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 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 ABN-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 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 70.7% 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 fluctuation. 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 part that makes little contribution to the delivered model accuracy is no trivial question.

If the original full-size model is considered as a search space, a sub-net retaining the most valuable information is searched while the rest of the structure is trimmed off. Existing pruning methods target on this question from two perspectives. The first one is to affect the search space so that it is easier to find out the interesting part(Ding et al., 2019) (Wen et al., 2016)(Liu et al., 2017) (Yu et al., 2018)(Zhuang et al., 2018)(Gordon et al., 2018). Another perspective is to improve the searching method, for example, by introducing reinforcement learning agents (He et al., 2018c), evolutionary algorithm (Liu et al., 2019b), knowledge distillation (Luo et al., 2017) and so on.

However, it requires high searching efforts to obtain a good pruning candidate network due to two main reasons: inaccurate evaluation method and/or high algorithmic complexity.

Firstly, we define what evaluation is in DNN pruning tasks. To evaluate if a pruned network, referred as sub-net or pruning candidate in this work, is equipped with the potential of delivering high model accuracy, there would normally be an evaluation process. Such process preempts a sub-net's potential by applying an evaluation process for a quick judgement rather than actually training the sub-net to its settled model accuracy because it can be very time-consuming to do so. For example, (He et al., 2018c) trains an reinforcement learning agent to guarantee that a good pruning strategy returns high reward. (Liu et al., 2019b) builds up a "PruningNet" to learn how to generate weights for any pruning candidate so that it delivers high accuracy. Another example is seen in (Yu & Huang,

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

2019b), which trains the full model itself in such a way that it is capable of showing the accuracy of the pruned structure with arbitrary pruning ratio.

No matter how complicated such evaluation process is designed, the eventual purpose is to equip the evaluation module with a capability that it returns the sub-nets' potential quickly in inference time and the potential is positively related to its final settled accuracy. This evaluator is then either used to yield high-potential candidates for a succeeding comparing/selection module (He et al., 2018c)(Yang et al., 2018b), or directly winner sub-net as its final deliverable (Yu et al., 2018)(Yu & Huang, 2019b).

As aforementioned, we found that the evaluation methods in existing works are either inaccurate or too complicated to be generally applied. Inaccurate evaluation means that selected wining sub-nets can not necessarily be fine-tuned to a high accuracy due to an inaccurate evaluation design. In this work, we provide a quantitative analysis to show that inaccurate evaluation methods delivers poor correlation between evaluated performance of sub-nets and their final converged accuracy. On the other side, Complicated evaluation means some carefully-designed evaluation methods that may introduce many hyper-parameters to components like reinforcement learning agents (He et al., 2018c), generative adversarial learning (Lin et al., 2019), knowledge distillation (Luo et al., 2017) and so on. It can be challenging to repeat the exact experimental results without treating the evaluation module carefully. Therefore, these methods are reckon to have relatively high algorithmic complexity that prevents them from being general applied.

To address above mentioned issues, we propose a fast and accurate evaluation method in our proposed pruning algorithm. The main novelty present in this work is described as below:

- An automated pruning method is proposed called Fast Neural Network Pruning (FNNP). Our pruning method is compatible to typical neural network training pipeline and do not change the original parameter distribution.
- In our proposed FNNP framework, the key component is an Adaptive Batch Normalization (ABN) based evaluation module. It delivers strong correlation between accuracy of pruning candidates and their final converged accuracy. Such correlation is described by our proposed correlation coefficient, which is used to quantitatively show why some evaluation methods in existing works are inaccurate.
- Our proposed FNNP outperforms many carefully-designed pruning methods by achieving higher accuracy in situations where the same base model are pruned by different levels of pruning. In the Res-Net 50 experiments, our FNNP outperforms all compared methods by 1.3 % to 3.9 %. Even in the challenging task of pruning compact model of MobileNet V1, our FNNP achieves the highest accuracy of 70.6% 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 to pruning search space and proposal of different searching methods.

Modification to pruning search space 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 techniques, i.e. our proposed fast evaluation method can be used to such regularized searching space.

