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 and compare the resulting architectures with architectures obtained by greedy algorithm.

Keywords: *hardware-aware latency pruning; hypernetwork*

DOI: 10.21469/22233792

1 Introduction

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

We call a main model the backbone-model, and a model that generates parameters for the backbone-model the hypernetwork. 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. Hypernetwork is a model that for a given trade-off generates subset of edges i.e. pruned model.

In this work we propose a method of using hypernetwork [1] to prune deep learning model with respect to performance of target platform. Since space of pruned models is discrete we use Gumbel-Softmax [4] trick to relax space and make optimization problem differentiable. We measure latency of model on target platform to prune model with respect to hardware properties. Example of our loss for different platforms is shown in Figure 1.

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 hypernetwork is trained end-to-end with backpropagation through the main model. Finally we compare the results of models pruned with hypernetwork and greedy

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

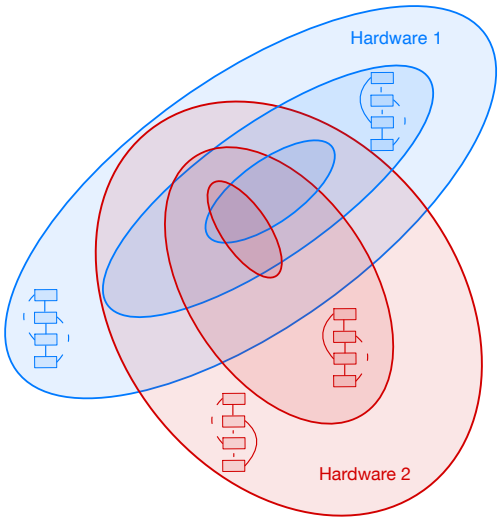

Figure 1 Example of possible loss graph for different hardware. The blue graph representing loss for Hardware 1 is different from the red graph representing loss for Hardware 2 because their latency diverges from one to another and our method proposes usage of loss function with latency regularization. Thus, optimal pruning diverges for different platforms.

algorithm to show significantly better trade-off between accuracy and latency. Lastly, we ensure that original and relaxed problems equivavelent.

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 $\widetilde{\gamma} \in \{0,1\}^{|E|}$ such that the output of edge e is multiplied with $\widetilde{\gamma}_e$. This means that we compute the output of edge e iff $\widetilde{\gamma}_e = 1$. Thus we formulate optimization problem for specific λ with discrete $\widetilde{\gamma}$ as:

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

where $\mathcal{L}_{\text{valid}}(\gamma) := \mathcal{L}\big(\widehat{y}_\gamma(\mathbf{X}), Y\big)$ is a loss function for a pruned network $\widehat{y}_\gamma$ on validation dataset $(\mathbf{X}, Y)$, $T(e)$ is the time to execute corresponding atomic operation.

We use hypernetwork to generate $\boldsymbol{\gamma}$ for regularization parameter $\lambda \in \mathbb{R}$, so the generated parameter is denoted as $\boldsymbol{\gamma}(\lambda)$. We want to optimize for arbitrary λ, thus we assume $\lambda \sim P$ for some distribution P. Since optimization on discrete space $\{0,1\}^{|E|}$ is not differentiable we use Gumbel-softmax trick [4] to relax space. Then the search space is replaced with relaxed space $[0;1]^{|E|}$. Thus we formulate optimization problem for relaxed $\boldsymbol{\gamma}(\lambda)$ as:

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

We will call a computational subgraph not broken if there exists path in the subgraph that connects input and output vertices. Hence, sufficient condition for graph to be not broken is non-constant output.

Since extremely large λ provides empty graph with zero latency and random quality, we want to find a set $\mathscr{L} \subset \mathbb{R}$ such that for $\lambda \in \mathscr{L}$ computational graph is not broken. Our hypothesis is that exists such Λ that for $\lambda \in [0; \Lambda]$ the graph is not broken.

Next theorem shows that such Λ exists.

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

$$\mathcal{L}_{valid}(\boldsymbol{\gamma}) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(\widehat{y}_{\boldsymbol{\gamma}}(\mathsf{x}_i), y_i\big)$$

where (x, y_i) is a validation dataset $(\mathbf{X}, Y)$ entry, ℓ is a loss function for a single entry. Let also

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

where $\Delta := \bigcup_{\gamma\in\mathbf{\Gamma}} \widehat{y}_{\boldsymbol{\gamma}}(\mathbf{X})$.

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

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

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

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

Thus,

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

■

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

The optimization algorithm for relaxed $\boldsymbol{\gamma}$ for specific λ is shown in Algorithm 1. The optimization algorithm for hypernetwork is shown in Algorithm 2.

