Concordant Multi-domain Architecture Search

K. D. Yakovlev[1],, and O. Y. Bakhteev[1,2]

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

Abstract. The paper investigates the problem of selection the structure of a deep learning model for multi-domain data. The model is a superposition of functions differentiable by parameters. We propose to consider the task as a multi-modeling task: a separate structure is optimized for each domain. Define a model structure as a set of vectors that define the contribution of various nonlinear operations to the final deep learning model. We consider two methods for regularization of the architecture: structural regularization and contrastive regularization. The paper shows that the proposed optimization problem makes it possible to find a compromise between the complexity of the final model and its predictive characteristics during inference on various domains.

Keywords: differential architecture search · deep learning · neural networks · model complexity control · multi-domain architecture search

1 Introduction

In this paper, the problem of selection the structure of a deep learning model for multi-domain data is considered. The model is understood as a superposition of functions that solves the problem of classification or regression [1]. The search for the model architecture is understood as the search for optimal structural parameters. Relaxation refers to the translation of a set of permissible structural parameters from discrete to continuous. The differentiable algorithm for searching for the DARTS [12] architecture is used as the basic algorithm. It solves the problem of searching for the architecture of the model by translating the search space of structural parameters from a discrete to a continuous representation. It is proposed to use gradient optimization methods. They use less computational resources than methods operating on a discrete set of structural parameters. This algorithm works with both convolutional and recurrent neural networks.

There are approaches that use experts to search for architecture. In [14], the search space is the same as in [6]. The importance of the mappings in the mix is regulated by experts. During the optimization process, the number of experts involved in the mix gradually decreases. This approach has a number of advantages. Firstly, the resulting discrete architecture will not be much mixed in quality relative to the optimal continuous one. Secondly, the effect of retraining decreases. Thirdly, the optimization process is accelerated due to the fact that the number of experts is reduced.

In this paper, we propose two types of regularization: structural regularization and contrastive regularization.

Structural regularization requires the structures of different domains to be pairwise similar in terms of Jensen-Shannon divergence. There are approaches that formalize this concept. In [20], the problem of predicting a complex target variable is investigated. A matching function is introduced, which is responsible for the proximity of latent representations of the independent and target variable. Then the prediction problem is solved in the hidden space.

We compare the proposed method with MPNAS[17] on different MNIST[11] dataset.

2 Problem statement

Due to the fact that gradient-based architecture search approach does not require large amount of computational resources compared to Reinforcement Learning based approach [16], we consider DARTS[12] as a baseline.

2.1 DARTS overview

Given a dataset $\mathfrak{D} = (\mathbf{X}, \mathbf{y})$. Each object $\mathbf{x} \in \mathbf{X}$ is assigned a target variable $y \in \mathbf{y}$.

In the approach described in DARTS [12], the architecture is considered, which is a sequence of *cells*. All cells have the same structure, but different parameter values. A cell is an oriented acyclic graph. Formally, there is a set of vertices $V = \{1, \dots, N\}$ and a set of edges $E = \{(i, j) \in V \times V : i < j\}$, where N is some natural number. Each edge of (i, j) corresponds to a nonlinear mapping $\mathbf{g}^{(i,j)}$, which is a component of the vector $\vec{\mathbf{g}}^{(i,j)}$.

The values that are calculated in the j-th node are calculated through values with smaller numbers:

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

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

The task of architecture search is to find nonlinear mappings $\mathbf{g}^{(i,j)}$ for each edge of the cell $(i, j) \in E$. This discrete optimization problem is reduced to a continuous one. To do this, the concept of *mixed operation* is introduced:

$$\mathbf{m}^{(i,j)} = \langle \mathbf{softmax}(\boldsymbol{\alpha}^{(i,j)}), \vec{\mathbf{g}}^{(i,j)}(\mathbf{x}^{(i)}) \rangle, \tag{1}$$

where $\boldsymbol{\alpha}^{(i,j)}$ are *structural parameters* that determine the importance of each mapping $\mathbf{g}^{(i,j)}$. Thus, each edge $(i, j) \in E$ corresponds to the vector $\boldsymbol{\alpha}^{(i,j)} \in \mathbb{R}^{\dim \vec{\mathbf{g}}^{(i,j)}}$. Let $\boldsymbol{\alpha}$ be the concatenation of vectors $\boldsymbol{\alpha}^{(i,j)}$ over all edges $(i, j) \in E$.

