4, DD-5) outperform their  $L_F$ -searched counterparts by an average of 0.7 % across all datasets, confirming the advantage procured by this new loss function. When considering models that leveraged Algorithm 1 to increase their number of layers (e.g. DD-2 and DD-4), this performance boost seems to be less significant (e.g. around 0.1 % on CIFAR-100). In addition, the loss impact is less important for *sparse* parsed models, especially on CIFAR-10 (where DD-1 and DD-3 achieve

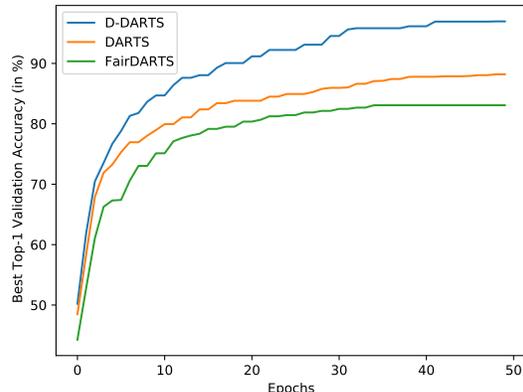


Figure 5: Lineplot showing the best validation top-1 accuracy while searching on CIFAR-10[10] w.r.t. the current epoch. D-DARTS clearly outperforms both DARTS and FairDARTS by a large margin.

similar scores). This could be explained by the fact that CIFAR-10 is a dataset simple enough for all methods to always affect the same weight importance to the same operation for each edge (as the *sparse* parsing method selects only one operation per edge).

### 5.2.3 Convergence Speed

We conducted an experiment on model convergence speed in order to compare the performance of D-DARTS with previous baselines [11, 5]. Fig. 5 shows that D-DARTS converges very quickly, hitting a final plateau around epoch 40 and outperforming both DARTS and FairDARTS respectively by 9 % and 14 %.

## 5.3 Memory Efficiency

When searching using  $L_T$  (see Eq. 8), additional tensors must be stored on GPU memory, due to the computations required to obtain the marginal contribution of each cell. The memory usage with  $L_T$  can be much more than what an  $L_F$ -based search requires. For example, an  $L_T$  search increase in memory consumption can be around 100% in memory consumption using an Nvidia RTX 3090 on CIFAR-100 [10] with a batch size of 72, taking up to

21911 MB of memory versus 11100 MB with  $L_F$ . In consequence, this can be identified as a critical issue of our method: we have to find means to reduce this memory consumption so we could search for deeper networks and make our method available to lower end GPUs.

## 5.4 Searching architectures on CIFAR

When searching and evaluating on CIFAR-10 and CIFAR-100 [10], we mainly use 8 layer networks and select either 1 or 2 operations per edge in order to test how smaller architectures compete with larger ones. We also compared the performance of models using FairDARTS [5] loss function with ones using our new ablation-based total loss function  $L_T$  so we may assert its effectiveness (see Section 5.2). For instance, when considering *edge* parsed models evaluated on CIFAR-10, the 8-cell model DD-4 reached a top-1 accuracy of 97.48 %, thus outperforming DD-2 (i.e. a model trained without ablation-based loss) by around 0.4 %. However, results are more mixed for *sparse* parsed models (DD-1, DD-3, DD-5) as they all achieve similar results (around 97 %) no matter the dataset CIFAR-10 or CIFAR-100, or the loss,  $L_T$  rather than  $L_F$ . This could be explained by the fact that CIFAR-10 is a dataset simple enough for all methods to always affect the same weight importance to the same operation for each edge (as the *sparse* parsing method selects only one operation per edge). Nonetheless, NestedDARTS models all reach competitive results in both datasets. The smaller ones, such as DD-1 or DD-3, can achieve the same level of performance than previous baselines while possessing significantly less parameters (e.g. 1.7M against 2.8M for the smallest model of [5]) while the largest can leverage better results (e.g. 84.15 % top-1 accuracy for DD-4 on CIFAR-100). One additional point to note is that our proposed ablation-based loss  $L_T$  seems to provide a larger increase in performance for CIFAR-100. For example, it is around 1 % for CIFAR-100 between the 8-cell versions of DD-2 and DD-4. However, this gap seems to tighten when resorting to Algorithm 1 to deepen the architectures (e.g. there is only a 0.1 % increase between the 14-cell versions of DD-2 and DD-4). Moreover, using Algorithm 1 effectively provides a performance boost in both datasets (e.g. around 0.3 % for DD-4 when using 14 cells instead of 8) thus asserting its usefulness. Finally, one last comment is that

there is a consequent performance variation (e.g. around 0.5 % between DD-4 and DD-5 on CIFAR-10) when using the *edge* parsing method rather than the *sparse* one while making the number of parameters double. Interestingly, this impact is more important in CIFAR-100 (around 0.8 %). This may be related to the fact that using larger architectures is less pertinent on simple datasets such as CIFAR-10 where *sparse* models already achieve very high top-1 scores (greater or equal to 97 %) while it is much more relevant on more challenging datasets like CIFAR-100. All results are presented in details in Table 1 and Table 2.

