Neural Architecture Search with Structure Complexity Control

Konstantin Yakovlev, Olga Grebenkova, Oleg Bakhteev and Vadim Strijov

May 21, 2022

Neural architecture search with structure complexity control[⋆]

K. D. Yakovlev[1], O. S. Grebenkova[1], O. Y. Bakhteev[1,2], and V. V. Strijov[1,2]

[1] MIPT, Russia
[2] Dorodnicyn Computing Center RAS, Russia
{iakovlev.kd, grebenkova.os, bakhteev, strijov}@phystech.edu

Abstract. The paper investigates the problem of deep learning model selection. We propose a method of a neural architecture search with respect to the desired model complexity called DARTS-CC. An amount of parameters in the model is considered as a model complexity. The proposed method is based on a differential architecture search algorithm (DARTS). Instead of optimizing structural parameters of the architecture, we consider them as a function depending on the complexity parameter. It enables us to obtain multiple architectures at one optimization procedure and select the architecture based on our computation budget. To evaluate the performance of the proposed algorithm, we conduct experiments on the Fashion-MNIST and CIFAR-10 datasets and compare the resulting architecture with architectures obtained by other neural architecture search methods.

Keywords: differential architecture search · deep learning · hypernetwork · neural networks · model complexity control.

1 Introduction

In this paper, we consider the problem of searching the architecture of a deep learning model with the control of its complexity. A model is considered as a directed graph with edges corresponding to non-linear functions on the dataset and the vertices corresponding to intermediate representations of the dataset under the operations. In this paper, we base on the differential algorithm DARTS [16]. It solves the problem of searching the model architecture by translating the search space of structural parameters from a discrete to a continuous representation. In contrast to previous algorithms that were based on discrete optimisation problem, DARTS does not suffer from combinatorial explosion. Due to

⋆ This paper contains results of the project Mathematical methods of intelligent big data analysis, which is carried out within the framework of the Program "Center of Big Data Storage and Analysis" of the National Technology Initiative Competence Center. It is supported by the Ministry of Science and Higher Education of the Russian Federation according to the agreement between the M.V. Lomonosov Moscow State University and the Foundation of project support of the National Technology Initiative from 11.12.2018, No 13/1251/2018. This research was supported by RFBR (projects 19-07-01155, 19-07-00885).

relaxation, the algorithm does not use an exhaustive search to solve the optimization task. Furthermore, this relaxation enables us to use gradient-based optimization for the model architecture selection.

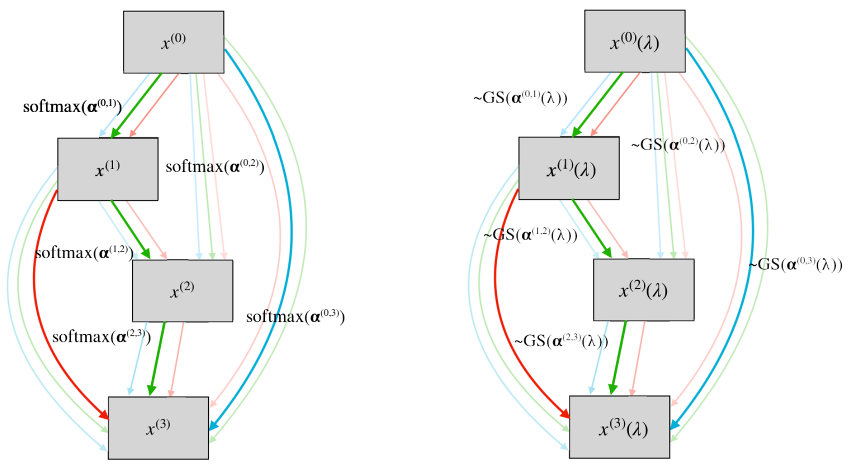
Although a significant success in neural architecture search-based model selection [25], the problem of model selection is still can be a challenge, especially when dealing with constraints on computational budget or model size [21, 11]. The paper [21] presents an approach to use regularization term to control the model complexity. It mainly focuses on the target hardware for further model exploitation. A similar approach can be found in the paper [11], where a neural architecture search is employed with a limited resource (RC-DARTS). Restrictions are added to the basic DARTS algorithm, such as the number of model parameters. In order to solve the problem of conditional optimization, an iterative projection algorithm is introduced, which consists of the fact that after a certain number of iterations of gradient descent, the projection occurs on the set specified by the constraints. However, both algorithms require the launch of an individual architecture search process for every set of complexity values, so they still can be too time-consuming.

