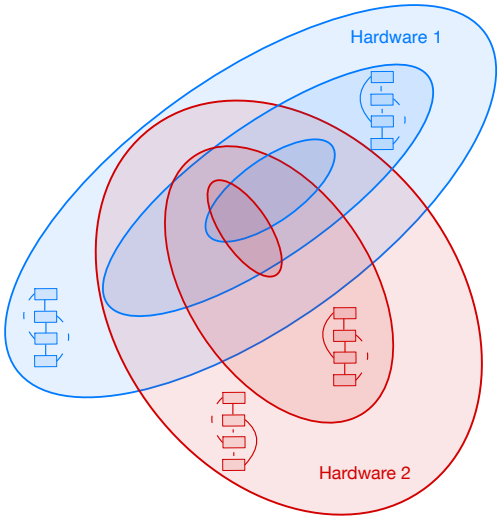
Adaptation of the architecture of the deep learning model with performance control*

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...

Since introduction of residual networks [3] models were able to grow in size without vanishing/exploding gradients. As bigger models accomplish better accuracy, they become over-parameterized [7].

One of the techniques to reduce over-parameterization is pruning [2, 5] that can reduce the number of parameters by factors. But we can target other characteristics to reduce. In this

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

paper we target empirically measured latency. There are other time related properties, but such latency better describes real world requirements [6].

(literature 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 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.

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:

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

where $\mathcal{L}_{\text{valid}}(\boldsymbol{\gamma}) := \mathcal{L}\big(\widehat{y}_{\boldsymbol{\gamma}}(\mathbf{X}\big), Y)$ is a loss function for a pruned network $\widehat{y}_{\boldsymbol{\gamma}}$ on validation dataset $(\mathbf{X}, Y)$, $T(e)$ is the time to execute corresponding atomic operation. Since optimization on discrete space is not differentiable we use Gumbel-Softmax approximation [4].

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

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

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 $\boldsymbol{\Gamma}_{valid}$.*

3 Computational experiment

(plan) Brief description of basic and main experiments. Describe backbone-model with Gumbel-Softmax relaxation.

(old) 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 [3] as base model and dataset Cifar-10 to train and validate.

3.1 Basic experiment

Main purpose of the basic experiment is to approximate Λ. Λ is the max value for λ which keeps better than random performance. It is important to obtain Λ, because we want to train hypernetwork for all λ which does not lead to broken computantional graph.

Algorithm 1 plots figure 1 which gives rough approximation for Λ. But for appropriate $(\lambda_1, \ldots, \lambda_n)$ estimation is acceptable, since hypernetwork fitting algorithm allows such imprecision.

We expect the accuracy versus latency plot to contain three parts: plateau for small λ since regularization does not impact enough, slope to random performance, random performance for $\lambda > \Lambda$.

As you can see on figure 1 our hypothesis has been confirmed. We can see plateau for $\lambda \leqslant 2$, slope for $2 \leqslant \lambda \leqslant 5$ and random performance for $\lambda \geqslant 5$. So it is safe to assume $\Lambda > 5$ for the main experiment.

3.2 Main experiment

In the main experiment we want to train hypernetwork to predict $\boldsymbol{\gamma}$ for each $\lambda \in [0, \Lambda]$. Since discrete search space of $\boldsymbol{\gamma}$ is finite, it is possible that obtained hypernetwork would result in piecewise-constant function.

It is useful for convergence to initialize hypernetwork to predict such $\boldsymbol{\gamma}$ that keeps computational graph not broken. So beforehand we find such vector of logits $\boldsymbol{\gamma}_{\text{init}}$ that provides better than random performce on relaxed backbone-model. To find such vector we generate $\boldsymbol{\gamma}_{\text{init}} \sim \mathcal{N}(\text{shift}, 1)$ for increasing shift till the condition is met.

