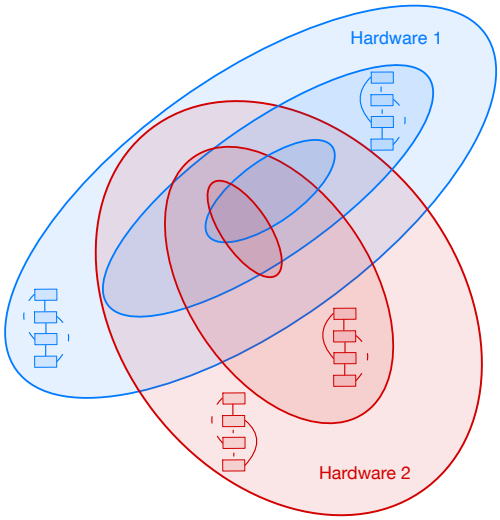
Adaptation of the architecture of the deep learning model with performance control*

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 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. As you can see on 1 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.

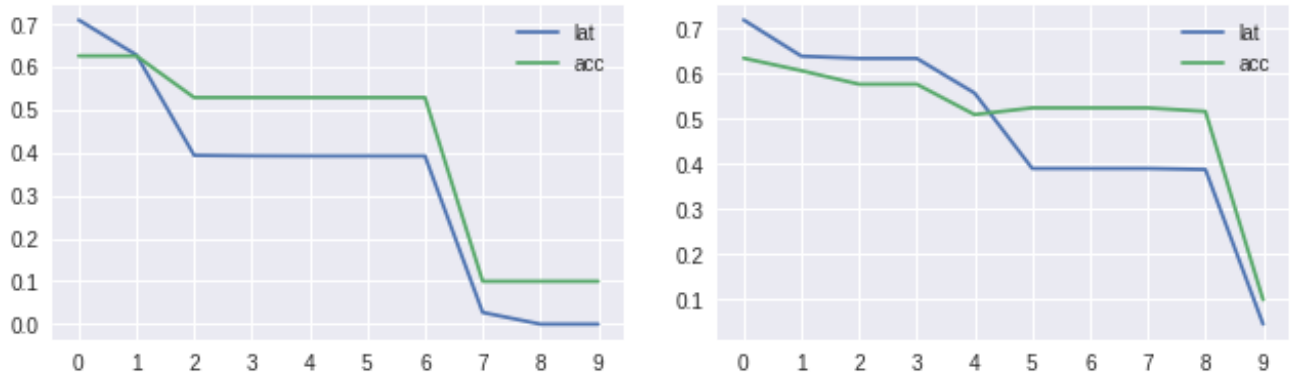


Figure 1 Accuracy and normalized latency plotted versus regularization parameter. The results were obtained for platform ... on CIFAR-10 dataset with pruned ResNet18.

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 $\boldsymbol{\gamma} \in \{0,1\}^{|E|}$ 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 $\mathbf{\Gamma}_{valid} \subset \{0,1\}^D$ 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

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

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

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

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

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

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,

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

■

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

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 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.

3.1 Ablation study/AUC...

work in progress...

Compare the method in terms of random pruning. First, sample may random pruned structures. Second, plot the curve speedup/accuracy and compare AUC of the proposed method (w/o hypernetwork) and AUC of random architectures.

Algorithm 1 hypernet fitting

load pre-trained model
initialize hypernet for the model
while hypernet($\lambda = 0$) has random accuracy **do**
    Sample $\boldsymbol{\gamma}_{\text{sampled}} \sim \mathcal{N}(\text{bias}, 0)$
    $\boldsymbol{\gamma} \leftarrow \text{hypernet}(\lambda)$
    Optimize hypernet for loss $\|\boldsymbol{\gamma} - \boldsymbol{\gamma}_{\text{sampled}}\|_2^2$
end while
while not converged **do**
    Sample $\lambda \sim U[0, \Lambda]$
    $\boldsymbol{\gamma} \leftarrow \text{hypernet}(\lambda)$
    Optimize hypernet for loss $\mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}) + \lambda \mathbf{T}^\top \boldsymbol{\gamma}$
end while

To present value of our method we obtain pruned models by our method and method ...(WIP). Then we measure latency and accuracy of each model and plot accuracy versus latency. To conduct numerical comparison we calculate area under the curve (AUC) for both methods. The results are written in table 1

method	AUC
hypernet (ours)	0.34254908561706543
random	0.06566789001226425

Table 1 Comparison of AUC for different methods

(For now) As you can see on 2 our method provides trade-off between accuracy and latency of pruned networks, while random provides broken computational graph.