This paper investigates the problem of the model complexity control during architecture search. Opposite to the listed works above, our method, which is called DARTS-CC, is based on the hypernetwork concept [8].A hypernetwork is a model that generates the parameters of the target deep learning model. In this paper, we employ hypernetworks to generate structural parameters that control the final architecture of the model. It enables us to obtain multiple neural architectures during the architecture search procedure for further model fine-tuning with an architecture that suits our computation budget in the best way. The main idea of our paper and the difference between our method and DARTS is shown in Fig. 1. Instead of using constant structural parameters $\boldsymbol{\alpha}^{(i,j)}$ that control the model architecture, we propose to consider them as functions on the model complexity parameter λ. We also use Gumbel-softmax distribution for sampling non-linear operations instead of using softmax function on the structural parameters $\boldsymbol{\alpha}^{(i,j)}$: this gives us the relaxed architecture very close to the target discrete one. To the best of our knowledge, this is the first paper that considers searching the family of models at once and then choosing the suitable complexity value and corresponding model at the inference step. So we do not need to restart the architecture search for models with different complexities. The computational experiment is performed on the Fashion-MNIST [22] and CIFAR-10 [12] dataset.

2 Problem statement

Below we briefly describe the DARTS method and our approach based on it. Given a classification problem with a dataset $\mathfrak{D} = (\mathbf{X}, \mathbf{y})$. For each object $\mathbf{x} \in \mathbf{X}$ there is a label $y \in \mathbf{y}$.

The approach presented in DARTS considers a neural architecture as a sequence of repeated submodels, called *cells*. Each cell has an identical structure


$x^{(0)}$
softmax($\boldsymbol{\alpha}^{(0,1)}$)
$x^{(1)}$
softmax($\boldsymbol{\alpha}^{(0,2)}$)
softmax($\boldsymbol{\alpha}^{(1,2)}$)
$x^{(2)}$
softmax($\boldsymbol{\alpha}^{(0,3)}$)
softmax($\boldsymbol{\alpha}^{(2,3)}$)
$x^{(3)}$
$x^{(0)}(\lambda)$
~GS($\boldsymbol{\alpha}^{(0,1)}(\lambda)$)
~GS($\boldsymbol{\alpha}^{(0,2)}(\lambda)$)
$x^{(1)}(\lambda)$
~GS($\boldsymbol{\alpha}^{(1,2)}(\lambda)$)
~GS($\boldsymbol{\alpha}^{(0,3)}(\lambda)$)
$x^{(2)}(\lambda)$
~GS($\boldsymbol{\alpha}^{(2,3)}(\lambda)$)
$x^{(3)}(\lambda)$

(a) (b)

Fig. 1: An overview of DARTS (a) and proposed method DARTS-CC (b).

but has different parameter values [25]. It is represented as a directed graph. The graph consists of vertices and edges between them. Formally, there is a set of vertices $V = \{1, \dots, N\}$ and a set of edges $E = \{(i, j) \in V \times V \mid i < j\}$, where N is the number of vertices. For each edge (i, j) between vertices i and j there is a non-linear function, called an *operation* $\mathbf{g}^{(i,j)}$ from the vector $\vec{\mathbf{g}}^{(i,j)}$ of all the available non-linear functions for the target edge (i, j). The values in each of the intermediate nodes $\mathbf{x}^{(j)}$ are defined through the values in the nodes with the lower number i:

$$\mathbf{x}^{(j)} = \sum_{(i,j)\in E} \mathbf{g}^{(i,j)}(\mathbf{x}^{(i)}), \tag{1}$$

where $\mathbf{x} = \mathbf{x}^{(0)}$.

The task of architecture search is to choose the non-linear operations $\mathbf{g}^{(i,j)}$ for each edge in the cell. In order to reduce the discrete optimization problem to the continuous optimization problem introduce a mixed operation for each edge (i, j):

$$\hat{\mathbf{g}}^{(i,j)}(\mathbf{x}^{(i)}) = \langle \mathbf{softmax}(\boldsymbol{\alpha}^{(i,j)}), \vec{\mathbf{g}}^{(i,j)}(\mathbf{x}^{(i)}) \rangle, \tag{2}$$

where $\boldsymbol{\alpha}^{(i,j)}$ is a structural parameter that assigns impact of each non-linear operation $\mathbf{g}^{(i,j)}$. Thus, each edge (i, j) is assigned a vector $\boldsymbol{\alpha}^{(i,j)}$ of dimension equal to $\dim \vec{\mathbf{g}}^{(i,j)}$. Let $\boldsymbol{\alpha} \in \mathbb{R}^s$ be a concatenation of all the structural parameter vectors $\boldsymbol{\alpha}^{(i,j)}$.