Let the dataset $\mathfrak{D}$ consist of a disjoint union of training and validation datasets: $\mathfrak{D} = \mathfrak{D}_{\text{train}} \sqcup \mathfrak{D}_{\text{valid}}$. The structural parameters $\boldsymbol{\alpha} \in \mathbb{R}^u$ are found from the following two-level optimization problem:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^u} \mathcal{L}_{\text{valid}}(\mathbf{w}^*, \boldsymbol{\alpha}), \tag{2}$$

$$s.t., \quad \mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha}). \tag{3}$$

Here $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{valid}}$ are the cross-entropy loss function on samples $\mathfrak{D}_{\text{train}}$ and $\mathfrak{D}_{\text{valid}}$ respectively, $\mathbf{w} \in \mathbb{R}^n$ is the vector of model parameters. The resulting operation is $\vec{\mathbf{g}}^{(i,j)}_{k^*}$, where $k^* = \arg\max_{1\le k\le \dim \vec{\mathbf{g}}^{(i,j)}} \alpha^{(i,j)}_k$.

The optimization algorithm is the following:

Algorithm 1 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 (3)
4: Update $\boldsymbol{\alpha}$ using optimization (2)
5: **end while**
6: **return** the final architecture from learned $\boldsymbol{\alpha}$.

2.2 Multi Domain architecture search

Given D domains $\{\mathfrak{D}_d\}_{d=1}^D$, $\mathfrak{D}_d = (\mathbf{X}_d, \mathbf{y}_d)$. Let $\mathcal{L}^{(d)}_{\text{train}}$ and $\mathcal{L}^{(d)}_{\text{valid}}$ be loss functions on train and validation datasets of $\mathfrak{D}_d$ respectively. The model has shared parameters $\mathbf{w}$ between domains, but has structural parameters individual for each domain. Now vector of structural parameters takes the following form $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_d]_{d=1}^D$, where $\boldsymbol{\alpha}_d$ is a vector of structural parameters for domain $\mathfrak{D}_d$. Based on [15], we formulate a two-level optimization problem of multi-domain architecture search:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \max_{\boldsymbol{p}} \sum_{d=1}^{D} p_d \mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*, \boldsymbol{\alpha}_d), \tag{4}$$

$$s.t., \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \max_{\boldsymbol{p}} \sum_{d=1}^{D} p_d \mathcal{L}^{(d)}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha}_d), \tag{5}$$

where $p_d \ge 0,\ 1 \le d \le D,\ \sum_{d=1}^D p_d = 1$. Thus, we find an architecture that gives acceptable quality on each domain.

The optimization algorithm is based on [15]:

Algorithm 2 DARTS, multi-domain setup

1: Initialize $\boldsymbol{\alpha} \in \mathbb{R}^u, \mathbf{w} \in \mathbb{R}^n,\ p_d = \frac{1}{D}$ for each $1 \le d \le D, \eta_p$ – learning rate for $\mathbf{p}$.
2: **while** not converged **do**
3: Update $\mathbf{w}$ using optimization (5)
4: Update $\mathbf{p} = \mathbf{softmax}(\eta_p[\hat{\mathcal{L}}_{\text{valid}}(\mathbf{w}, \boldsymbol{\alpha}_d)]_{d=1}^D)$
5: Update $\boldsymbol{\alpha}$ using optimization (4)
6: **end while**
7: **return** the final architecture from learned $\boldsymbol{\alpha}$.

Here $\hat{\mathcal{L}}_{\text{valid}}(\mathbf{w}, \boldsymbol{\alpha})$ is a mini-batch approximation of true validation loss $\mathcal{L}_{\text{valid}}(\mathbf{w}, \boldsymbol{\alpha})$.

2.3 Concordant architectures

The goal is to obtain an architecture that will be used for all domains. Now we will define a resulting architecture. Suppose we have a learned structural parameters for all domains: $\{\boldsymbol{\alpha}_d\}_{d=1}^D$. In resulting architecture there may be no more than $\min(D, \dim \tilde{\mathbf{g}}^{(i,j)})$ chosen operations between nodes i and j. We include operation $\vec{\mathbf{g}}_m^{(i,j)}$ if and only if there exists domain d, for which the following is true $m = \arg\max_{1 \leq k \leq \dim \vec{\mathbf{g}}^{(i,j)}} (\boldsymbol{\alpha}_d^{(i,j)})_k$. When we apply a model to the object $\mathbf{x}$ from domain d, we do a forward pass only through the operations that are chosen for domain d.

