Differential neural ensemble search with diversity control

A Preprint

K. Yakovlev, O.Bakhteev, K. Petrushina, P. Babkin

Abstract

In our research we investigate a new method of sampling the deep learning models using hypernetwork. A hypernetwork is a neural network which controls diversity of the models. It translates a real number into an architecture of a sampled neural network. This method samples deep learning models in one-shot regime, without any additional calculations losses. To evaluate the performance of the proposed algorithm, we conduct experiments on the Fashion-MNIST and CIFAR-10 datasets and compare the resulting ensembles with ones sampled by other searching algorithms.

Keywords differential search · neural ensembles · hypernetwork · diversity control

1 Introduction

Nowadays methods of neural architecture search (NAS) are well-explored and proved to be an effective way of creating more effective and efficient neural networks [2, 7, 4]. Some of these methods use different ways to smooth out the architecture so optimum for it can be found by wide range of methods for smooth optimization problems. On the other hand, neural ensemble search (NES) is modern and not as well investigated problem as NAS, although it is known that ensembles of deep learning models show better results in different applied problems [3]. Our paper investigates an method of sampling deep learning models in a new way that gives compatible results and has its own sphere of implementations.

Our method takes the result of NAS as a base architecture than it samples architectures which are close to the optimal one in terms of Jensen-Shannon divergence (JSd). Architectures differ in terms of λ, idea of method is shown in Fig. 1. Basic architecture is gained with $\lambda = \lambda_1$. Blue ellipses are equidistant surfaces of architectures. Staring with diverse parameter $\lambda = \lambda_2$ resulting architecture performs unacceptable accuracy, so architectures beyond the surface are not included into ensemble. Method can control whether sampled architectures are close enough to the optimal one so they perform good accuracy on the original dataset and are diverse enough so every architecture makes its own contribution to the final answer.

To sample architectures we use hypernetwork [1]. This network generates parameters for another network, which is called target network. Previously hypernetwoks were intended to control different characteristics such as complexity of architecture [6] or parameters of the target model [5] in several modern papers. In our paper it controls diversity of the target models, so every sampled model differs from previously sampled ones in terms of JSd.

The hypernetwork uses JSd to measure difference between two architectures which is symmetric and finite in contrast to more popular Kullback–Leibler divergence. Our main idea of sampling different model is to use a regularizer, based on JSd as a source of diversity.

This way we are able to sample deep learning models in one-shot, without and additional computational losses. To sum up the scheme of our method: find a base architecture using DARTS, sample architectures in one-shot via differentiable algorithm. Inference answer is ensemble of the sampled deep learning models.

We conduct experiments on CIFAR and MNIST datasets to evaluate performance of the proposed method in terms of accuracy and time. Also we compare the performance with state-of-art NES and NAS algorithms [2?].