Split the dataset $\mathfrak{D}$ into train and validation parts: $\mathfrak{D} = \mathfrak{D}_{\text{train}} \sqcup \mathfrak{D}_{\text{val}}$. Formulate a two-level optimization problem:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^s} \mathcal{L}_{\text{val}}(\mathbf{w}^*, \boldsymbol{\alpha}), \tag{3}$$

$$\text{s.t.} \quad \mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha}). \tag{4}$$

Here $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{train}}$ are the cross-entropy loss functions of the model on the validation $\mathfrak{D}_{\text{val}}$ and on the train dataset $\mathfrak{D}_{\text{train}}$, respectively, $\mathbf{w} \in \mathbb{R}^n$ is the vector of model parameters.

2.1 Architecture complexity control

Our modification of the DARTS method consists of two parts. First, similar to [3] instead of using softmax operation in mixing operations $\hat{\mathbf{g}}^{(i,j)}$ we employ Gumbel-softmax distribution. We sample edge values at each training step from it:

$$\hat{\mathbf{g}}^{(i,j)}(\mathbf{x}^{(i)}) = \langle \boldsymbol{\gamma}^{(i,j)}, \vec{\mathbf{g}}^{(i,j)}(\mathbf{x}^{(i)})\rangle, \quad \boldsymbol{\gamma}^{(i,j)} \sim \mathcal{GS}(\mathbf{exp}(\boldsymbol{\alpha}^{(i,j)}), t), \tag{5}$$

where $\mathcal{GS}$ is a Gumbel-softmax distribution [10], t controls the temperature of the distribution. With $t \to 0$ the distribution tends to be similar to the discrete, which allow us to obtain a relaxed version of the architecture close to the discrete one.

Define an *architecture complexity* as an amount of parameters that the corresponding model has. In order to control it modify the loss function $\mathcal{L}_{\text{val}}$:

$$\mathcal{L}'_{\text{val}} = \mathsf{E}_{\boldsymbol{\gamma}} \left(\mathcal{L}_{\text{val}} + \lambda \sum_{(i,j)\in E} \langle \boldsymbol{\gamma}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle \right), \tag{6}$$

where $\boldsymbol{\gamma}$ is a concatenation of all the $\boldsymbol{\gamma}^{(i,j)}$ vectors, $(i,j) \in E$, $\mathbf{n}$ is a vector of numbers of parameters for each operation $\mathbf{g}^{(i,j)} \in \vec{\mathbf{g}}^{(i,j)}$, λ is a regularization coefficient.

The following theorem shows that we get an optimization close to the discrete one when the temperature tends to zero.

Theorem 1 *Let for each $\mathbf{x} \in \mathbf{X}$, for each $\mathbf{w} \in \mathbb{R}^n$ the function $\mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\gamma})$ is continuous by $\boldsymbol{\gamma}$. Then the following expression is true:*

$$\lim_{t\to+0} \mathsf{E}_{\boldsymbol{\gamma}} \left(\mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\gamma}) + \lambda \sum_{(i,j)\in E} \langle \boldsymbol{\gamma}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle \right) =$$
$$\mathsf{E}_{\tilde{\boldsymbol{\gamma}}} \left(\mathcal{L}_{\text{val}}(\mathbf{w}, \tilde{\boldsymbol{\gamma}}) + \lambda \sum_{(i,j)\in E} \langle \tilde{\boldsymbol{\gamma}}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle \right) =$$
$$\sum_{1\le k^{(i,j)}\le \dim \vec{\mathbf{g}}^{(i,j)},\ (i,j)\in E} \mathcal{L}_{\text{val}}(\mathbf{w}, [\mathbf{e}^{(i,j)}(k^{(i,j)})]) \prod_{(l,m)\in E} \text{softmax}(\boldsymbol{\alpha}^{(l,m)})_{k^{(i,j)}} +$$
$$\lambda \sum_{(i,j)\in E} \langle \mathbf{softmax}(\boldsymbol{\alpha}^{(i,j)}), \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle, \tag{7}$$

where $\mathbf{e}^{(i,j)}$ is a one-hot vector:

$$\mathbf{e}^{(i,j)}(k) \in \mathbb{R}^{\dim \vec{\mathbf{g}}^{(i,j)}}, \quad 1 \le k \le \dim \vec{\mathbf{g}}^{(i,j)}, \quad e_k^{(i,j)}(k) = 1, \quad e_m^{(i,j)}(k) = 0,\ m \ne k,$$