2.4 Structural regularization

To get a simpler architecture, we are introducing a regularizer that forces each pair of categorical distributions defined by structural parameters to be close in terms of any distance between two distributions. For concreteness, we use Jensen-Shannon divergence. Formally, the regularizer is the following:

$$R([\boldsymbol{\alpha}_d]_{d=1}^D) = \frac{2}{D(D-1)} \sum_{d=1}^{D} \sum_{d'=1,\ d' \neq d}^{D} \text{JS}(P_d || P_{d'}), \tag{6}$$

where random variable $P_d \sim \text{Categorical}(\textbf{softmax}(\boldsymbol{\alpha}_d))$, $\text{JS}(P||Q)$ is a Jensen-Shannon divergence:

$$\text{JS}(P||Q) = \frac{1}{2}\text{D}_{KL}(P||M) + \frac{1}{2}\text{D}_{KL}(Q||M), \tag{7}$$

where $M = \frac{1}{2}(P+Q)$, $\text{D}_{KL}(P||Q)$ is a Kullback-Leibler divergence.

Therefore, the final two-level optimization problem that includes proposed regularizer will be the following:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \max_{\boldsymbol{p}} \sum_{d=1}^{D} p_d \mathcal{L}_{\text{valid}}^{(d)}(\mathbf{w}^*, \boldsymbol{\alpha}_d) + \beta_{\text{struct}} R([\boldsymbol{\alpha}_d]_{d=1}^D), \tag{8}$$

$$s.t., \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \max_{\boldsymbol{p}} \sum_{d=1}^{D} p_d \mathcal{L}_{\text{train}}^{(d)}(\mathbf{w}, \boldsymbol{\alpha}_d), \tag{9}$$

where $\beta > 0$ is a regularization coefficient. To solve the optimization problem, we will use Algorithm 2.

2.5 Contrastive regularization

The approach is based on triplet loss [2]. The idea is to maximize agreement between hidden representations of the objects from doamins with number d and d'.

Formally, given a model, parametrized by $\mathbf{w}$, structural parameters $\boldsymbol{\alpha}$, and a batch of objects with its labels from each domain d of size b: $\mathcal{S}_b^{(d)} = \{(\mathbf{x}_i^{(d)}, y_i^{(d)})\}_{i=1}^b$. Let $\mathbf{h}(\mathbf{x}, \mathbf{w}, \boldsymbol{\alpha})$ be the vector of all hidden states after each cell.

We define a *triplet* $(\mathbf{x}_i^{(d)}, \mathbf{x}_j^{(d')}, \mathbf{x}_k^{(d')})$ if $y_i^{(d)} = y_j^{(d')},\ y_i^{(d)} \neq y_k^{(d')}$. We use batch hard strategy for triplet mining [8]. Define a triplet loss for two batches:

$$\mathcal{L}_{\text{triplet}}(\mathbf{w}, \boldsymbol{\alpha}, \mathcal{S}_b^{(d)}, \mathcal{S}_b^{(d')}) = \\ \sum_{(i,j,k)\in\mathcal{T}} \max(0, \mu + \|\mathbf{h}(\mathbf{x}_i^{(d)}, \mathbf{w}, \boldsymbol{\alpha}_d) - \mathbf{h}(\mathbf{x}_j^{(d')}, \mathbf{w}, \boldsymbol{\alpha}_{d'})\|_2 - \|\mathbf{h}(\mathbf{x}_i^{(d)}, \mathbf{w}, \boldsymbol{\alpha}_d) - \mathbf{h}(\mathbf{x}_k^{(d')}, \mathbf{w}, \boldsymbol{\alpha}_{d'})\|_2), \tag{10}$$

where $\mathcal{T}$ is a set of all triplets for two given batches $\mathcal{S}_b^{(d)}, \mathcal{S}_b^{(d')}$. The total triplet loss is the following:

$$\mathcal{L}_{\text{triplet}}(\mathbf{w}, \boldsymbol{\alpha}) = \mathsf{E}_{d,d'\sim\text{Cat}(\mathbf{p})} \mathsf{E}_{\mathcal{S}_b^{(d)}, \mathcal{S}_b^{(d')}} \mathcal{L}_{\text{triplet}}(\mathbf{w}, \boldsymbol{\alpha}, \mathcal{S}_b^{(d)}, \mathcal{S}_b^{(d')}). \tag{11}$$