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

As explained in Section 1, inaccurate evaluation is mainly caused by improperly using global BN statistics to sub-nets. For example, in (Li et al., 2016), a sensitivity analysis is applied to DNNs, which essentially is an examination about how model accuracy is affected by conducting layer-by-layer pruning. However, for fast evaluation, sub-nets are directly evaluated with validation dataset using the global BN statistics. Our experiments show that such global BN based model performance has very weak correlation to the model's final converged accuracy (Figure 3) and hence sensitivity curves drawn in (Li et al., 2016) is inaccurate. The same problem can be seen in (He et al., 2018c).

A short-term fine-tune block proposed in (Yang et al., 2018b) can fix the BN statistics to some extend. But at least 10^4 iterations are required to additionally train each pruning candidate. So this block can be time consuming considering the large amount of plausible pruning candidates. Meta-pruning (Liu et al., 2019b) separately trains a "PruningNet" to generate weights for pruning candidates, so the sub-nets can be formed in inference time and evaluated quickly. But the "PruningNet" is consist of fully-connected layers and introduces a few millions of extra trainable weights, leading to a memory-heavy training process in hardware. Furthermore, some introduced evaluation components requiring skillful hyper-parameter tuning such as reinforcement learning agents (He et al., 2018c), generative adversarial learning (Lin et al., 2019), knowledge distillation (Luo et al., 2017) and so on. It can be difficult to repeat the exact experimental results without carefully initializing and tuning the evaluation blocks. We consider above pruning techniques having either slow or complicated evaluation process. In this work, we aim to use adaptive BN to implement a simple, fast yet efficient sub-net evaluation process that delivers the state-of-the-art pruning performance.

By recalculating the statistics for BN in the pruned sub-net, our adaptive BN can be quickly adapted from the full network to the pruned networks. Thus, the evaluation of the pruned networks can be accomplished in a flash, which significantly accelerates the process of network pruning. The adaptive BN has also been noticed by literature. In the classical recognition tasks, recalculating the statistics has been used to improve the recognition accuracies (Wang et al., 2018). (Li et al., 2018) also use adaptive BN for cross-domain recognition. In the network architecture design, post-BN is used to boost the network capacity (Luo et al., 2019). The recent neural architecture search also use adaptive BN for fast evaluation of the networks architecture (Guo et al., 2019; Chu et al., 2019b;a). Even in the network pruning, (Yu et al., 2018; Liu et al., 2019b) also use post BN to improve the inference accuracy for the sub-nets or the agency networks. However, none of the above works analyzes the property of adaptive BN, especially why adaptive BN works. In contrast, in our works, we have an insight into the correlation between the network used recalculated statistics and the networks trained to convergence.

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 pretrained weight, i.e., they separate the network architecture and network parameter. This contradicts our knowledge that some tasks have to be pretrained by ImageNet.

3 METHODOLOGY

A typical neural network training and pruning pipeline is generalized and visualised in Figure 1.

After a model is trained, pruning is applied to the full-size model. In this work, we focus on structured filter pruning as it does not introduce further sparsity to the survived kernels. A filter pruning task can be 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 training loss and $\mathcal{A}$ is the neural network model. r_l is the pruning ratio applied to the l^{th} layer. Given some constraints $\mathcal{C}$ (i.e. aimed amount of parameters, operations or execution latency), a group of pruning candidates can be evaluated to find out the best pruning ratio combination $(r_1, r_2, ..., r_L)^*$, also referred as pruning strategy, in an evaluation process. such pruning candidates,

