FNNP: Fast Neural Network Pruning Using Adaptive Batch Normalization

Anonymous authors
Paper under double-blind review

Abstract

Finding out the computational redundant part of a trained Deep Neural Network (DNN) is the key question that pruning algorithms target on. Many algorithms try to predict the model performance of the pruned sub-nets by introducing various evaluation methods. But they are either inaccurate or very complicated for general application. In this work, we present a pruning method called Fast Neural Network Pruning (FNNP), in which a simple yet efficient evaluation component called adaptive-BN-based evaluation is applied to unveil a strong correlation between different pruned DNN structures and their final settled accuracy. This strong correlation allows us to fast spot the pruned candidates with the highest potential accuracy without actually fine-tuning them. FNNP does not require any extra regularization or supervision introduced to a common DNN training pipeline but still can achieve better accuracy than many carefully-designed pruning methods. In the experiments of pruning MobileNet V1 and ResNet-50, FNNP outperforms all compared methods by up to 3.8%. Even in the more challenging experiments of pruning the compact model of MobileNet V1, our FNNP achieves the highest accuracy of 70.9% with an overall 50% operations (FLOPs) pruned. All accuracy data are Top-1 ImageNet classification accuracy. Source code and models are accessible to open-source community[1].

1 Introduction

Deep Neural Network (DNN) pruning aims to reduce computational redundancy from a full model with an allowed accuracy range. Pruned models usually result in a smaller energy or hardware resource budget and, therefore, are especially meaningful to the deployment to power-efficient front-end systems. However, how to trim off the parts of a network that make little contribution to the model accuracy is no trivial question.

DNN pruning can be considered as a searching problem. The searching space consists of all legitimate pruned networks, which in this work are referred to as sub-nets or pruning candidates. In such space, how to obtain the sub-net with the highest accuracy with reasonably small searching efforts is the core of the pruning task.

Particularly, an evaluation process can be commonly found in existing pruning algorithm pipelines. Such a process preempts the potential of sub-nets so that the best pruning candidate can be selected to deliver the final pruning strategy. For example, Filter Pruning (Li et al., 2016) makes the decision of pruning strategy based on accuracies of sub-nets without fine-tuning, which is called sensitivity analysis. The advantage of using an evaluation module is saving time because training all sub-nets to convergence for comparison can be very time-consuming and hence impractical.

However, we found that the evaluation methods in existing works are sub-optimal. Concretely, they are either inaccurate or complicated.

By saying "inaccurate", it means the winning sub-nets from the evaluation process do not necessarily deliver high accuracy when they converge (Li et al., 2016; He et al., 2018c). This will be quantitatively proved in Section 4 with a proposed correlation index. To our knowledge, we are the first to introduce correlation-based analysis for sub-net selection in pruning tasks. Moreover, we

[1]https://github.com/anonymous47823493/FNNP

demonstrate that the reason such evaluation is inaccurate is the use of sub-optimal statistical values for Batch Normalization (BN) layers. In this work, we use a so-called "adaptive BN" trick to fix the issue and effectively reach a higher correlation for our proposed evaluation process.

By saying "complicated", it points to the fact that the evaluation process in some works relies on tricky or heavy components such as a reinforcement learning agent (He et al., 2018c), auxiliary network training(Liu et al., 2019b), knowledge distillation (Luo et al., 2017), generative adversarial learning (Lin et al., 2019) and so on. These methods require careful hyper-parameter tuning or separately training auxiliary models. These requirements make it potentially difficult to repeat the results and these pruning methods can be time-consuming due to their high algorithmic complexity.

The above-mentioned issues in current works motivate us to propose a better pruning algorithm that equips with a faster and more accurate sub-net evaluation mechanism, which eventually provides state-of-the-art pruning performance. The main novelty in this work is described below:

- An automated pruning method is proposed called Fast Neural Network Pruning (FNNP). In this algorithm, we are the first to introduce a correlation-based quantitative analysis to measure the effectiveness of the evaluation process in pruning tasks.
- We propose an evaluation method based on adaptive batch normalization. This method can provide a fast and accurate estimation of the converged accuracy of a pruned model.
- Our proposed FNNP outperforms many complex pruning methods and achieves state-of-the-art performance. In the ResNet-50 experiments, our FNNP outperforms all compared methods by 1.3 % to 3.8 %. Even in the challenging task of pruning a compact model of MobileNet V1, our FNNP achieves the highest accuracy of 70.9% with an overall 50 % operations (FLOPs) pruned. All accuracy data are Top-1 ImageNet classification accuracy.