vector $[\mathbf{e}^{(i,j)}(k^{(i,j)})]$ *is a concatenation of vectors* $\mathbf{e}^{(i,j)}(k^{(i,j)})$ *over all edges* $(i,j) \in E$*, vector* $\tilde{\boldsymbol{\gamma}}$ *is a concatenation of the following multinomially distributed variables* $\tilde{\boldsymbol{\gamma}}^{(i,j)}$*:*

$$p\left(\tilde{\boldsymbol{\gamma}}^{(i,j)} = \mathbf{e}^{(i,j)}(k)\right) = \frac{\exp(\alpha_k^{(i,j)})}{\sum_{m=1}^{\dim \vec{\mathbf{g}}^{(i,j)}} \exp(\alpha_m^{(i,j)})} = \text{softmax}(\boldsymbol{\alpha}^{(i,j)})_k. \tag{8}$$

In our notation $\text{softmax}(\boldsymbol{\alpha}^{(i,j)})_k$ *is the k-th component of the vector* $\mathbf{softmax}(\boldsymbol{\alpha}^{(i,j)})$.

Proof. From zero temperature property of Gumbel-Softmax distribution [18] it follows that:

$$\boldsymbol{\gamma}^{(i,j)} \xrightarrow[t\to+0]{\text{a.s.}} \tilde{\boldsymbol{\gamma}}^{(i,j)},$$

Then, by the Mann-Wald theorem:

$$\mathcal{L}_{\text{val}}\left(\mathbf{w}, \boldsymbol{\gamma}\right) \xrightarrow[t\to+0]{\text{a.s.}} \mathcal{L}_{\text{val}}\left(\mathbf{w}, \tilde{\boldsymbol{\gamma}}\right),$$

$$\langle \boldsymbol{\gamma}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle \xrightarrow[t\to+0]{\text{a.s.}} \langle \tilde{\boldsymbol{\gamma}}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle.$$

Since the vector $\boldsymbol{\gamma}^{(i,j)}$ is an element of a simplex, then $\boldsymbol{\gamma}$ is an element of a compact set. According to the extreme value theorem, $|\mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\gamma})|$ is bounded on it. Similarly, $|\langle \boldsymbol{\gamma}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle|$ as a function of $\boldsymbol{\gamma}^{(i,j)}$ is also bounded on it. Using dominated convergence theorem, we can swap mathematical expectation and limit:

$$\lim_{t\to+0} \mathsf{E}_{\boldsymbol{\gamma}} \mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\gamma}) = \mathsf{E}_{\tilde{\boldsymbol{\gamma}}} \mathcal{L}_{\text{val}}(\mathbf{w}, \tilde{\boldsymbol{\gamma}}), \tag{9}$$

$$\lim_{t\to+0} \mathsf{E}_{\boldsymbol{\gamma}^{(i,j)}} \langle \boldsymbol{\gamma}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle = \mathsf{E}_{\tilde{\boldsymbol{\gamma}}^{(i,j)}} \langle \tilde{\boldsymbol{\gamma}}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle. \tag{10}$$

Write down the mathematical expectation (9) in explicit form, using (8):

$$\begin{aligned} &\mathsf{E}_{\tilde{\boldsymbol{\gamma}}} \mathcal{L}_{\text{val}}(\mathbf{w}, \tilde{\boldsymbol{\gamma}}) = \\ &\sum_{1 \le k^{(i,j)} \le \dim \vec{\mathbf{g}}^{(i,j)},\ (i,j)\in E} \mathcal{L}_{\text{val}}(\mathbf{w}, [\mathbf{e}^{(i,j)}(k^{(i,j)})]) p(\tilde{\boldsymbol{\gamma}} = [\mathbf{e}^{(i,j)}(k^{(i,j)})]) = \\ &\sum_{1 \le k^{(i,j)} \le \dim \vec{\mathbf{g}}^{(i,j)},\ (i,j)\in E} \mathcal{L}_{\text{val}}(\mathbf{w}, [\mathbf{e}^{(i,j)}(k^{(i,j)})]) \prod_{(l,m)\in E} \text{softmax}(\boldsymbol{\alpha}^{(l,m)})_{k^{(i,j)}}. \end{aligned} \tag{11}$$

Do the same with the mathematical expectation (10):

$$\begin{aligned} \mathsf{E}_{\tilde{\boldsymbol{\gamma}}^{(i,j)}} \langle \tilde{\boldsymbol{\gamma}}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle = \langle \mathsf{E}_{\tilde{\boldsymbol{\gamma}}^{(i,j)}} \tilde{\boldsymbol{\gamma}}^{(i,j)}, \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle = \\ \langle \mathbf{softmax}(\boldsymbol{\alpha}^{(i,j)}), \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle. \end{aligned} \tag{12}$$

From (9), (10), (11), and (12) follows (7).