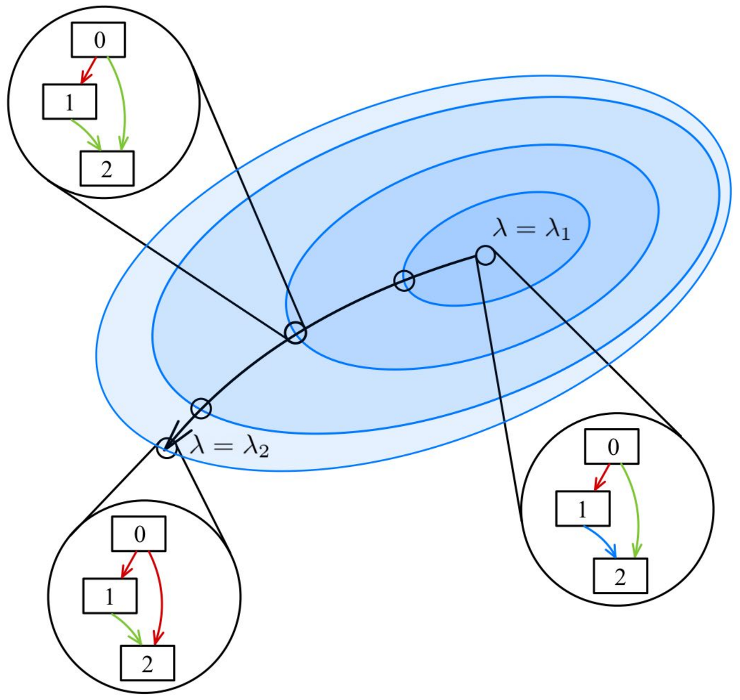

Рис. 1: Architecture space

2 Problem statement

Contrary to the selection of one single architecture in conventional NAS algorithm, this paper focuses on the problem of selecting a well-performing neural network ensemble with diverse architectures from the NAS search space, i.e., neural ensemble search (NES). In our formal statements $\boldsymbol{\alpha}$ denotes an architecture of a model, i.e. a set of operations between nodes. $\boldsymbol{\alpha}^*$ is a NAS resulting architecture, i.e. an optimal architecture. $\boldsymbol{w}^*_{\boldsymbol{\alpha}}$ is vector of optimal parameters for architecture $\boldsymbol{\alpha}^*$. λ is a measure of diversity, a real number from 0 to Λ. $\boldsymbol{\alpha}(\lambda)$ is an architecture that corresponds λ, the architecture is gained after solving problem below. $f(\boldsymbol{w}_{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$ is an output of an architecture $\boldsymbol{\alpha}$ and model parameter $\boldsymbol{w}_{\boldsymbol{\alpha}}$. S – a set of architectures included into ensemble. $\mathcal{L}_{train}$, $\mathcal{L}_{val}$ are training and validation losses, respectively. We imply their dependence on preset dataset and do not denote it explicitly, because we do not deal with it.

Given the ensemble scheme, NES can be formally framed as

$$\min_S \mathcal{L}_{val}\left(\frac{1}{|S|}\sum_{\boldsymbol{\alpha}\in S} f(\boldsymbol{w}^*_{\boldsymbol{\alpha}}, \boldsymbol{\alpha})\right)$$
$$s.t.\ \forall \boldsymbol{\alpha} \in S\ \boldsymbol{w}^*_{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{w}} \mathcal{L}_{train}(f(\boldsymbol{w}^*_{\boldsymbol{\alpha}}, \boldsymbol{\alpha}))$$

We rearrange the problem: general for all architectures. Also architectures differ in terms of λ so resulting functions can be calculated in terms of expected values.

$$\min_{\boldsymbol{\alpha}} \mathbb{E}_\lambda[\mathcal{L}_{val}(\boldsymbol{w}^*, \boldsymbol{\alpha}(\lambda)) - \lambda JS(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}(\lambda))]$$
$$s.t.\ \boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_\lambda[\mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha}(\lambda))]$$

2.1 Computational experiment

In our model λ is a random value, distributed according to uniform distribution from 0 to Λ ($\sim U(0, \Lambda)$). Main goal of basic experiment is to estimate Λ and also to investigate dependence of resulting architecture's performance on λ.

2.1.1 Experiment planning

In our basic experiment we run the algorithm on fashionMNIST dataset for several different λ and looked at resulting architecture and accuracy. Further we will use λ distributed randomly, but in basic experiment it is fixed.

2.1.2 Preliminary report

Obtained results are depicted in the table below. Maximal accuracy of sampled architectures is written down. Also we recorded amount of matched operations between nodes i.e. number of edges between particular nodes that contain the same operation in obtained architecture and the basic one.

This results allow us to choose $\Lambda = 32$. However amount of matching edges does not really coincide with our expectations. Architectures almost do not intersect. It can be explained by the fact that the optimized function is not convex, so algorithm fall into different minimums, however we can get more matching edges by using negative λ.