2 Related work

As mentioned in the previous section, existing DNN pruning approaches can be considered as efforts from two perspectives: modification of the weight distribution and proposal of different searching methods.

Modification of the weight distribution mainly refers to regularization to the original model parameter distribution for pruning purposes. For example, introduced regularization includes LASSO (Wen et al., 2016) and so on. Our proposed method is orthogonal with this type of technique.

Meanwhile, different searching methods were proposed to spot good pruning candidates. Pruning was mainly handled by hand-crafted heuristics in early time (Li et al., 2016). So a pruned candidate network is obtained by human expertise and evaluated by training it to the converged accuracy, which can be very time consuming considering the large number of plausible sub-nets. Then more automated approaches such as greedy strategy were introduced to save manual efforts (Yang et al., 2018b). More recently, versatile techniques were proposed to achieve automated and efficient pruning strategies such as reinforcement learning (He et al., 2018c), knowledge distillation (He et al., 2018c), generative adversarial learning mechanism (Lin et al., 2019) and so on.

However, two issues that remain in these works, as briefly discussed in Section 1, are an inaccurate evaluation for sub-nets and/or complicated evaluation that slows down the sub-net selection.

Our work is also related to AutoML, especially neural architecture search (NAS). Recently, there has been a growing interest in developing algorithmic solutions to automate the manual process of architecture design. Notable works are (Zoph & Le, 2017; Real et al., 2018; Liu et al., 2019a; Cai et al., 2019a;b; Guo et al., 2019; Chu et al., 2019b;a). But all these methods are not combined with pre-trained weight, i.e., they separate the network architecture and network parameter. This contradicts our knowledge that some tasks have to be pre-trained by ImageNet.

3 Methodology

A typical neural network training and pruning pipeline is generalized and visualized in Figure 1. Pruning is normally applied to a trained full-size network for redundancy removal purposes. An evaluation process is then followed up to select the pruned sub-net with the best accuracy potential.

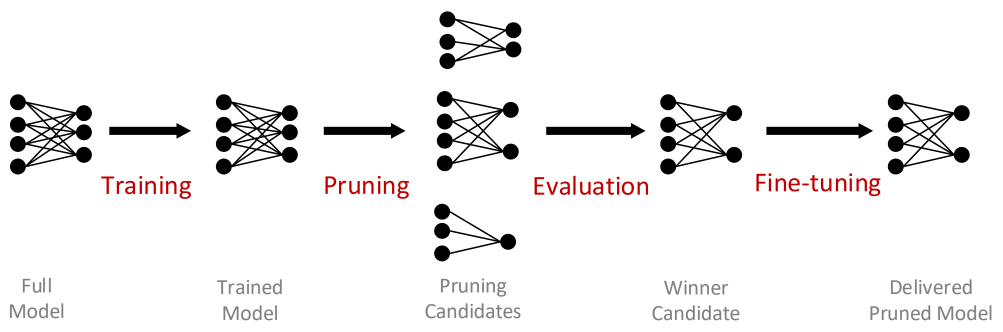

Figure 1: A typical pipeline for neural network training and pruning

Such a winner sub-net is optionally fine-tuned to gain accuracy back from losing parameters in the trimmed filters. In this work, we focus on structured filter pruning approaches, which can be generally formulated as

$$\begin{aligned}(r_1, r_2, ..., r_L)^* &= \underset{r_1, r_2, ..., r_L}{\arg\min} \; \mathcal{L}(\mathcal{A}(r_1, r_2, ..., r_L; w)) \\ &s.t. \quad \mathcal{C} < constraints\end{aligned} \tag{1}$$

where $\mathcal{L}$ is the loss function and $\mathcal{A}$ is the neural network model. r_l is the pruning ratio applied to the l^{th} layer. During the evaluation process, given some constraints $\mathcal{C}$ such as targeted amount of parameters, operations or execution latency, a group of pruning candidates can be evaluated to pick a pruning ratio combination $(r_1, r_2, ..., r_L)^*$, also referred as a pruning strategy, that allows the highest possible inference accuracy once applied to the full-size network. All pruning candidates form a searching space and the evaluation module assesses the returned sub-nets from some searching method, and deliver the winner candidate to the optional final fine-tuning step.

