Adaptation of the architecture of the deep learning model with performance control*

***S. S. Borodin*[1], *K. D. Yakovlev*[1], *and O. Y. Bakhteev*[1,2]**
{borodin.ss, iakovlev.kd, bakhteev}@phystech.edu
[1] MIPT, Russia
[2] Dorodnicyn Computing Center RAS, Russia

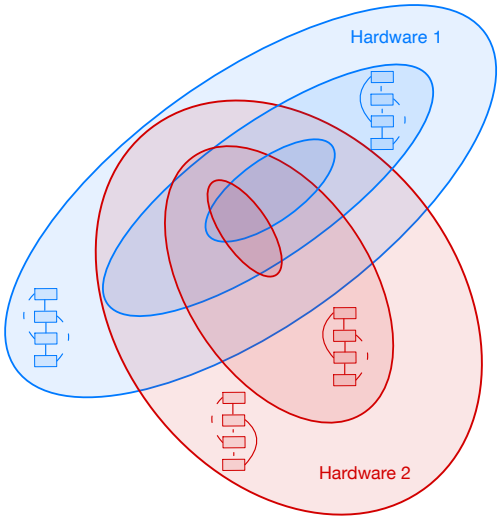
Will be rewritten since problem was corrected.

The paper investigates the problem of deep learning model selection. A method can consider different requirements to optimize performance on a wide range of hardware. The method allows to control trade-off between prediction quality and performance on the intermediate stage of model selection without additional model fitting. The method is based on a modification of a differential architecture search algorithm (DARTS) called DARTS-CC. The method can adapt to benefits and weaknesses of hardware, which is especially important for mobile devices with limited computation budget. To evaluate the performance of the proposed algorithm, we conduct experiments on the Fashion-MNIST and CIFAR-10 datasets using/simulating different hardware (e.g. ...) and compare the resulting architecture with architectures obtained by other neural architecture search methods.

DOI: 10.21469/22233792

1 Introduction

Will be rewritten since problem was corrected.



*The research was supported by the Russian Foundation for Basic Research (grants 00-00-0000 and 00-00-00001).

Discovering architectures for different problems is a difficult task. If you want to find an architecture that considers hardware properties the task becomes inhumanly hard. Optimizing for mobile devices is crucial because of their limited computation budget. With a wide variety of devices it is important to optimize searching not for one, but many different capabilities and requirements.

There are different approaches to finding architectures: evolutionary, RL (reinforced learning) and gradient-based. According to [7] gradient-based show similar prediction quality to other approaches but mainly with lower search cost.

On of the most popular algorithm is DARTS [5]. Main idea of the algorithm is to represent NN as DAG. Where each connection can be of the "candidate operations (e.g., convolution, max pooling, zero)"with different possibilities.

FBNet [3] uses this algorithm to optimize latency. Model shows great results decreasing FLOPS and CPU latency almost without decreasing accuracy.

DARTS-CC [2] uses this algorithm to optimize model complexity with trade-off control. This modification of DARTS can produce different architectures in one fitting run with control to trade-off between quality and complexity.

We propose a method that will combine approaches of DARST-CC [2] and FBNet [3] to obtain multiple architectures optimized for different requirements with different trade-offs between prediction quality and performance in one run.

2 Problem statement

Given a model with structure $\Gamma = (V, E)$, where E is a set of atomic operations such as convolution, pooling, activation etc. The task is to find a subset of edges for which the pruned model gives comparable quality with significant speed-up on a particular device.

Optimization problem:

$$\mathbb{E}_\lambda \mathbb{E}_\gamma \left[\mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}(\lambda)) + \lambda \sum_{e \in E} T(e) \boldsymbol{\gamma}(\lambda)_e \right] \to \min_{\boldsymbol{\gamma}(\lambda)},$$

where $\mathcal{L}_{\text{val}}(\boldsymbol{\gamma})$ is a loss function for a pruned network which is a shorthand of $\mathcal{L}_{\text{valid}}(\widehat{y}_{\boldsymbol{\gamma}}(x), y)$, $\lambda \in \mathbb{R}$ is regularization parameter, $\boldsymbol{\gamma}(\lambda) \in \{0,1\}^{|E|}$, $T(e)$ — time to execute corresponding atomic operation. Since optimization on discrete space is not differentiable we use Gumbel-Softmax approximation [2].