## 5.5 Transferring to ImageNet

In order to test our approach on a more challenging dataset, we transferred our best models searched on CIFAR-10 and CIFAR-100 [10] to ImageNet [16], and trained each with and without our proposed ablation loss  $L_{AB}$  in order to conduct an ablation study (see Section 5.2). We trained models using an RTX 3090 and we kept the same hyperparameters and tricks as in [5, 11]. Training a model for 400 epochs takes around 10 days on a single GPU. Table 3 shows that model DD-5 (transferred from CIFAR-100) reached a top-1 accuracy of 75.03 %, outperforming DARTS [11] by 1.7 %. However, DD-5 does not reach the level of performance of FairDARTS-D [5], mainly because it was searched on CIFAR-100 and not on ImageNet. Finally, there is an important gap (around 3.7 %) between DD-2 and DD-5, showing that using both  $L_T$  and CIFAR-100 makes a significant impact on performance.

## 6 Discussion

In Section 4 we proposed a new approach for differentiable architecture search that is based on a cell-level distributed search mechanism. Instead of searching for building blocks, we directly search for a complete super network composed of multiple subnets nested at the cell-level. To bring an additional performance boost to this new mechanism, we also introduced an ablation-based loss function that leverages Game Theory concepts in order to take into account the marginal contributions of each cell to the common goal (i.e. the classifica-

Models	Params (M)	Parsing Method	Loss	Top-1 (%)	Layers	Search Cost (GPU-days)	Searched On	Type
NASNet-A[24]	3.3M	N.A.	N.A.	97.35	N.A.	2000	CIFAR-10	RL
DARTS[11]	3.3	<i>darts</i>	$L_{CE}$	97.00	20	1.5	CIFAR-10	GD
PC-DARTS[22]	3.6	<i>darts</i>	$L_{CE}$	97.43	20	3.8	CIFAR-10	GD
P-DARTS[4]	3.4	<i>darts</i>	$L_{CE}$	97.50	20	0.3	CIFAR-10	GD
FairDARTS-a[5]	2.8	<i>sparse</i>	$L_F$	97.46	20	0.4	CIFAR-10	GD
DD-1	<b>1.7</b>	<i>sparse</i>	$L_F$	97.00	<b>8</b>	0.5	CIFAR-10	GD
DD-2	3.3	<i>edge</i>	$L_F$	97.11	<b>8</b>	0.5	CIFAR-10	GD
Ours DD-3	<b>1.7</b>	<i>sparse</i>	$L_T$	97.02	<b>8</b>	0.5	CIFAR-10	GD
DD-4	3.9	<i>edge</i>	$L_T$	97.48	8	0.9	CIFAR-100	GD
DD-4	3.9	<i>edge</i>	$L_T$	<b>97.75</b>	14	0.9	CIFAR-100	GD
DD-5	<b>1.7</b>	<i>sparse</i>	$L_T$	97.01	<b>8</b>	0.9	CIFAR-100	GD

Table 1: Comparison of models on CIFAR-10 [10]. Each reported Top-1 accuracy is the best of 4 independent runs. For previous baselines, results are the official numbers from their respective papers with the search cost expressed with the GPU used by the authors.

tion task). Moreover, since it would be too costly to directly search for large networks, we presented Algorithm 1, a procedure to automatically derive larger architectures from smaller ones by stacking multiple times specific sequences of cells.

In Section 5 we showed that these proposed concepts perform well but are not exempt from limitations. In particular, this approach is less efficient w.r.t. memory (see Section 5.3) as with previous baselines [11, 5]. Finding ways to reduce video memory consumption would allow to directly search for larger architectures (e.g. 12 or 14 layers) that could prove highly beneficial for challenging tasks such as ImageNet [10].

Nonetheless, combining the *sparse* threshold parsing method with our distributed design allowed to obtain architectures that are of an unprecedentedly small size (around 1.7 M for the tiniest) and can still yield competitive results. Furthermore, while using the *edge* threshold parsing method, it is also possible to search for larger size models that reach state-of-the-art results. This demonstrates the flexibility and the usefulness of our method.