3.1 Motivation

In many published approaches (He et al., 2018c; Li et al., 2016) in this domain, pruning candidates compare with each other in terms of evaluation accuracy. The sub-nets with higher evaluation accuracy are selected and expected to also deliver high accuracy after fine-tuning. However, we realize this is not necessarily true, especially considering a particular example that a pruned network, which delivers only 0.5% top-1 accuracy, can provide 65% fine-tuned accuracy. We wonder how strong the low-range evaluation accuracy is positively related to the final converged accuracy. Why does the evaluation process in these methods suffer from such massive performance degradation? The above questions triggered our investigation into the commonly used evaluation process, which is called vanilla evaluation in this work.

Interestingly, we found that it is the batch normalization layer that largely affects the evaluation. Without fine-tuning, pruning candidates have parameters that are a subset of those in the full-size model. So the layer-wise feature map data are also affected by the changed model dimensions. However, vanilla evaluation still uses Batch Normalization (BN) inherited from the full-size model. The out-dated statistical values of BN layers eventually drag down the evaluation accuracy to a surprisingly low range and, more importantly, break the correlation between evaluation accuracy and the final converged accuracy of the pruning candidates in the strategy searching space. Fine-tuning all pruning candidates and then compare them is an accurate way to carry out the evaluation, however, it is very time-consuming to do the training-based evaluation for even single-epoch fine-tuning due to the large scale of the searching space. We give quantitative analysis later in this section to demonstrate this point.

Firstly, to quantitatively demonstrate the idea of vanilla evaluation and the problems that come with it, we symbolize the original BN (Ioffe & Szegedy, 2015) as below:

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{2}$$

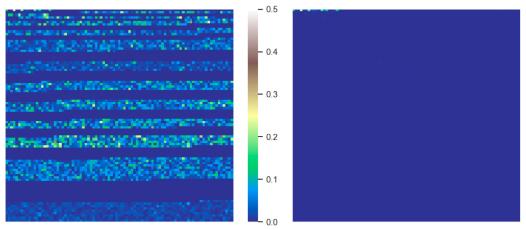

Figure 2: Visualization of distances of BN statistics in terms of mean values. Each pixel refers to the distance of one BN moving mean of a channel in MobileNetV1. **Left:** $\|\mu_T - \mu_{val}\|_2$, distance between global statistics and the true values. **Right:** distance between the adaptive-BN statistics and the true values $\|\hat{\mu_T} - \mu_{val}\|_2$

Where β and γ are trainable scale and bias terms. ϵ is a term with small value to avoid zero division. For a mini-batch with size N, the statistical values of μ and σ^2 are calculated as below:

$$\begin{aligned} \mu_{\mathcal{B}} &= E[x_{\mathcal{B}}] = \frac{1}{N}\sum_{i=1}^{N} x_i, \\ \sigma_{\mathcal{B}}^2 &= Var[x_{\mathcal{B}}] = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_{\mathcal{B}})^2. \end{aligned} \tag{3}$$

During training, μ and σ^2 are calculated with the moving mean and variance:

$$\begin{aligned} \mu_t &= m\mu_{t-1} + (1-m)\mu_{\mathcal{B}}, \\ \sigma_t^2 &= m\sigma_{t-1}^2 + (1-m)\sigma_{\mathcal{B}}^2, \end{aligned} \tag{4}$$

where m is the momentum coefficient and subscript t refers to the number of training iterations. In a typical training pipeline, if the total number of training iteration is T, μ_T and σ_T^2 are used in testing phase. These two items are called global BN statistics, where ”global” refers to the full-size model.