Write down an optimization problem:

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \sum_{d=1}^{D} \sum_{p_d} \mathcal{L}_{\text{valid}}^{(d)}(\mathbf{w}^*, \boldsymbol{\alpha}), \tag{12}$$

$$\text{s.t.} \quad \mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{d=1}^{D} p_d \mathcal{L}_{\text{train}}^{(d)}(\mathbf{w}, \boldsymbol{\alpha}) + \beta_{\text{contr}} \mathcal{L}_{\text{triplet}}(\mathbf{w}, \boldsymbol{\alpha}). \tag{13}$$

3 Theorems(informal)

The following theorem gives us a sufficient match condition. If two close in terms of Jensen-Shannon divergence categorical distributions are sparce enough, then they match.

Theorem 1. *There exists $\varepsilon \in (0, \frac{1}{2n}]$ such that for any two random variables $P \sim \text{Cat}(\boldsymbol{\alpha})$ and $Q \sim \text{Cat}(\boldsymbol{\beta})$, $\dim \boldsymbol{\alpha} = \dim \boldsymbol{\beta} = n$ with the following constrains:*

$$\alpha_k \leq \varepsilon, \quad k \neq i, \tag{14}$$

$$\beta_k \leq \varepsilon, \quad k \neq j, \tag{15}$$

$$\text{JS}(P||Q) \leq \frac{1}{2}\log 2, \tag{16}$$

the following is true:

$$\arg\max_k \alpha_k = \arg\max_k \beta_k.$$

Proof. Take any $\boldsymbol{\alpha},\ \boldsymbol{\beta}$ and $\leq \varepsilon \leq \frac{1}{2n}$. First, prove that when $i \neq j$, from (14), (15) follows that $\mathrm{JS}(P||Q) \geq \frac{1}{2}\log 2$. To do that, rewrite Jensen-Shannon divergence:

$$\mathrm{JS}(P||Q) = \frac{1}{2}(\mathrm{D}_{\mathrm{KL}}(P||M) + \mathrm{D}_{\mathrm{KL}}(Q||M)), \tag{17}$$

where $M = \frac{1}{2}(P+Q)$. Now rewrite KL divergence.

$$\mathrm{D}_{\mathrm{KL}}(P||M) = \sum_k \alpha_k \log \frac{\alpha_k}{\frac{\alpha_k+\beta_k}{2}} = \sum_k \alpha_k \log \alpha_k - \sum_k \alpha_k \log \frac{\alpha_k+\beta_k}{2}. \tag{18}$$

In the same way rewriting $\mathrm{D}_{\mathrm{KL}}(Q||M)$, we obtain:

$$JS(P||Q) = -\sum_k \frac{\alpha_k+\beta_k}{2} \log \frac{\alpha_k+\beta_k}{2} + \frac{1}{2}\sum_k \alpha_k \log \alpha_k + \frac{1}{2}\sum_k \beta_k \log \beta_k \tag{19}$$

Find lower bound for each term in (19).

$$-\sum_k \frac{\alpha_k+\beta_k}{2} \log \frac{\alpha_k+\beta_k}{2} = -\sum_{k\neq i, k\neq j} \frac{\alpha_k+\beta_k}{2} \log \frac{\alpha_k+\beta_k}{2} - \sum_{k\in\{i,j\}} \frac{\alpha_k+\beta_k}{2} \log \frac{\alpha_k+\beta_k}{2} \tag{20}$$

For $k \neq i, j$ the expression $-\frac{\alpha_k+\beta_k}{2} \log \frac{\alpha_k+\beta_k}{2}$ is non-negative. For $k \in \{i, j\}$ we have:

$$\frac{1-\varepsilon(n-1)}{2} \leq \frac{\alpha_k+\beta_k}{2} \leq \frac{1+\varepsilon}{2}. \tag{21}$$

Then for (20) we have the following lower bound:

$$-\sum_k \frac{\alpha_k+\beta_k}{2} \log \frac{\alpha_k+\beta_k}{2} \geq 2 \min_{x\in[\frac{1-\varepsilon(n-1)}{2}, \frac{1+\varepsilon}{2}]} -x\log x = \tag{22}$$

$$\min(-(1-\varepsilon(n-1))\log\frac{1-\varepsilon(n-1)}{2}, -(1+\varepsilon)\log\frac{1+\varepsilon}{2}) \tag{23}$$

It is easy to see that the lower bound (23) tends to $\log 2$ when ε tends to zero. Moreover, lower bound is continuous in the point $\varepsilon = 0$.