## 7 Conclusion and Future Work

In this paper, we proposed a new cell-based approach for DARTS in section 4 and showed in section 5 that it effectively achieves state-of-the-art results on popular datasets while restricting architectures to a moderate size. In addition, our novel architecture derivation algorithm allows us to derive larger architectures from only a small number of cells, without further training.

However, this does come at the cost of an increase in memory usage. This is mainly due to the greater number of optimizers and search parameters as well as additional computations done by the loss function that slows down the search process as we have to cope with a lower batch size. Thus, our future work will focus on improving memory usage to speed up the search process. Furthermore, many ideas around this approach have not been explored yet, such as improving the selection of search hyperparameters, improving cell optimizers or reducing the search cost by adding a weight sharing mechanism.

## References

- [1] Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time al-

Models	Params (M)	Parsing Method	Loss	Top-1 (%)	Layers	Search Cost (GPU-days)	Searched On	Type	
DARTS[11]	3.3	<i>darts</i>	$L_{CE}$	82.34	20	1.5	CIFAR-100	GD	
P-DARTS[4]	3.6	<i>darts</i>	$L_{CE}$	84.08	20	0.3	CIFAR-100	GD	
FairDARTS[5] <sup>†</sup>	3.5	<i>sparse</i>	$L_F$	83.80	20	0.4	CIFAR-100	GD	
Ours	DD-2	3.3	<i>edge</i>	$L_F$	82.90	<b>8</b>	0.5	CIFAR-10	GD
	DD-2	3.3	<i>edge</i>	$L_F$	84.06	14	0.5	CIFAR-10	GD
	DD-3	<b>1.7</b>	<i>sparse</i>	$L_T$	81.92	<b>8</b>	0.5	CIFAR-10	GD
	DD-4	3.9	<i>edge</i>	$L_T$	83.86	<b>8</b>	0.9	CIFAR-100	GD
	DD-4	7.6	<i>edge</i>	$L_T$	<b>84.15</b>	14	0.9	CIFAR-100	GD
DD-5	<b>1.7</b>	<i>sparse</i>	$L_T$	81.1	<b>8</b>	0.9	CIFAR-100	GD	

Table 2: Comparison of models on CIFAR-100 [10]. Each reported Top-1 accuracy is the best of 4 independent runs. For previous baselines, results are the official numbers from their respective papers with the search cost expressed with the GPU used by the authors. †: Result obtained by running the code released by the authors.

Models	Params (M)	+× (M)	Parsing Method	Loss	Top-1 (%)	Layers	Search Cost (GPU-days)	Searched On	Type	
ResNet50[7]	28	3800	N.A.	$L_{CE}$	<b>76.15</b>	50	N.A.	N.A.	manual	
NASNet-A[24]	5.3	564	N.A.	N.A.	74.0	N.A.	2000	ImageNet	RL	
DARTS[11]	4.7	574	<i>darts</i>	$L_{CE}$	73.3	14	4	CIFAR-100	GD	
PC-DARTS[22]	5.3	586	<i>darts</i>	$L_{CE}$	74.9	14	3.8	CIFAR-10	GD	
P-DARTS[4]	5.1	577	<i>darts</i>	$L_{CE}$	74.9	14	0.3	CIFAR-100	GD	
FairDARTS-D[5]	4.3	440	<i>sparse</i>	$L_F$	75.6	14	3	ImageNet	GD	
Ours	DD-2	5.44	590	<i>sparse</i>	$L_F$	71.34	14	0.5	CIFAR-10	GD
	DD-5	5.74	617	<i>sparse</i>	$L_T$	75.03	14	0.9	CIFAR-100	GD

Table 3: Comparison of models on ImageNet [16] in the *mobile* setting. For previous baselines, results are the official numbers from their respective papers with the search cost expressed with the GPU used by the authors.

- gorithm for shapley value approximation. In *International Conference on Machine Learning*, pages 272–281. PMLR, 2019.
- [2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.
- [3] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.
- [4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation, 2019.
- [5] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search, 2020.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [8] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. Explainability in deep reinforcement learning, 2020.

- [9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.
- [10] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- [11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.
- [12] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.
- [13] Richard Meyes, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. Ablation studies in artificial neural networks, 2019.
- [14] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.
- [15] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020.
- [16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [17] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.
- [18] L. S. Shapley. The value of an n-person game. 1951.
- [19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [20] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020.
- [21] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search, 2020.
- [22] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guojun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search, 2020.
- [23] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2017.
- [24] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.