To reduce stochasticity in our experiments we use a loss function with simplified regularization term:

$$\mathcal{L}'_{\text{val}} = \mathcal{L}_{\text{val}} + \lambda \sum_{(i,j)\in E} \langle \mathbf{softmax}(\boldsymbol{\gamma}^{(i,j)}), \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle, \tag{13}$$

Note that both the original loss function (6) and its modified version (13) become equivalent to the discrete optimization with $t \to +0$.

In order to control the model complexity after the structure optimization, we employ the concept of hypernetwork. Let Λ be a set of the regularization parameter λ values. A hypernetwork is a parametric mapping from the set Λ to the set of model's structural parameters $\mathbb{R}^s$ [7]:

$$\boldsymbol{\alpha} : \Lambda \times \mathbb{R}^u \to \mathbb{R}^s,$$

where $\mathbb{R}^u$ hypernetwork parameter space.

Instead of using a fixed structural parameters vector $\boldsymbol{\alpha}^{(i,j)}$, we redefine it as a hypernetwork:

$$\boldsymbol{\alpha}^{(i,j)} = \boldsymbol{\alpha}^{(i,j)}(\lambda, \boldsymbol{a}^{(i,j)}), \quad \lambda \in \Lambda,$$

where $\mathbf{a}^{(i,j)}$ is a vector of the parameters of the $\boldsymbol{\alpha}^{(i,j)}$ function.

In this paper each function $\boldsymbol{\alpha}^{(i,j)}, (i,j) \in E$ is a piecewise linear function:

$$\boldsymbol{\alpha}^{(i,j)}(\lambda, \mathbf{a}^{(i,j)}) = \sum_{k=0}^{N-1} \Bigg(\frac{\lambda - r_k}{r_{k+1} - r_k} \boldsymbol{a}_k^{(i,j)} + \left(1 - \frac{\lambda - r_k}{r_{k+1} - r_k}\right) \boldsymbol{a}_{k+1}^{(i,j)} \Bigg) I\left[\lambda \in [r_k, r_{k+1}]\right], \tag{14}$$

where $\boldsymbol{a}_k^{(i,j)}$ are the parameters of the function, $r_k \in \mathbb{R}$ are the limits of each linear part of the function, I is an indicator function. For better model fitting, we also use a hypernetwork piecewise linear layer for the last model layer.

The final optimization function is the following:

$$\min_{\boldsymbol{a}\in\mathbb{R}^u} \mathsf{E}_{\lambda \in p(\lambda)} \Bigg(\mathsf{E}_{\boldsymbol{\gamma}^{(i,j)} \sim \mathcal{GS}(\boldsymbol{\alpha}^{(i,j)},\lambda)} \mathcal{L}_{\text{val}}(\mathbf{w}^*, \boldsymbol{\gamma}) + \lambda \sum_{(i,j)\in E} \langle \mathbf{softmax}(\boldsymbol{\alpha}^{(i,j)}(\lambda, \mathbf{a})), \mathbf{n}(\vec{\mathbf{g}}^{(i,j)})\rangle \Bigg), \tag{15}$$

$$\text{s.t.} \quad \mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \mathsf{E}_{\lambda \in p(\lambda)} \Big(\mathsf{E}_{\boldsymbol{\gamma}^{(i,j)} \sim \mathcal{GS}(\boldsymbol{\alpha}^{(i,j)},\lambda)} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\gamma}) \Big), \tag{16}$$

where $p(\lambda)$ is a predefined distribution over Λ, $\mathbf{a} \in \mathbb{R}^u$ is a concatenation of all the parameters of the functions $\boldsymbol{\alpha}^{(i,j)}$, $\boldsymbol{\gamma}$ is a concatenation of all vectors $\boldsymbol{\gamma}^{(i,j)}$.

The optimization algorithm is shown in Fig. 2.