Now find a lower bound for the second term of (19):

$$\frac{1}{2}\sum_k \alpha_k \log \alpha_k = \frac{1}{2}\sum_{k\neq i} \alpha_k \log \alpha_k + \frac{1}{2}\alpha_i \log \alpha_i \geq \tag{24}$$

$$\frac{1}{2}(n-1) \min_{x\in[0,\varepsilon]} x\log x + \frac{1}{2} \min_{x\in[1-(n-1)\varepsilon,1]} x\log x. \tag{25}$$

Due to the fact that $x\log x$ is a convex function on $[0, +\infty]$, and it achieves a global minima at the point $x = e^{-1}$. Since $\varepsilon \leq e^{-1}$ for any n, then $\min_{x\in[0,\varepsilon]} x\log x = \varepsilon\log\varepsilon$. Thus, (25) takes the following form:

$$\frac{1}{2}\left(\varepsilon\log\varepsilon + (1-(n-1)\varepsilon)\log(1-(n-1)\varepsilon)\right) \tag{26}$$

Finally,

$$\frac{1}{2}\sum_{k}\alpha_k\log\alpha_k \geq \frac{1}{2}\left(\varepsilon\log\varepsilon + (1-(n-1)\varepsilon)\log(1-(n-1)\varepsilon)\right). \tag{27}$$

The same lower bound is true for the third term of (19). It is easy to see that (27) tends to zero when ε tends to zero and it is a continuous function at the point $\varepsilon = 0$. Finally, we have that the lower bound of $\mathrm{JS}(P||Q)$ tends to $\log 2$ when ε tends to zero and it is continuous at $\varepsilon = 0$. Then there exist $\varepsilon_0 \in (0, \frac{1}{2n}]$ for which any $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, that satisfy the restrictions (14), (15), $i \neq j$, are distant in terms of JS-divergence:

$$\mathrm{JS}(P||Q) > \frac{1}{2}\log 2. \tag{28}$$

Therefore, for ε_0 the theorem is true.

Theorem 2. *Consider an optimization problem for a given* $\mathbf{p}$*:*

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^u}\sum_{d=1}^{D} p_d\mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*, \boldsymbol{\alpha}), \tag{29}$$

$$s.t. \quad \mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n}\sum_{d=1}^{D} p_d\mathcal{L}^{(d)}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha}). \tag{30}$$

Let $\vec{\mathbf{g}}^{(i,j)}_{\text{ext}}$ *be a vector where each mapping from* $\vec{\mathbf{g}}^{(i,j)}$ *is duplicated* D *times. Thus, vector of parameters of corresponding extended model can be written as* $\mathbf{w}_{\text{ext}} = [\mathbf{w}_d]_{d=1}^{D}$*, where* $\mathbf{w}_d$ *corresponds to parameters of mappings of* d*-th submodel.*

Let $\boldsymbol{\alpha}^*_d$ *be the solution of the DARTS optimization problem* (2) *on the* d*-th domain. Let* $\mathbf{w}^*_d$ *be the corresponding solution of* (3)*. Let* $\boldsymbol{\alpha}^*_{ext}$ *be a solution of* (29) *with extended model, and* $\mathbf{w}^*$ *is a corresponding solution of* (30)*. Then the following is true:*

1. $\mathbf{w}^*_{\text{ext}} = [\mathbf{w}^*_d]_{d=1}^{D}$ *is a solution of* (5) *for a given* $\boldsymbol{\alpha}_{\text{ext}} = [[\boldsymbol{\alpha}^*_d, \textit{if } d = d', \textit{else} -\infty]_{d=1}^{D}]_{d'=1}^{D}$.
2. $\sum_{d=1}^{D} p_d\mathcal{L}^{(d)}_{valid}(\mathbf{w}^*, \boldsymbol{\alpha}^*_{\text{ext}}) \leq \sum_{d=1}^{D} p_d\mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*_d, \boldsymbol{\alpha}^*_d)$.