Once $\boldsymbol{\gamma}_{\text{init}}$ is found we optimize hypernetwork end-to-end for arbitrary λ as shown in 2. More specific we assume that $\lambda \sim \text{U}[0, \Lambda]$. We used piecewise-constant function with 10 intervals with

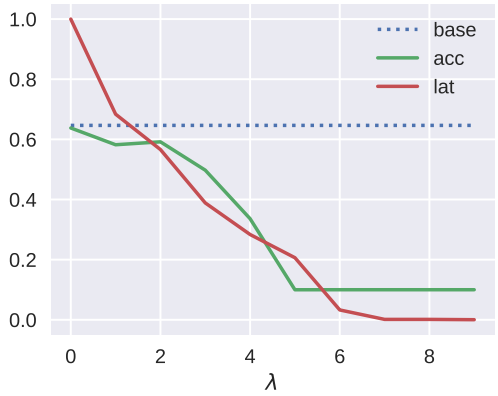


Figure 1 Accuracy and normalized latency plotted versus regularization parameter. The results were obtained on CIFAR-10 dataset with pruned ResNet18.

Algorithm 1 basic algorithm

```
(X, Y) ← train dataset
load backbone-model
for λ = λ_1, ..., λ_n do
    while not converged do
        ŷ_γ ← backbone-model.prune(γ)
        loss ← L(ŷ_γ(X), Y) + λT^⊤γ
        Optimize γ for loss
    end while
end for
plot figure 1
```

Algorithm 2 hypernetwork train

```
(X, Y) ← train dataset
load backbone-model
initialize hypernetwork with γ_init        ▷ make hypernetwork predict γ_init for all λ
while not converged do
    Sample λ ~ U[0, Λ]
    γ ← hypernetwork(λ)
    ŷ_γ ← backbone-model.prune(γ)
    loss ← L(ŷ_γ(X), Y) + λT^⊤γ
    Optimize hypernetwork for loss
end while
```

length = 1. On figure 2 we can see some of intervals have same accuracy and latency. Numerical values for merged intervals are shown in table 1.

3.3 Ablation study/AUC comparison

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.

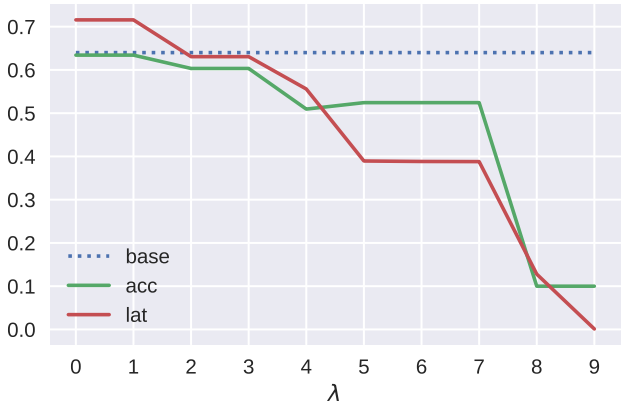


Figure 2 Accuracy and normalized latency plotted versus regularization parameter. The results were obtained on CIFAR-10 dataset with pruned ResNet18. This plot shows performance of pruned with hypernetwork backbone-model.

Table 1 Comparison of accuracy and acceleration of pruned with hypernetwork backbone-model for different λ. Here "accuracy diff"-column shows difference between non-pruned model and pruned model accuracy, "acceleration"-column is calculated as 1/latency.

λ	accuracy (%)	accuracy diff. (%)	acceleration
base	64	0	1
$[0; 2]$	63.42	0.58	1.397
$[2; 4]$	60.33	3.66	1.585
$[4; 5]$	50.94	13.06	1.798
$[5; 8]$	52.44	11.56	2.567

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 2

Table 2 Comparison of AUC for different methods

method	AUC
hypernet (ours)	0.342
random	0.065

(For now) As you can see on 3 our method provides trade-off between accuracy and latency of pruned networks, while random provides broken computational graph.

3.4 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.