Algorithm 1 DARTS-CC

```
1: Initialize $\mathbf{a} \in \mathbb{R}^u, \mathbf{w} \in \mathbb{R}^n$
2: while not converged do
3:     Sample $\lambda \sim p(\lambda)$
4:     Sample $\boldsymbol{\gamma}^{(i,j)} \sim \mathcal{GS}(\boldsymbol{\alpha}^{(i,j)}, \lambda)$ for each $(i,j) \in E$
5:     Update $\mathbf{w}$ using optimization (16)
6:     Update $\mathbf{a}$ using optimization (15)
7: end while
8: return the final architecture from learned $\mathbf{a}$.
```

Algorithm 2 DARTS

```
1: Initialize $\boldsymbol{\alpha} \in \mathbb{R}^u, \mathbf{w} \in \mathbb{R}^n$
2: while not converged do
3:     Update $\mathbf{w}$ using optimization (4)
4:     Update $\boldsymbol{\alpha}$ using optimization (3)
5: end while
6: return the final architecture from learned $\boldsymbol{\alpha}$.
```

Fig. 2: An algorithm for the proposed method DARTS-CC and DARTS.

3 Computational experiment

The purpose of the computational experiment is to analyze the proposed method efficiency for the neural architecture search task [3]. The experiments were conducted on Fashion-MNIST [22] and CIFAR-10 [12] datasets. First, we ran an experiment to search for an architecture consisting of three cells on both datasets. We compared the proposed method with DARTS and random architecture search. After that, we ran an experiment with a large-scale architecture on CIFAR-10 and compared the results with other existing neural architecture search methods.

For the architecture search experiments, the architectures were trained for 50 epochs. The parameters of the optimization procedure were similar to those used in [6]: we used SGD for the parameter optimization with a learning rate decreasing from 0.025 to 0.001 and momentum set to 0.9. We used weight decay $3 \cdot 10^{-4}$ for the model parameters and 10^{-3} for the structural parameters. Similar to DARTS, after the architecture search step, we retrained the model from scratch with obtained architecture. During retraining, we also randomly dropped operations with the probability increasing from 0.0 to 0.2.

3.1 Search of the small architectures

During the experiments with architecture containing three cells, we did not use a *cutout* procedure: a heuristic of pruning uninformative operations from the retrained model. For all the architecture search steps, we used a batch size equal to 64. The final models were trained for 100 epochs with batch size equal to 96 and learning rate decreasing from 0.025 to 0.0.

[3] The source code for the computational experiment can be found at https://github.com/Intelligent-Systems-Phystech/DARTS-CC.

As a set Λ we used a set of values $[10^{-10}, 10^{-6}]$, $\log_{10} \lambda \sim \mathcal{U}(-10, -6)$. for Fashion-MNIST and $[10^{-8}, 10^{-4}]$, $\log_{10} \lambda \sim \mathcal{U}(-4, -8)$. We calibrated the intervals for the λ values in order to obtain architectures with approximately similar amount of parameters to architectures obtained by DARTS. During model training we used Gumbel-Softmax distribution with temperature t decreasing from 1 to 0.2.

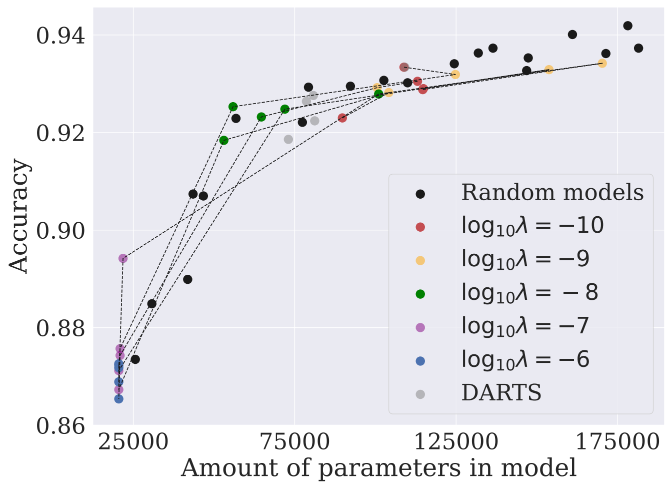


Fig. 3: The comparison of the proposed method with DARTS for the Fashion-MNIST dataset: both methods show similar performance. The proposed method allows to vary λ parameter to control the model architecture: the higher λ we use the simpler model we obtain. The lines connect models obtained from the same experiment run.

The Figure 3 shows a dependency of the obtained models' performance on the parameter amount. Both the proposed method and DARTS gave model architectures with suboptimal performance on the Fashion-MNIST dataset. For both the proposed method and DARTS, we ran the experiment 5 times. For the proposed method we evaluated the architectures obtained from different λ values: $\lambda \in \{10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}\}$. As we can see, the proposed method gives a comparable performance to the DARTS approach. However, it allows calibrating the model complexity to find a trade-off between desired performance and model complexity.