Theorem 1. *Let $\boldsymbol{\Gamma}_{valid} \subset \{0,1\}^D$ is a set for which the computational graph is not broken. Let also loss function be in the following form*

$$\mathcal{L}_{valid}(\boldsymbol{\gamma}) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{\gamma}), y_i)$$

where $f(\boldsymbol{\gamma}) \in \Delta$. Let also

$$\inf_{p \in \Delta} \frac{1}{n} \sum_{i=1}^{n} \ell(p, y_i) > \sup_{\boldsymbol{\gamma} \in \boldsymbol{\Gamma}_{valid}} \mathcal{L}_{valid}(\boldsymbol{\gamma}).$$

Then the following statement is true. Suppose that $\boldsymbol{\gamma}^* \notin \boldsymbol{\Gamma}_{valid}$ is a solution of the following problem

$$\boldsymbol{\gamma}^* = \arg\min_{\boldsymbol{\gamma}} \mathcal{L}_{valid}(\boldsymbol{\gamma}) + \lambda \mathbf{T}^\top \boldsymbol{\gamma}.$$

Then

$$\lambda \geqslant \frac{\inf\limits_{p\in\Delta} \frac{1}{n}\sum_{i=1}^{n} \ell(p, y_i) - \sup\limits_{\boldsymbol{\gamma}\in\boldsymbol{\Gamma}_{valid}} \mathcal{L}_{valid}(\boldsymbol{\gamma})}{\mathbf{T}^\top \mathbf{1}} =: \Lambda > 0.$$

Proof. $\boldsymbol{\gamma}^*$ the solution of the problem. Hence, $\forall \boldsymbol{\gamma}' \in \boldsymbol{\Gamma}_{\text{valid}}$

$$\mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}^*) + \lambda \mathbf{T}^\top \boldsymbol{\gamma}^* \leqslant \mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}') + \lambda \mathbf{T}^\top \boldsymbol{\gamma}' \leqslant \sup_{\gamma\in\boldsymbol{\Gamma}_{\text{valid}}} \mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}) + \lambda \mathbf{T}^\top \mathbf{1}.$$

Since $\forall i \;\; \mathbf{T}_i \geqslant 0$ the following statement is true

$$\mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}^*) + \lambda \mathbf{T}^\top \boldsymbol{\gamma}^* = \inf_{\gamma\in\{0,1\}^D} \frac{1}{n}\sum_{i=1}^{n} \ell(f(\boldsymbol{\gamma}), y_i) + \lambda \mathbf{T}^\top \boldsymbol{\gamma} \geqslant \inf_{p\in\Delta} \frac{1}{n}\sum_{i=1}^{n} \ell(p, y_i).$$

Thus,

$$\sup_{\gamma\in\boldsymbol{\Gamma}_{\text{valid}}} \mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}) + \lambda \mathbf{T}^\top \mathbf{1} \geqslant \inf_{p\in\Delta} \frac{1}{n}\sum_{i=1}^{n} \ell(p, y_i).$$

■

Corollary 1. *For $\lambda < \Lambda$ the solution lies in $\boldsymbol{\Gamma}_{valid}$.*

Algorithm 1 hypernet fitting

```
load pre-trained model
initialize hypernet for the model
while hypernet(λ = 0) has random accuracy do
    Sample γ_sampled ~ N(bias, 0)
    γ ← hypernet(λ)
    Optimize hypernet for loss ‖γ − γ_sampled‖_2^2
end while
while not converged do
    Sample λ ~ U[0, Λ]
    γ ← hypernet(λ)
    Optimize hypernet for loss L_valid(γ) + λ T^⊤ γ
end while
```

3 Computational experiment

The experiment contains of two parts. In the first part we get pretrained model, fix λ and then optimize γ. In the second part we train hypernetwork to find γ for arbitrary λ. We will use ResNet18 [1] as base model and dataset Cifar-10 to train and validate.

Main purpose of the first part is to approximate Λ. Λ is the max value for λ which keeps positive metrics. But before all of it we measure execution time of each module. Then for each $\lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$ we optimize γ for loss with time regularization. Finally, we plot accuracy versus λ. It is expected that plot will have plateau for small λ and then will decrease to worst quality for big λ.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Received January 01, 2017