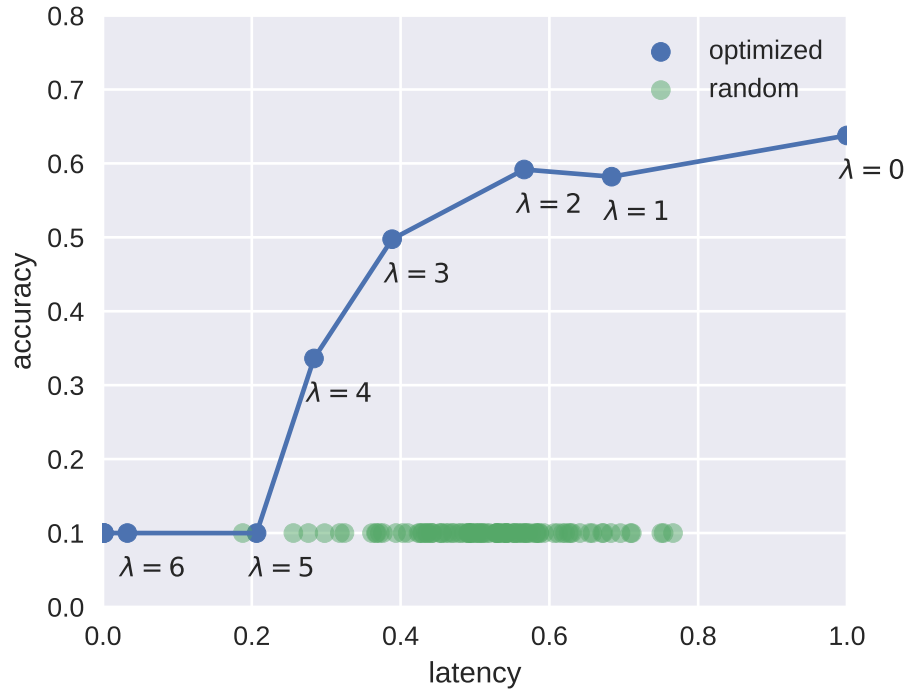


Figure 3 Comparing accuracy-latency dependence of models obtained by our method and random subarchitectures.

Initially we obtain pruned relaxed models for $\lambda_1, \ldots, \lambda_n$. For each pruned model we discretize γ, 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 4 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.

References

[1] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.

[2] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[3] 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.

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

[5] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[6] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2820–2828, 2019.

[7] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

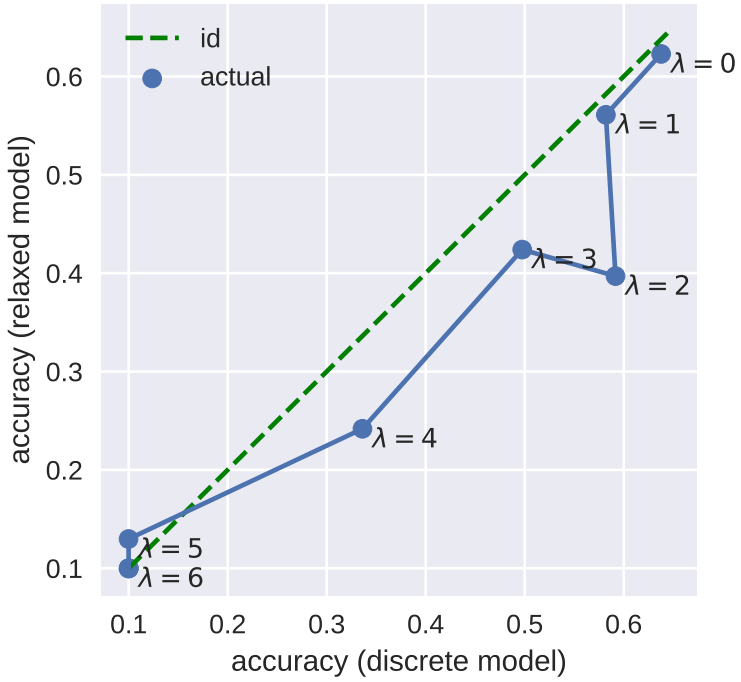


Figure 4 Comparing accuracy of relaxed and discrete models for different λ.

102 *Received January 01, 2017*