As briefly mentioned before, the vanilla evaluation used in (He et al., 2018c; Li et al., 2016) apply global BN statistics to the pruned networks to fast evaluate their accuracy potential, which we think leads to the low-range accuracy and unfair candidate selection. If the global BN statistics are out-dated to the sub-nets, we should re-calculate μ_T and σ_T^2 with adaptive values by conducting a few iterations of inference on part of the training set, which essentially adapts the BN statistical values to the pruned network connections. Concretely, we freeze all the network parameters while resetting the moving average statistics. Then, we update the moving statistics by a few iterations of forward-propagation, using Equation 4, but without backward propagation. We note the adaptive BN statistics as $\hat{\mu_T}$ and $\hat{\sigma_T^2}$.

We consider BN statistics calculated on validation data as the true statistics, noted as μ_{val} and σ_{val}^2. We expect $\hat{\mu_T}$ and $\hat{\sigma_T^2}$ to be as close as possible to the true BN statistics values so they could deliver close computational results. So we attempt to visualize the distance of BN statistical values gained from different evaluation methods. Here, we show the mean values as an example in Figure 2. The mean value distance between global statistics and the true values can be formulated as $\|\mu_T - \mu_{val}\|_2$, which is visualized in Figure 2 left. The distance between the mean values sampled from adaptive BN and the true values can be formulated as $\|\hat{\mu_T} - \mu_{val}\|_2$, which is visualized in Figure 2 right. The visual observation shows that adaptive BN provides closer statistical values to the true values while global BN is way further. A possible explanation is that the global BN statistics are out-dated and not adapted to the pruned network connections. So they mess up the inference accuracy during evaluation for the pruned networks.

3.2 Correlation Measurement

As mentioned before, a “good” evaluation process in the pruning pipeline should present a strong positive correlation between the evaluated pruning candidates and their corresponding converged accuracy. Here, we compare two different evaluation methods, adaptive-BN-based and vanilla evaluation, and study their correlation with the fine-tuning-based evaluation. So we symbolize a vector of accuracy for all pruning candidates in the searching space (Figure 3) separately using the above three evaluation methods as $X1$, $X2$, Y correspondingly.

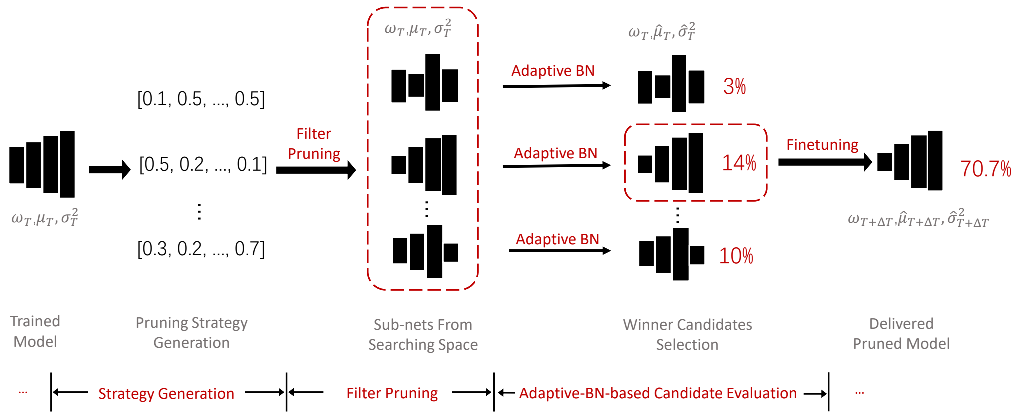
$\omega_T, \mu_T, \sigma_T^2$
[0.1, 0.5, ..., 0.5]
[0.5, 0.2, ..., 0.1]
[0.3, 0.2, ..., 0.7]
Filter Pruning
Adaptive BN
Adaptive BN
Adaptive BN
$\omega_T, \hat{\mu}_T, \hat{\sigma}_T^2$
3%
14%
10%
Finetuning
70.7%
$\omega_{T+\Delta T}, \hat{\mu}_{T+\Delta T}, \hat{\sigma}_{T+\Delta T}^2$
Trained Model
Pruning Strategy Generation
Sub-nets From Searching Space
Winner Candidates Selection
Delivered Pruned Model
Strategy Generation
Filter Pruning
Adaptive-BN-based Candidate Evaluation