The similar results for the CIFAR-10 dataset can be found in Figure 4. As we can see, the proposed method and DARTS show similar performance on both datasets. The ability to control the model complexity enables us to obtain models with different structures at once and tune the final models based on the desired model complexity.

The cells obtained for different λ are shown in Figure 5 for the Fashion-MNIST dataset. As we can see, the resulting cells tend to have a less complex structure with increasing λ parameter.

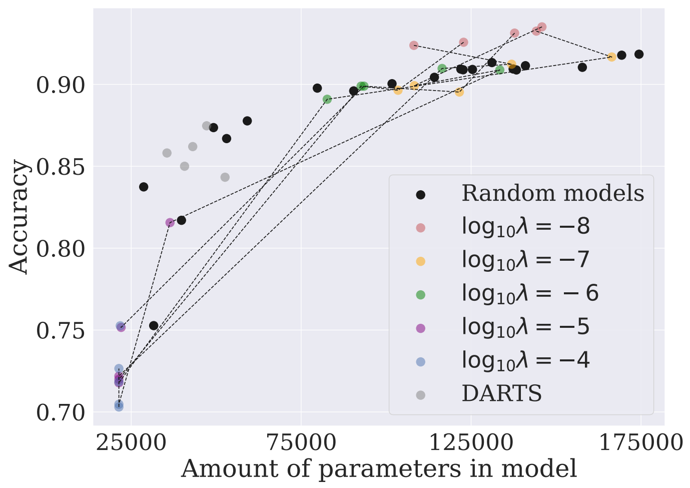


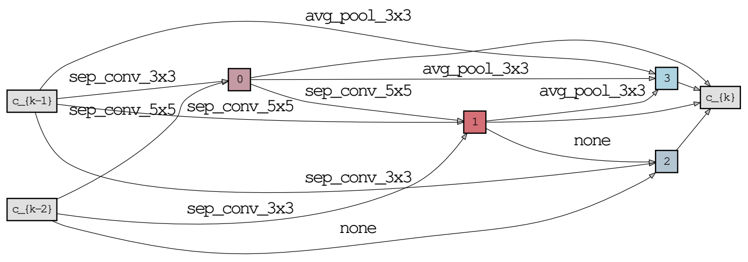
Fig. 4: The comparison of the proposed method with DARTS for the CIFAR-10 dataset.

3.2 Large-scale architecture search

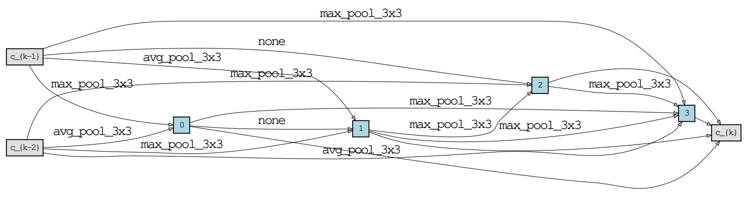
In order to compare the proposed method with different neural architecture search methods, we conducted an additional experiment on the CIFAR-10 dataset. For this experiment, we used settings similar to those described in [6]. We used a model with eight cells during architecture search and with 20 cells during the retraining model from scratch. The final models were trained for 600 epochs with batch size equal to 72 and learning rate decreasing from 0.025 to 0.0.

To evaluate the proposed method, we ran the experiment 3 times. As a set Λ we used a set of values $\lambda \in [10^{-10}, 10^{-6}]$, $\log_{10} \lambda \sim \mathcal{U}(-10, -6)$. For the proposed method, we evaluated the architectures obtained from the highest and the lower values of $\lambda : \lambda \in \{10^{-10}, 10^{-6}\}$. The results of the experiments are shown in table 1. As we can see, the resulting models achieve a performance comparable to other neural architecture search methods.

Based on the obtained results, we can conclude that the proposed method allows us to obtain models with different model complexity per one architec-



(a)



(b)

Fig. 5: An example of cells obtained on the Fashion-MNIST dataset for $\lambda = 10^{-10}$ (a) and $\lambda = 10^{-6}$ (b). The colors of each node in the cell vary from blue to red. The closer to the red color, the greater the total number of parameters of the incoming edges. Here c_{k-1} and c_{k-2} are the outputs of cells $k-1$ and $k-2$, respectively, c_{k} is the output node of cell k.

ture search optimization procedure. The resulting architectures give us a rather good performance comparable to architectures of other neural architecture search methods. At the same time, our model enables us to control the model complexity and obtain different models based on the desired computational budget.