3 Computational experiment

The purpose of the computational experiment is to analyze the proposed method efficiency. The experiments were conducted on CIFAR-10 dataset. We use ResNet18 [3] as a backbone-model. Initially we train the model for 25 epochs, measure time of each atomic operation and use mean latency, then latency is normalized to sum up to 1. Before experiment we

Algorithm 1 basic algorithm

$(\mathbf{X}, Y) \leftarrow$ train dataset
load backbone-model
while not converged **do**
    $\widehat{y}_{\boldsymbol{\gamma}} \leftarrow$ backbone-model.prune($\boldsymbol{\gamma}$)
    loss $\leftarrow \mathcal{L}\big(\widehat{y}_{\gamma}(\mathbf{X}), Y\big) + \lambda \mathbf{T}^{\top}\boldsymbol{\gamma}$
    Optimize $\boldsymbol{\gamma}$ for loss
end while

Algorithm 2 hypernetwork train

$(\mathbf{X}, Y) \leftarrow$ train dataset
load backbone-model
initialize hypernetwork with $\boldsymbol{\gamma}_{\text{init}}$ ▷ make hypernetwork predict $\boldsymbol{\gamma}_{\text{init}}$ for all λ
while not converged **do**
    Sample $\lambda \sim \mathrm{U}[0, \Lambda]$
    $\boldsymbol{\gamma} \leftarrow$ hypernetwork(λ)
    $\widehat{y}_{\boldsymbol{\gamma}} \leftarrow$ backbone-model.prune($\boldsymbol{\gamma}$)
    loss $\leftarrow \mathcal{L}\big(\widehat{y}_{\gamma}(\mathbf{X}), Y\big) + \lambda \mathbf{T}^{\top}\boldsymbol{\gamma}$
    Optimize hypernetwork for loss
end while

wrap backbone-model into interpreter that allows us to use Gumbel-Softamx trick for pruning reparameterization. Specifically we use RelaxedBernoulli since there are only two categories: 0 and 1.

In the basic experiment we optimize $\boldsymbol{\gamma}$ for different λ to approximate Λ. Then in the main experiment we train hypernetwork using Λ. We conducted two ablation study experiments. First experiment compares results of hypernetwork and greedy algorithm. In the second one we compare results of relaxed and discrete pruned models to ensure equivalence of original and relaxed problems.

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 2 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 2 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.

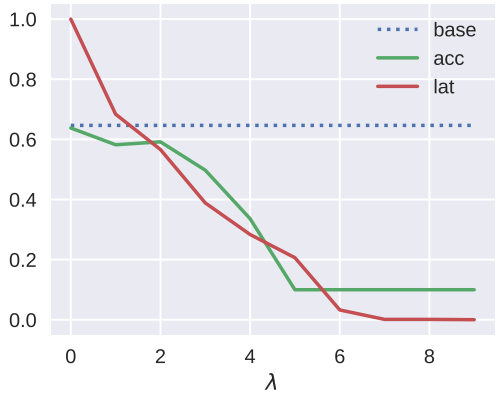

Figure 2 Accuracy and normalized latency plotted versus regularization parameter. The results were obtained on CIFAR-10 dataset with pruned ResNet18.

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 performance 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. We start the search with shift $= 4$, increasing by 0.5, and find appropriate value shift $= 4.5$.

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 use piecewise-constant function with 10 intervals with length $= 1$. So $\Lambda = 10$. On figure 3 we can see some of intervals have same accuracy and latency. Numerical values for merged intervals are shown in table 1. We have achieved significant acceleration with reasonable accuracy decrease compared to greedy algorithm.

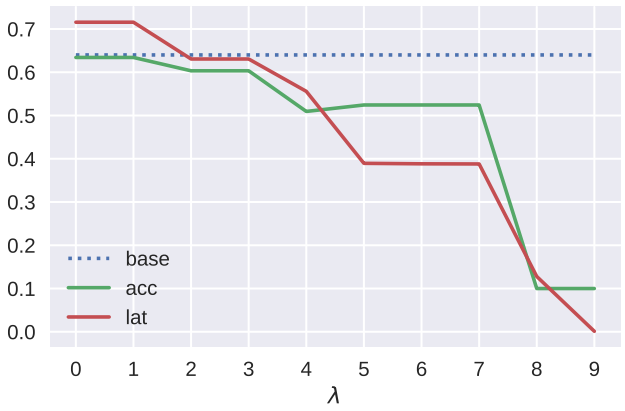

Figure 3 Accuracy and normalized latency plotted versus regularization parameter. The results were obtained on CIFAR-10 dataset with pruned ResNet18. This plot shows performance of pruned with hypernetwork backbone-model.

3.3 Ablation study/AUC comparison

To present value of our method we obtain pruned models by our method and greedy algorithm. Then we measure latency and accuracy of each model and plot accuracy versus latency. Plotted values are shown in 4 To conduct numerical comparison we calculate area under the curve (AUC) for both methods. The results are written in table 2.