Таблица 1: preliminary results

λ	accuracy, %	matched operations
0 (optimum)	91.24	-
1/2	91.24	4
2	91.26	3
32	89.21	2
64	87.09	1

3 Method

Below we briefly described our method for solving problem of classification with dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$. For each object $\mathbf{x} \in \mathbf{X}$ there is a label $y \in \mathbf{y}$. We solve the problem via NES, sampling architectures according to novel methodology which is formally described below.

3.1 architecture

Let $\mathcal{V} = \{1, \dots, N\}$ be a set of vertices, where N is a number of vertices, and $\mathcal{E} = \{(i, j) \in V \times V \mid i < j\}$ a set of edges between them. $\mathcal{O}$ is a set of operations to choose from. Usually it contains pooling, convolutions, etc. For each edge there is an operation $o \in \mathcal{O}$ that transits information from one node to another.

NAS algorithms search for optimal architecture. As it was mentioned below, architecture of neural network is a set of operations between nodes. In NAS methods architecture is a vector constructed by following rules. For each edge $(i, j) \in \mathcal{E}$, $\boldsymbol{\alpha}^{(i,j)}$ is a vector, which assigns impact of each operation. $\boldsymbol{\alpha}$ is a concatenation of all structural parameters vectors $\boldsymbol{\alpha}^{(i,j)}$.

3.2 reguralizer

Source of diversity in out method is a regularizer with diversity parameter λ. By subtracting it from a loss function we promote method for finding an architecture which differs from the optimal one.

We use Jensen-Shannon divergence to measure diversity of two architectures. NES algorithms give discrete architecture as a result of their work, i.e. initially optimal architecture $\boldsymbol{\alpha}^*_{\boldsymbol{init}}$ is discrete, so JSd cannot be calculated with $\boldsymbol{\alpha}^*_{\boldsymbol{init}}$ as an argument. We smooth the architecture using smooth parameter τ to solve this problem.

$$\boldsymbol{\alpha}^* = (1 - \tau)\boldsymbol{\alpha}^*_{\boldsymbol{init}} + \tau \frac{1}{|\mathcal{O}|}$$

Assuming τ close to one we obtain a smoothed architecture that contains the same information about architecture as a initial one, but we also can use it in JSd.

3.3 diversity control

In order to control diversity we employ the concept of hypernetwork. A hypernetwork is a parametric mapping from $[0, \Lambda]$ to the set of model structural parameters [6].

$$\boldsymbol{\alpha} : [0, \Lambda], \mathbb{R}^u \to \mathbb{R}^s$$

Where $\mathbb{R}^u$ is a hypernetwork parametric space and $\mathbb{R}^s$ is space of model structural parameters. In this terms $\boldsymbol{\alpha}$ can be redefined using hypernetwork

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}(\lambda, \boldsymbol{a}) \text{ or } \boldsymbol{\alpha}^{(\boldsymbol{i},\boldsymbol{j})} = \boldsymbol{\alpha}^{(\boldsymbol{i},\boldsymbol{j})}(\lambda, \boldsymbol{a}^{(\boldsymbol{i},\boldsymbol{j})})$$

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

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

In our method λ is sampled from predefined distribution $p(\lambda) = U(0, \lambda)$ and new architecture is obtained from hypernetwork.

Список литературы

[1] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. 2016.

[2] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In Proc. ICLR, 2019.

[3] A. R. Narayanan, A. Zela, T. Saikia, T. Brox, and F. Hutter. Multi-headed neural ensemble search. 2021.

[4] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik-Manor. XNAS: Neural architecture search with expert advice. In Proc. NeurIPS, pages 1975–1985, 2019.

[5] J. von Oswald, C. Henning, B. F. Grewe, and J. Sacramento. Continual learning with hypernetworks. 2022.

[6] K. D. Yakovlev, O. S. Grebenkova, O. Y. Bakhteev, and V. V. Strijov. Neural architecture search with structure complexity control. 2022.

[7] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter. Understanding and robustifying differentiable architecture search. In Proc. ICLR, 2020.