Figure 3: Fast Neural Network Pruning (FNNP) Workflow

Figure 4 visually illustrates the relation among $X1$, $X2$, Y. Pearson Correlation Coefficient (PCC) (Soper et al., 1917) $\rho(X, Y)$ is firstly tried to compare the correlation between $\rho(X1, Y)$ and $\rho(X2, Y)$. Apparently, $\rho(X1, Y)$, shown in Figure 4 right, form a clearer trend (solid line) while the $\rho(X2, Y)$ (Figure 4 left) is less trendy, which is also proved by the quantitative difference between $\rho(X1, Y) = 0.757$ and $\rho(X2, Y) = 0.204$. However, we emphasize a positive correlation between accuracy vectors and particularly care about the correlation about samples with high (top-k) accuracy in pruning tasks. Therefore, we propose a correlation metric as the following:

$$\phi_{XY}(k) = \frac{1}{k}\sum_{i=1}^{k} \min\left(\frac{k}{find(X, Y[i])}, 1\right), \tag{5}$$

where $Y[i]$ denotes the i-th best accuracy in Y. The function $find(X, Y[i])$ tries to insert the $Y[i]$ into the sorted X and returns its ranking. If the ranking is outside top k, the $min(*)$ function caps the fraction of $\frac{k}{find(X,Y[i])}$ with 1, otherwise the fraction item itself that accumulates within the range of $1 \leq i \leq k$. The intuition of Equation 5 is to highlight the matched high accuracy samples in both variables X and Y while ignoring the negative correlation as it is not expected trends in any evaluation process of pruning tasks.

	$\phi_{X1,Y}(5)$	$\phi_{X2,Y}(5)$
75% FLOPs	0.883	0.465
62.5% FLOPs	0.910	0.758
50% FLOPs	0.757	0.398

Table 1: Positive correlation quantified by $\phi_{XY}(k)$. Adaptive-BN-based evaluation largely improves the correlation compared to vanilla evaluation.

We calculated the proposed correlation coefficient with different pruning rates as shown in the first column in Table 1. The correlation from the adaptive-BN-based evaluation exceeds vanilla-evaluation-based methods by up to 0.418. Compared to vanilla experiments, the pruning candidates that gain high accuracy in the adaptive-BN-based evaluation are more likely to also perform well after fine-tuning. More correlation analysis can be found in Section 4.

3.3 Fast Neural Network Pruning Algorithm

Based on the discussion about the accurate evaluation process in pruning, we now present the overall workflow of our Fast Neural Network Pruning (FNNP) framework in Figure 3. Our pruning pipeline contains three parts, pruning strategy generation, filter pruning, and adaptive-BN-based evaluation.

Strategy generation outputs pruning strategies in the form of layer-wise pruning rate vectors like $(r_1, r_2, ..., r_L)$ for a L-layer model. The generation process follows pre-defined constraints such as inference latency, a global reduction of operations (FLOPs) or parameters and so on. Concretely, it randomly samples L real numbers from a given range $[0, R]$ to form a pruning strategy, where r_l

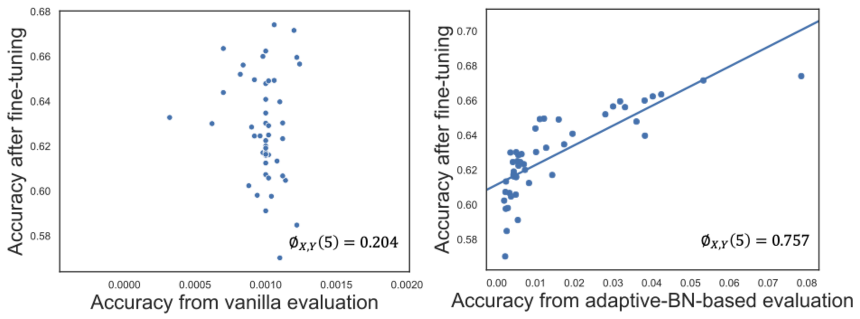

Figure 4: Correlation between fine-tuning accuracy and inference accuracy gained from vanilla evaluation (left), adaptive-BN-based evaluation (right) based on MobileNet V1 experiments on ImageNet Top-1 classification results)

denotes the pruning ratio for the l^{th} layer. R is the largest pruning ratio applied to a layer. Additionally, this module is replaceable and hence updatable. Though other strategy generation methods can be considered such as evolutionary algorithm, we found that a simple random sampling is good enough for the entire pipeline to quickly yield pruning candidates with state-of-the-art accuracy.