Proof. 1. First note that for a given structural parameters $\boldsymbol{\alpha}_{\text{ext}}$ we have that

$$\mathcal{L}^{(d)}_{\text{train}}(\mathbf{w}_{\text{ext}}, \boldsymbol{\alpha}_{\text{ext}}) = \mathcal{L}_{\text{train}}(\mathbf{w}_d, \boldsymbol{\alpha}^*_d) \tag{31}$$

for any arbitrary vector $\mathbf{w}_{\text{ext}} = [\mathbf{w}_d]_{d=1}^{D}$. It follows from the fact that output values of each node (1) are equal. Thus, the objective from (30) can be written as follows:

$$\sum_{d=1}^{D} p_d\mathcal{L}^{(d)}_{\text{train}}(\mathbf{w}_{\text{ext}}, \boldsymbol{\alpha}_{\text{ext}}) = \sum_{d=1}^{D} p_d\mathcal{L}^{(d)}_{\text{train}}(\mathbf{w}_d, \boldsymbol{\alpha}^*_d). \tag{32}$$

Minimization of (32) gives us the solution $\mathbf{w}^*_{\text{ext}} = [\mathbf{w}^*_d]_{d=1}^{D}$.

2. Since $\boldsymbol{\alpha}^*_{\text{ext}}$ is a solution of (29), we have the following inequality:

$$\sum_{d=1}^{D} p_d \mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*, \boldsymbol{\alpha}^*_{\text{ext}}) \leq \sum_{d=1}^{D} p_d \mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*_{\text{ext}}, \boldsymbol{\alpha}_{\text{ext}}). \tag{33}$$

Using (31), we rewrite the right term of (33) and obtain the desired inequality:

$$\sum_{d=1}^{D} p_d \mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*, \boldsymbol{\alpha}^*_{\text{ext}}) \leq \sum_{d=1}^{D} p_d \mathcal{L}^{(d)}_{\text{valid}}(\mathbf{w}^*_d, \boldsymbol{\alpha}^*_d). \tag{34}$$

Theorem 3. *(Hypothesis) Final architecture obtained from* (12) *has more parameters than architecture obtained from* (8)*.*

4 Computational experiment

4.1 Basic experiment

In this experiment we compared the performance of the following models on domains: MNIST-0 and MNIST-180:

- Random Search DARTS: the model with random DARTS architecture was trained on the full dataset (MNIST-0 and MNIST-180) and evaluated on the full dataset.
- DARTS: we performed an architecture search on a single dataset and evaluated on both datasets separately.
- DARTS-full: we perform a DARTS architecture search on both datasets and evaluated on both datasets.
- MdDARTS: we performed an architecture search on both datasets in the proposed multi-domain setup. We evaluated the model on both datasets separately.

	MNIST-0	MNIST-180	MNIST-0+180
Random search, seed=0	97.10	97.60	97.35
Random search, seed=1	95.46	95.80	95.63
Random search, seed=2	92.29	93.46	92.86
DARTS; MNIST-0	95.81	36.30	–
DARTS; MNIST-180	47.31	97.55	–
DARTS-full	96.01	96.44	96.23
MdDARTS	93.93	96.99	–
MPNAS	95.69	94.65	–

Table 1: Caption

	MNIST-0	MNIST-120	MNIST-240	MNIST-0+120+240
Random search, seed=0				
Random search, seed=1				
Random search, seed=2				
MdDARTS	95.95	96.45	91.25	–
MPNAS				