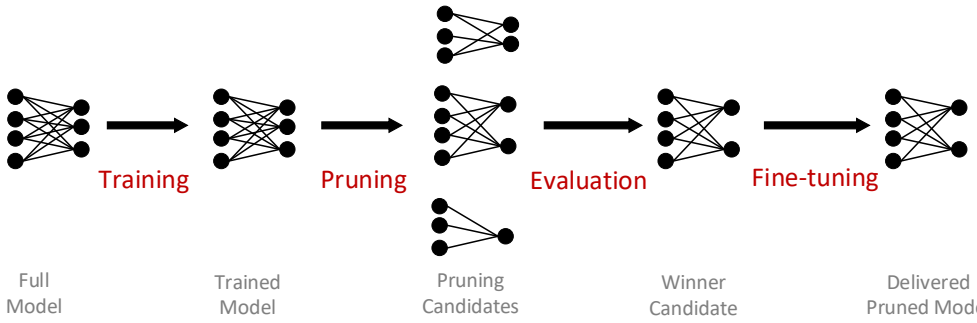

Figure 1: A typical pipeline for neural network training and pruning

or sub-nets, form a searching space and the evaluation module picks a winner candidate with the best potential. Finally, the winner candidate is optionally fine-tuned to reach the final converged accuracy before delivered from the end of the pipeline.

As discussed in Section 2, the evaluation methods in existing works are found to be inaccurate or complicated. As an example, in (Li et al., 2016) and He et al. (2018c), the reason that we consider their evaluation as inaccurate is the use of global BN statistics. To quantitatively demonstrate the idea, we symbolize the BN Ioffe & Szegedy (2015) as below:

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

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:

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

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

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

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^2_T are used in testing phase.

We realize that in order to apply fast evaluation to the pruning candidates (Figure 1), what we all need to do, is to obtain the adapted μ and σ to each pruned structure. The Adapted Bach Normalization (ABN) statistical values, noted as $\hat{\mu}$ and $\hat{\sigma}^2$ in this work, are obtained by sampling a small part of training data for a few more iterations.

However, in (Li et al., 2016) and (He et al., 2018c), μ_T and σ_T, called Post-Training BN (PBN) Statistics in this work, are directly used to evaluate different pruning candidates. We compare the correlation between evaluated model and fine-tuned model in terms of test accuracy. The results are shown in Figure 3 and Figure 3 respectively. From an observation, PBN statistics present poor correlation, compared to the stronger correlation unveiled by ABN. This explains why PBN-based evaluation used in many existing methods is inaccurate. A "short-term fine tune block" block in (Liu et al., 2019b) is also proposed to fix the issue of inaccurate BN statistics. An one-epoch short-term fine-tuning noticeably help to deliver strong correlation in the evaluation process, as shown in Figure 3. Training 1 epoch takes relatively much longer time. To quantitatively describe such observation, we introduce a correlation index in Section 4 together with its related discussion.

Based on the fast and more accurate evaluation module, we now present the overall workflow of our Fast Neural Network Pruning (FNNP) framework in Figure 2.

Our pruning pipeline contains three main parts, pruning strategy generation, filter pruning and ABN-based candidate evaluation.

