

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

S. S. Borodin¹, K. D. Yakovlev¹, and O. Y. Bakhteev^{1,2}

{borodin.ss, iakovlev.kd, bakhteev}@phystech.edu

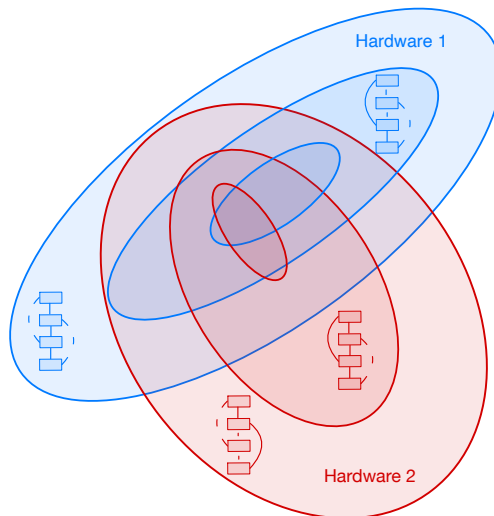
¹ MIPT, Russia

² Dorodnicyn Computing Center RAS, Russia

The paper investigates the problem of structural pruning with respect to target hardware properties. A method considers performance of a target platform to optimize both accuracy and latency on the platform. We use a hypernet to generate a pruned model for a desired trade-off between accuracy and latency. The hypernet is trained end-to-end with backpropagation through the main model. The model adapts 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 CIFAR-10 dataset with ResNet18 as a backbone-model using different hardware (e.g. ...) and compare the resulting architectures with

DOI: 10.21469/22233792

1 Introduction



work in progress...

(motivation) Pruning is good with [citations]... blah-blah-blah.

(iterature review and state-of-the-art) ...

In this work we propose a method of using hypernet [1] to prune main model with respect to performance of target platform. ...

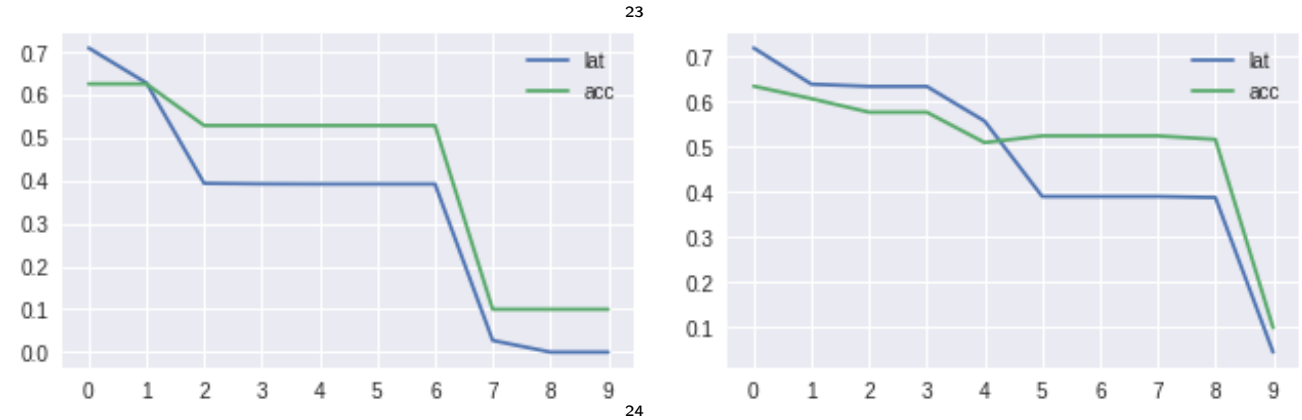
We call a main model the backbone-model, and a model that generates parameters for the backbone-model the hypernet. In this paper we consider a backbone-model as computational

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

graph, where each edge is an atomic operation such as convolution, pooling, activation etc. Parameter of latency regularization referred to as trade-off. Hypernet is a model that for a given trade-off generates subset of edges i.e. pruned model.

Since extremely small latency trade-off provides empty graph with zero latency and random quality, we conduct a base experiment to find such limit for the parameter that it keeps computational graph connected. Once such limit is found we conduct main experiment. In the main experiment the hypernet is trained end-to-end with backpropagation through the main model.

We conduct experiments on ResNet18 [2] with CIFAR-10 dataset, we have achieved 2.5 times acceleration with 10% accuracy decrease, 1.6 times acceleration with 1% accuracy decrease. (Pick one probably new fig. Fig will be converted to pdf. Plot base accuracy. Caption, etc.)



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.

Pruned model is parameterised with $\gamma \in \{0, 1\}^{|E|}$ such that the output of edge e is multiplied with γ_e . This means that we compute the output of edge e iff $\gamma_e = 1$. Since we use hypernetwork to generate γ for regularization parameter $\lambda \in \mathbb{R}$ the generated parameters are denoted as $\gamma(\lambda)$. Thus we formulate optimization problem as:

$$\mathbb{E}_\lambda \mathbb{E}_\gamma \mathcal{L}_{\text{valid}}(\gamma(\lambda)) + \lambda \sum_{e \in E} T(e) \gamma(\lambda)_e \rightarrow \min_{\gamma(\lambda)},$$

where $\mathcal{L}_{\text{valid}}(\gamma) := \mathcal{L}_{\text{valid}}(\hat{y}_\gamma(\mathbf{X}), Y)$ is a loss function for a pruned network \hat{y}_γ on validation dataset (\mathbf{X}, Y) , $T(e)$ — time to execute corresponding atomic operation. Since optimization on discrete space is not differentiable we use Gumbel-Softmax approximation [3].

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

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

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

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

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

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

Then

$$\lambda \geq \frac{\inf_{p \in \Delta} \frac{1}{n} \sum_{i=1}^n \ell(p, y_i) - \sup_{\gamma \in \Gamma_{\text{valid}}} \mathcal{L}_{\text{valid}}(\gamma)}{\mathbf{T}^\top \mathbf{1}} =: \Lambda > 0.$$

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

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

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

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

Thus,

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

32

33 **Corollary 1.** For $\lambda < \Lambda$ the solution lies in Γ_{valid} . ■

Algorithm 1 hypernet fitting

```

1: load pre-trained model
2: initialize hypernet for the model
3: while hypernet( $\lambda = 0$ ) has random accuracy do
4:   Sample  $\gamma_{\text{sampled}} \sim \mathcal{N}(\text{bias}, 0)$ 
5:    $\gamma \leftarrow \text{hypernet}(\lambda)$ 
6:   Optimize hypernet for loss  $\|\gamma - \gamma_{\text{sampled}}\|_2^2$ 
7: end while
8: while not converged do
9:   Sample  $\lambda \sim U[0, \Lambda]$ 
10:   $\gamma \leftarrow \text{hypernet}(\lambda)$ 
11:  Optimize hypernet for loss  $\mathcal{L}_{\text{valid}}(\gamma) + \lambda \mathbf{T}^\top \gamma$ 
12: 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 [2] 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 $\lambda = \lambda_1, \lambda_2, \dots, \lambda_n$ 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] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.
- [2] 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.
- [3] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Received January 01, 2017