Table 2: Caption

5 Notation

- $\mathfrak{D} = \mathfrak{D}_{\text{train}} \sqcup \mathfrak{D}_{\text{val}}$ – dataset
- $\mathbf{x} \in \mathbf{X}$ – object from object space
- $y \in \mathbf{Y}$ – target from target space
- V – set of vertices
- E – set of edges
- $\mathbf{x}^{(i)}$ – intermediate representation of the object $\mathbf{x}$ corresponding to the node i
- $\mathbf{g}^{(i,j)}$ – nonlinear mapping corresponding the edge (i, j)
- $\vec{\mathbf{g}}^{(i,j)}$ – vector of all available mappings for the edge (i, j)
- $\boldsymbol{\alpha}^{(i,j)}$ – structural parameters for the edge (i, j)
- $\mathbf{m}^{(i,j)}$ – mixed operation
- $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ – loss functions on $\mathfrak{D}_{\text{train}}$ and $\mathfrak{D}_{\text{val}}$ respectively
- $\mathbf{w} \in \mathbb{R}^n$ – vector of model parameters
- $\vec{\mathbf{g}}_k^{(i,j)}$ – vector of experts having the same type (ex. dilated convolution), but having different complexity
- $\boldsymbol{\gamma}^{(i,j)}$ – structural parameters
- $\boldsymbol{\gamma}_{\text{exp}}^{(i,j)}$ – structural parameters from N - 1 dimensional simplex. They control importance of each of N experts
- $\boldsymbol{\gamma}_{\text{comp}}^{(i,j),k}$ – structural parameters from dim $\vec{\mathbf{g}}_k^{(i,j)}$ - 1 dimensional simplex. They control importance of each mapping among $\vec{\mathbf{g}}_k^{(i,j)}$
- $\Lambda \subset \mathbb{R}$ – a set of regularization parameters λ
- $\mathbf{h}_{\text{exp}}^{(i,j)}(\mathbf{x}, \mathbf{s}^{(i,j)}, \lambda)$ – hypernetwork, where $\mathbf{s}_{\text{exp}}^{(i,j)}$ is a vector of its parameters
- $\mathbf{h}_k^{(i,j)}(\mathbf{x}, \mathbf{s}_k^{(i,j)}, \lambda)$ – hypernetwork, where k is a number of expert, $\mathbf{s}_k^{(i,j)}$ is a vector of the hypernetwork parameters
- $\mathcal{GS}(\boldsymbol{\alpha}, t)$ – gumbel-softmax distribution

6 Conclusion

References

1. Bakhteev, O.Y., Strijov, V.V.: Deep learning model selection of suboptimal complexity. Autom. Remote. Control **79**(8), 1474–1488 (2018)

2. Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In: Bmvc. vol. 1, p. 3 (2016)
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
4. Chen, X., Hsieh, C.J.: Stabilizing differentiable architecture search via perturbation-based regularization. CoRR **abs/2002.05283** (2020)
5. Chen, X., Wang, R., Cheng, M., Tang, X., Hsieh, C.J.: Drnas: Dirichlet neural architecture search. CoRR **abs/2006.10355** (2020)
6. Chu, X., Zhou, T., 0046, B.Z., Li, J.: Fair darts: Eliminating unfair advantages in differentiable architecture search. CoRR **abs/1911.12126** (2019), http://arxiv.org/abs/1911.12126
7. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. CoRR **abs/1609.09106** (2016), http://arxiv.org/abs/1609.09106
8. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017)
9. 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
10. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/ kriz/cifar.html
11. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
12. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. CoRR **abs/1806.09055** (2018), http://arxiv.org/abs/1806.09055
13. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016)
14. Nayman, N., Noy, A., Ridnik, T., Friedman, I., Jin, R., Zelnik-Manor, L.: Xnas: Neural architecture search with expert advice. arXiv preprint arXiv:1906.08031 (2019)
15. Qian, Q., Zhu, S., Tang, J., Jin, R., Sun, B., Li, H.: Robust optimization over multiple domains. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4739–4746 (2019)
16. Ren, P., Xiao, Y., Chang, X., Huang, P.Y., Li, Z., Chen, X., Wang, X.: A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys (CSUR) **54**(4), 1–34 (2021)
17. Wang, Q., Ke, J., Greaves, J., Chu, G., Bender, G., Sbaiz, L., Go, A., Howard, A., Yang, M.H., Gilbert, J., et al.: Multi-path neural networks for on-device multi-domain visual classification. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3019–3028 (2021)
18. Yakovlev, K.: https://github.com/Intelligent-Systems-Phystech/2021-Project85
19. Yao, Q., Xu, J., Tu, W.W., Zhu, Z.: Efficient neural architecture search via proximal iterations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 6664–6671 (2020)
20. Yaushev, F.Y., Isachenko, R.V., Strijov, V.: Concordant models for latent space projections in forecasting. Sistemy i Sredstva Informatiki [Systems and Means of Informatics] **31**(1), 4–16 (2021)