Filter pruning process prunes the full-size trained model according to the generated pruning strategy from the previous module. Similar to a normal filter pruning method, the filters are firstly ranked according to their $\mathcal{L}1$-norm and the r_l of the least important filters are trimmed off permanently. The sampled pruning candidates from the searching space are ready to be delivered to the next evaluation stage after this process.

The adaptive-BN-based candidate evaluation module provides a BN statistics adaptation and fast evaluation to the pruned candidates handed over from the previous module. Given a pruned network, it freezes all learnable parameters and traverses through a small amount of data in the training set to calculate the adaptive BN statistics $\hat{\mu}$ and $\hat{\sigma}^2$. In practice, we sampled 1/55 of the total training set for 50 iterations in our ImageNet experiments, which takes only 10-ish seconds in a single Nvidia 2080 Ti GPU. Next, this module evaluates the performance of the candidate networks on a small part of training set data, called sub-validation set, and picks the top ones in the accuracy ranking as winner candidates. The correlation analysis presented in Section 4 guarantees the effectiveness of this process. After a fine-tuning process, the winner candidates are finally delivered as outputs.

4 Experiments

As we claimed in previous sections, adaptive-BN-based evaluation allows a more efficient and effective pruning performance compared to existing methods. Section 4.1 shows the efficiency of the adaptive-BN-based evaluation and Section 4.2 presents the high pruning performance (effectiveness) of our FNNP algorithm. Both efficiency and effectiveness of our method are proposed through comparisons with existing state-of-the-art approaches.

This part involves pruning experiments on a compact model MobileNetV1 (Howard et al., 2017) and a larger model ResNet-50 (He et al., 2016) with various pruning rates. All experiments are based on the ImageNet dataset, which is a large-scale image classification dataset with 1000 classes containing about 1.28M training images and 50K validation images. We evaluate the pruning candidates on sub-validation set, which contains 10000 images, 10 for each class, randomly obtained from the original ImageNet training set.

4.1 Efficiency of FNNP compared to state-of-the-art methods

Our proposed pruning evaluation based on adaptive BN turn the prediction of subnet accuracy into a very fast and reliable process, so our FNNP is much less time-consuming to complete the entire pruning pipeline than other heavy evaluation based algorithms. In this part, we compare the execution cost for various state-of-the-art pruning algorithms to demonstrate the efficiency of our method.

Table 2 compares the computational costs of picking the best pruning strategy among 1000 potential pruning candidates. As ThiNet (Luo et al., 2017) and Filter Pruning (Li et al., 2016) require manually assigning layer-wise pruning ratio, The final GPU hours are the estimation of completing the pruning

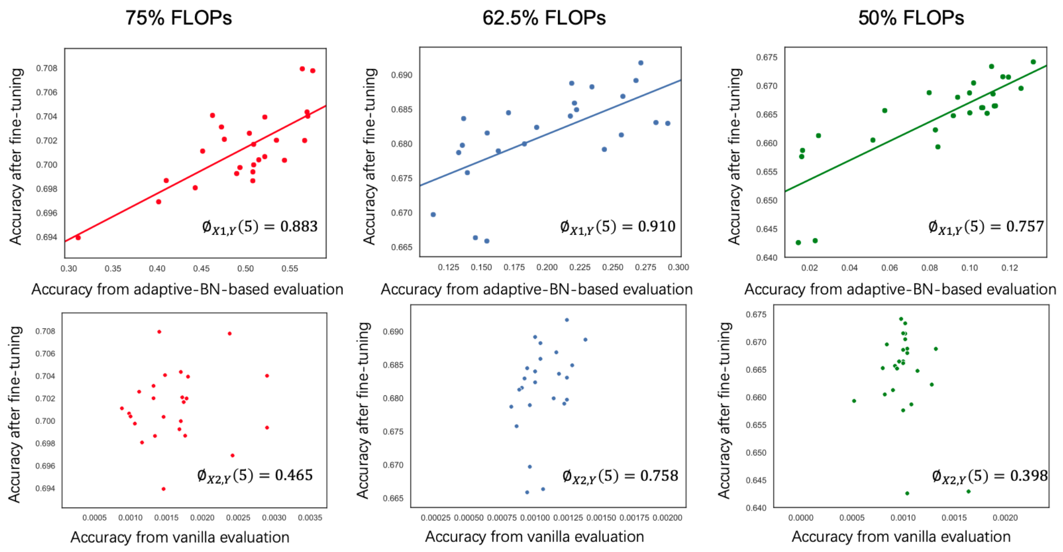

Figure 5: Vanilla vs. adaptive-BN evaluation: Correlation between evaluation and fine-tuning accuracy with different pruning ratios (MobileNet V1 on ImageNet classification Top-1 results)