4 Conclusion

In this paper, we considered the problem of deep learning model selection with model complexity control. To control the model complexity, we proposed a modification of the differential architecture search algorithm called DARTS-CC. We considered the structural parameters of the architecture as a function of the model complexity parameter. To evaluate the performance of the proposed method, we conducted experiments on the Fashion-MNIST and CIFAR-10 datasets and compared the obtained architecture with architectures gathered from the DARTS method. The results showed the comparable performance of the method. The proposed method allows us to control the trade-off between model characteristics: the model performance and the model complexity. Future research plans include researching the robustness of the obtained architecture and properties of the proposed optimization problem.

Table 1: Top-1 Accuracy on CIFAR-10 dataset. The baseline results are taken from [20].

Architecture	Accuracy(%)	Params(M)
DenseNet-BC[9]	96.54	25.6
NASNetA + cutout[25]	97.35	3.3
AmoebaNet-B + cutout[19]	97.87	34.9
PNAS[15]	96.59 ± 0.09	3.2
NAONet[17]	96.82	10.6
SMASHv2[2]	95.97	16
SETN + cutout[5]	97.31	4.6
GDAS + cutout[6]	96.25	2.5
DARTS(2nd order) + cutout[16]	97.24 ± 0.09	3.3
SNAS (mild) + cutout[23]	97.02	2.9
PR-DARTS DL1 + cutout[13]	97.26 ± 0.12	3.2
PC-DARTS + cutout[24]	97.43	3.6
PDARTS + cutout[4]	97.50	3.4
Amended-DARTS S1 + cutout[1]	97.19 ± 0.21	3.5
DARTS+ with cutout[14]	97.68	3.7
DARTS-CC + cutout, $\lambda = 10^{-10}$	97.13 ± 0.08	4.4 ± 0.5
DARTS-CC + cutout, $\lambda = 10^{-6}$	96.58 ± 0.38	2.2 ± 0.2

References

1. Bi, K., Hu, C., Xie, L., Chen, X., Wei, L., Tian, Q.: Stabilizing darts with amended gradient estimation on architectural parameters. arXiv preprint arXiv:1910.11831 (2019)
2. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344 (2017)
3. Chang, J., Zhang, X., Guo, Y., Meng, G., Xiang, S., Pan, C.: Differentiable architecture search with ensemble gumbel-softmax. arXiv preprint arXiv:1905.01786 (2019)
4. 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. pp. 1294–1303 (2019)
5. Dong, X., Yang, Y.: One-shot neural architecture search via self-evaluated template network. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3681–3690 (2019)
6. Dong, X., Yang, Y.: Searching for a robust neural architecture in four gpu hours. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1761–1770 (2019)
7. Grebenkova, O., Bakhteev, O.Y., Strijov, V.: Variational deep learning model optimization with complexity control
8. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. CoRR **abs/1609.09106** (2016), http://arxiv.org/abs/1609.09106

9. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
10. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
11. Jin, X., Wang, J., Slocum, J., 0001, M.H.Y., Dai, S., Yan, S., Feng, J.: Rc-darts: Resource constrained differentiable architecture search. CoRR **abs/1912.12814** (2019), http://arxiv.org/abs/1912.12814
12. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
13. Laube, K.A., Zell, A.: Prune and replace nas. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). pp. 915–921. IEEE (2019)
14. Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., Li, Z.: Darts+: Improved differentiable architecture search with early stopping. arXiv preprint arXiv:1909.06035 (2019)
15. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV). pp. 19–34 (2018)
16. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. CoRR **abs/1806.09055** (2018), http://arxiv.org/abs/1806.09055
17. Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.Y.: Neural architecture optimization. arXiv preprint arXiv:1808.07233 (2018)
18. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016)
19. 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. vol. 33, pp. 4780–4789 (2019)
20. Tanveer, M.S., Khan, M.U.K., Kyung, C.M.: Fine-tuning darts for image classification. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 4789–4796. IEEE (2021)
21. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10734–10742 (2019)
22. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
23. Xie, S., Zheng, H., Liu, C., Lin, L.: Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926 (2018)
24. Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.J., Tian, Q., Xiong, H.: Pc-darts: Partial channel connections for memory-efficient architecture search. arXiv preprint arXiv:1907.05737 (2019)
25. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710 (2018)