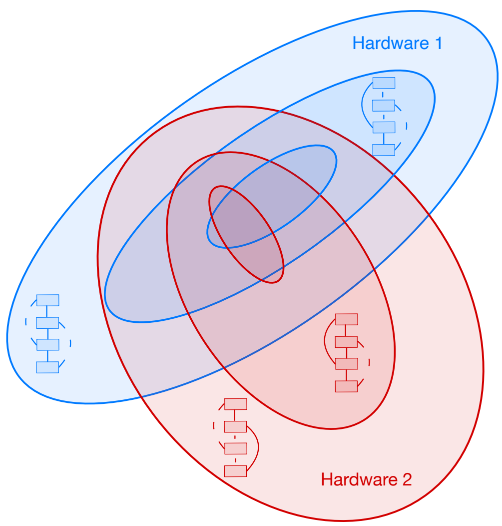
Adaptation of the architecture of the deep learning model with performance control*

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

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 hypernetwork to generate a pruned model for a desired trade-off between accuracy and latency. The hypernetwork 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



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

rewriting...

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.

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

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

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

where $\mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}) := \mathcal{L}_{\text{valid}}(\widehat{y}_{\boldsymbol{\gamma}}(\mathbf{X}), Y)$ is a loss function for a pruned network $\widehat{y}_{\boldsymbol{\gamma}}$ 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 $\boldsymbol{\Gamma}_{valid} \subset \{0,1\}^D$ 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, $\forall \boldsymbol{\gamma}' \in \boldsymbol{\Gamma}_{\text{valid}}$

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

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

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

Thus,

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

∎

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 [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 $\lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$ 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] 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