pipeline for 1000 random strategies. In practice, the real computation cost highly depends on the expert's heuristic practice of trial-and-error. The computation time for AMC (He et al., 2018c) and Meta-pruning can be long because training either an RL network or an auxiliary network itself is time-consuming and tricky. Among all compared methods, our FNNP is the most efficient method as each evaluation takes no more than 50 iterations, which takes 10 to 20 seconds in a single Nvidia 2080 Ti GPU. So the total candidate selection is simply an evaluation comparison process, which also can be done in no time.

Method	**Evaluation Method**	**Candidate Selection**	**GPU Hours**
ThiNet (Luo et al., 2017)	finetuning	1000×10 finetune epochs	~ 8000
NetAdapt (Yang et al., 2018b)	finetuning	10^4 training iterations	864
Filter Pruning (Li et al., 2016)	vanilla	1000×25 finetune epochs	~ 20000
AMC (Yang et al., 2018b)	vanilla	Training an RL agent	-
Meta-Pruning (Liu et al., 2019b)	PruningNet	Training an auxiliary network	-
FNNP	adaptive-BN	$< 1000 \times 50$ inference iterations	25

Table 2: Comparison of computation costs of various pruning methods in the task where all pruning methods are executed to find the best pruning strategy from 1000 potential strategies (candidates).

4.2 Effectiveness of FNNP compared to state-of-the-art methods

Firstly, we show more details about the correlation between evaluated accuracy and fine-tuning accuracy for adaptive-BN-based and vanilla evaluation separately in different pruning rates. Each vertical pair of sub-figures in Figure 5 shows the above comparison in a particular pruning rate marked at the column top. Noticeably, the X axis presents a much smaller value range in vanilla evaluation sub-figures because, as analyzed in Section 3, the global BN statistical values messed up the inference accuracy while the adaptive BN allows pruning candidates to fall into a reasonably higher accuracy range. The correlations for our adaptive-BN-based evaluation are 0.42, 0.15 and 0.36 higher than the vanilla evaluation in 75%, 62.5%, and 50% pruning rates separately.

Secondly, we compare our method with state-of-the-art methods on MobileNetV1 and ResNet-50 to prove its effectiveness. FNNP outperforms all of our tested methods in all experiments.

ResNet We compare the top-1 accuracy of ResNet-50 on ImageNet under different FLOPs constraints. For each FLOPs constraint (3G, 2G, and 1G), 1000 pruning strategies are generated. Then the adaptive-BN-based evaluation method is applied to each candidate. We just fine-tune the top-2 candidates and return the best as delivered pruned model. It is shown that FNNP outperforms other methods under the constraints listed in Table 3.

ThiNet (Luo et al., 2017) prunes the channels uniformly for each layer other than finding an optimal pruning strategy, which hurts the performance significantly. Meta-Pruning (Liu et al., 2019b) trains an auxiliary network called "PruningNet" to predict the weights of the pruned model. But the adopted vanilla evaluation may mislead the searching of the pruning strategies. As shown in Table 3, our FNNP algorithm outperform all compared methods given different pruned network targets.