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

Table 2 Comparison of AUC for different methods

method	AUC
hypernetwork (ours)	0.342
random	0.065

(For now) As you can see on 4 our method provides trade-off between accuracy and latency of pruned networks, while random provides broken computational graph.

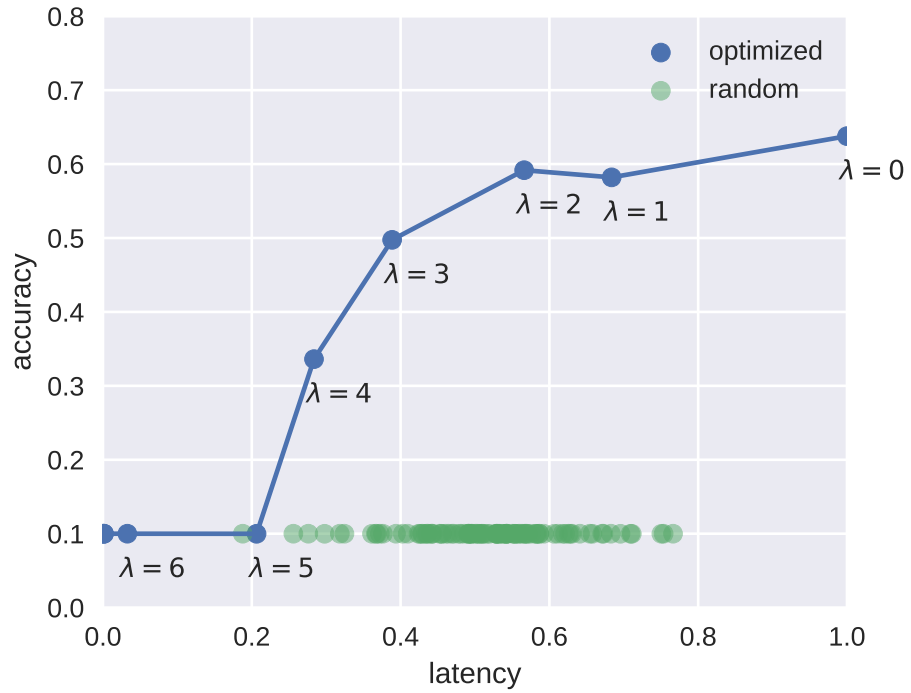

Figure 4 Comparing accuracy-latency dependence of models obtained by our method and random subarchitectures.

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.

Initially we obtain pruned relaxed models for $\lambda_1, \ldots, \lambda_n$. For each pruned model we discretize $\boldsymbol{\gamma}$, 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 5 actual curve is close to identity curve, but it does not follow identity curvature.

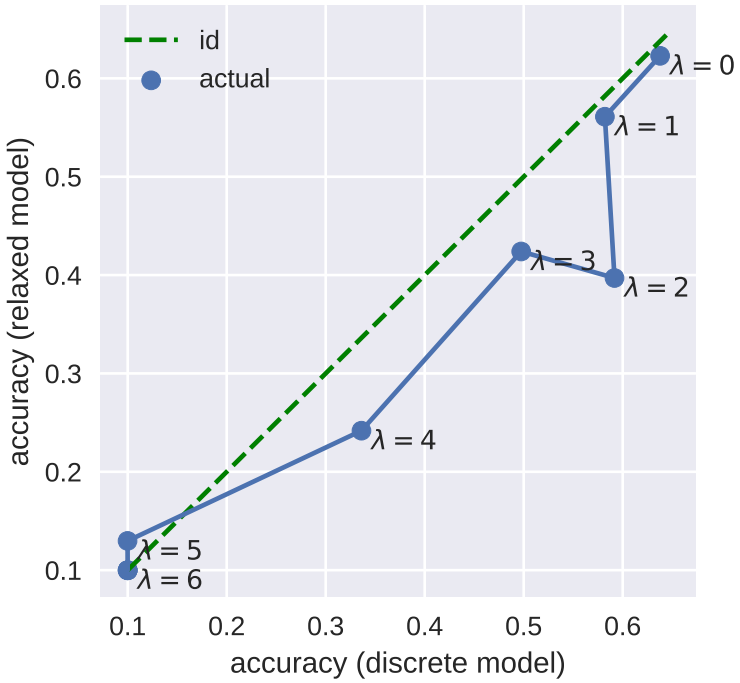

Figure 5 Comparing accuracy of relaxed and discrete models for different λ.

4 Concluding Remarks

In this paper we have investigated the problem of structural pruning with respect to target hardware properties. To make search differentiable we relaxed the problem with gumbel-softmax trick. To solve relaxed problem we have trained piecewise-constant hypernetwork end-to-end for arbitrary trade-off parameter with backpropagation through the backbone-model. We compared accuracy of relax-pruned and discrete-pruned models, to ensure equivalence of relaxed and original problems. Thus achieving better accuracy for the same latency and better latency for the same accuracy than greedy algorithm.

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.

Received January 01, 2017