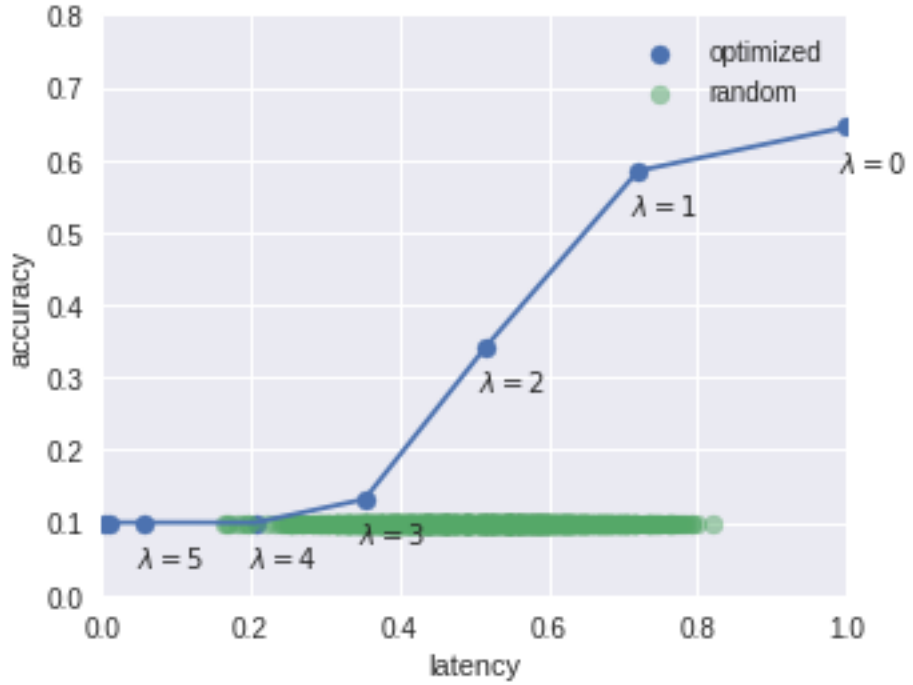


Figure 2 Comparing accuracy-latency dependence of models obtained by our method and random subarchitectures.

3.2 Ablation study/Comparison of discrete and relaxed model

Since optimization of original task on discrete space is not differentiable, we relax optimization. This approach changes optimization problem. In this experiment we ensure that solution for the relaxed problem is an approximation of a solution for the original problem.

Initially we obtain pruned relaxed models for $\lambda_1, \ldots, \lambda_n$. For each pruned model we discretize parameters, thus obtaining discrete models. Finally we measure accuracy for each model and plot the results: accuracy of relaxed models versus accuracy of discrete models for each λ_k. Complete equivalence between tasks should result in identity curve. As we can see on figure 3 actual curve is close to identity curve, but it does not follow identity curvature.

May be plot multiple runs. May be plot latency in the same way. Convert to pdf.

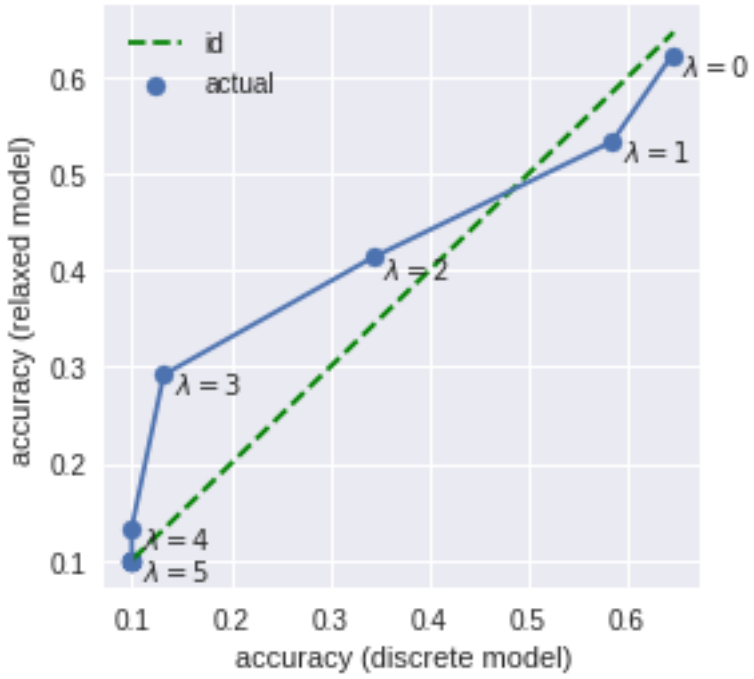


Figure 3 Comparing accuracy of relaxed and discrete models for different λ.

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