Operations (FLOPs) after pruning	Method	FLOPs	Top1-Acc
3G	ThiNet-70 (Liu et al., 2019b)	2.9G	75.8%
	AutoSlim (Yu & Huang, 2019)	3.0G	76.0%
	Meta-Pruning (Liu et al., 2019b)	3.0G	76.2%
	FNNP	3.0G	**77.1%**
2G	0.75 × ResNet-50 (He et al., 2016)	2.3G	74.8%
	Thinet-50 (Luo et al., 2017)	2.1G	74.7%
	AutoSlim (Yu & Huang, 2019)	2.0G	75.6%
	CP (He et al., 2017)	2.0G	73.3%
	FPGM (He et al., 2018b)	2.31G	75.59%
	SFP (He et al., 2018a)	2.32G	74.61%
	GBN (You et al., 2019)	1.79G	75.18%
	GDP (Lin et al., 2018)	2.24G	72.61%
	DCP (Zhuang et al., 2018)	1.77G	74.95%
	Meta-Pruning (Liu et al., 2019b)	2.0G	75.4%
	FNNP	2.0G	**76.4%**
1G	0.5 × ResNet-50 (He et al., 2016)	1.1G	72.0%
	ThiNet-30 (Luo et al., 2017)	1.2G	72.1%
	AutoSlim (Yu & Huang, 2019)	1.0G	74.0%
	Meta-Pruning (Liu et al., 2019b)	1.0G	73.4%
	FNNP	1.0G	**74.2%**

Table 3: Comparisions of ResNet-50 and other pruning methods

MobileNet MobileNets are more challenging to prune on large datasets like ImageNet as they are already very compact. We compare the top-1 ImageNet classification accuracy under the same FLOPs constraint (about 280M FLOPs). 1500 pruning strategies are generated with this FLOPs constraint. Then adaptive-BN-based evaluation is applied to each candidate. After fine-tuning the top-2 candidates, the pruning candidate that returns the highest accuracy is selected as the final output.

AMC (He et al., 2018c) trains their pruning strategy decision agent based on the pruned model without fine-tuning, which may lead to a problematic selection on the candidates. NetAdapt (Yang et al., 2018b) searches for the pruning strategy based on a greedy algorithm, which may drop into a local optimum. It is shown that FNNP achieves the best performance among all studied methods again in this task (see Table 4).

Method	FLOPs	Top1-Acc
0.75 × MobileNetV1 (Howard et al., 2017)	325M	68.4%
AMC (He et al., 2018c)	285M	70.5%
NetAdapt (Yang et al., 2018b)	284M	69.1%
Meta-Pruning (Liu et al., 2019b)	281M	70.6%
FNNP	284M	**70.9%**

Table 4: Comparisions of MobileNetV1 and other pruning methods

5 Discussion and Conclusions

We presented our FNNP framework, in which a fast and accurate evaluation based on adaptive batch normalization is proposed. To quantitatively show the advantages of this module over other methods, a correlation coefficient is proposed. Our experiments show the efficiency and effectiveness of our FNNP method by delivering better pruning performance than our studied approaches.

Apart from the study shown in this work, we believe our adaptive-BN-based evaluation module is general enough to plug-in and improve existing works. For example, the "short-term fine-tune" block in (Yang et al., 2018a) can be seamlessly replaced by our adaptive-BN-based module for a faster sub-net selection. (He et al., 2018c) may also efficiently train its reinforcement learning agent with the winner pruning candidates generated from our adaptive-BN-based evaluation module. On the other side, our pipeline can also be upgraded by existing methods such as the evolutionary algorithm used in (Liu et al., 2019b) to improve the basic random strategy.

REFERENCES

Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *NIPS*, 2019a.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019b.

Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. Scarletnas: Bridging the gap between scalability and fairness in neural architecture search. *CoRR*, abs/1908.06022, 2019a.

Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *CoRR*, abs/1907.01845, 2019b.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *CoRR*, abs/1904.00420, 2019.

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*, pp. 770–778, 2016.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018a.

Yang He, Ping Liu, Ziwei Wang, and Yi Yang. Pruning filter via geometric median for deep convolutional neural networks acceleration. *arXiv preprint arXiv:1811.00250*, 2018b.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018c.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, pp. 2425–2432, 2018.

Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019a.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. *arXiv preprint arXiv:1903.10258*, 2019b.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

HE Soper, AW Young, BM Cave, Alice Lee, and Karl Pearson. On the distribution of the correlation coefficient in small samples. appendix ii to the papers of" student" and ra fisher. *Biometrika*, 11 (4):328–413, 1917.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pp. 2074–2082, 2016.

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications. apr 2018a. URL `http://arxiv.org/abs/1804.03230`.

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 285–300, 2018b.

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jiahui Yu and Thomas Huang. Network slimming by slimmable networks: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 875–886, 2018.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.