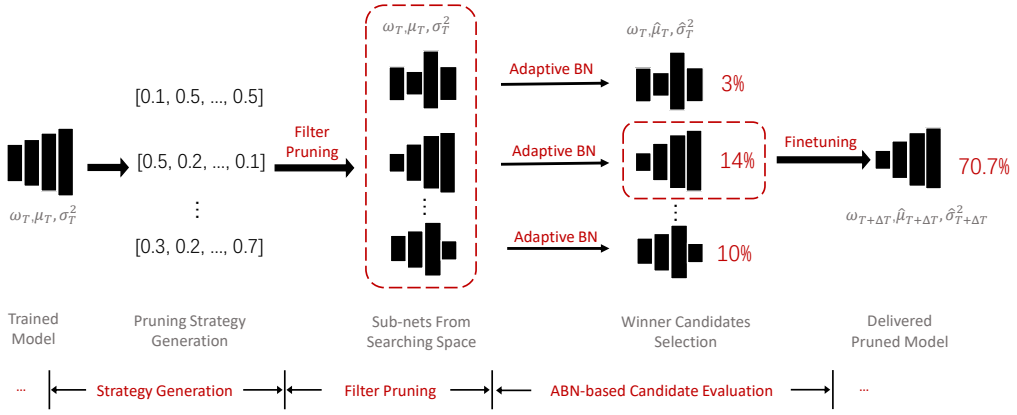

Figure 2: Fast Neural Network Pruning (FNNP) framework

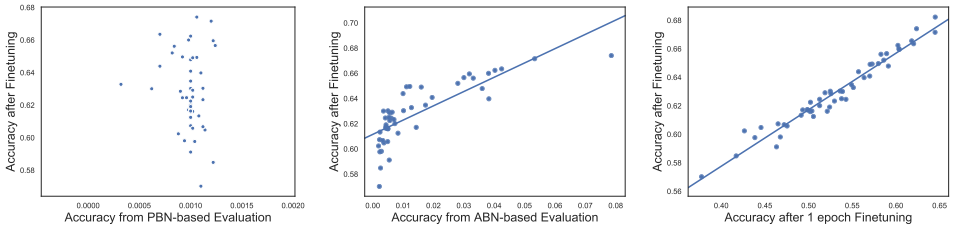

Figure 3: Correlation between fine-tuning accuracy and post-evaluation accuracy with three different evalution methods. From left to right: PBN-based, ABN-based and 1 epoch fine-tuning (MobileNet V1 on ImageNet classification Top-1 results)

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 a pre-defined constraints such as inference latency, global reduction in terms of operation (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 denotes the pruning ratio for the l^{th} layer. R is the largest pruning ratio applied to a layer and it forms the boundary of sub-net searching space. In later part, we will present the ABN presented correlation is affected by R. Additionally, other generation methods can be used to this module such as evolutionary algorithms. In this work, we found that a simple random sampling is good enough for the entire pipeline to quickly yield sub-nets with state-of-the-art accuracy.

Filter pruning block prunes the full-size trained model following the generated pruning vectors from the strategy generation module. The filters in the baseline module are ranked according to their $\mathcal{L}1$-norm and the r_l of the smallest 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 ABN-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 traverse through a small amount of data in the training set to calculate the ABN statistics $\hat{\mu}$ and $\hat{\sigma}^2$. In practice, we sampled 1/55 of the total training set for 30 iterations in our ImageNet experiments, which takes only 10-ish seconds in a single Nvidia 1080 Ti GPU. Next, this module evaluates the performance of the candidate networks on the 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 pruned winner candidates are finally delivered as output.

4 Experiments

In this section, we first demostrate the effectiveness and characteristic of the proposed network evaluating method(adaptive BN). Further we illustrate its efficiency. Finally we compare our method with state-of-the-art filter pruning methods to prove its effectiveness.

4.1 CORRELATION BETWEEN THE PRUNED & THE CONVERGED MODEL

Basic settings We perform correlation experiments with MobileNetV1 (Howard et al., 2017) on ImageNet (Deng et al., 2009). ImageNet is large-scale image classification dataset with 1000 classes. It contains about 1.28M training images and 50K validation images. MobileNetV1 (Howard et al., 2017) is a typical light-weighted convolution neural network. We first train the model to converged and randomly generate 50 pruning strategies to prune out filters different ratios(0 to 0.7) for each layer. We then fine-tune each pruned model to converged(75k iterations, batch size is 512).

Correlation analysis We first visualize the correlation between the pruned model and the converged model in different settings(before adaptive BN, after adaptive BN, fine-tuning for 1 epoch), See Figure 3. We also caculate the correlation coefficients (Soper et al., 1917) of them(-0.12, 0.70 and 0.97).

For more quantitive analysis, we introduce a new metirc to evaluate the positive correlation. Given a set of models, let X and Y denote the accuracies of these models evaluated in different aspects. For example, X and Y are the accuracies of the pruned models either tested by using our adaptive BN or by finetuning them until convergence. Moreover, X and Y have been sorted in a descending order. Then, the correlation between X and Y is defined as

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

where $Y[i]$ denotes the top-i best accuracies in Y. Ideally, the top-1 model is the target we are searching for. However, obtaining the top-1 model is difficult due to the NP-hard problem. Thus, we relax top-1 to top-k (e.g. top-5). Here, $find(A, a)$ is a function that returns the index of a in A. Intuitively, $\frac{k}{find(X,Y[i])}$ implies whether a top accuracy in Y is also high in X. The function $\min(\cdot)$ regularizes the correlation to be smaller than 1. In this new metric, a higher correlation rate $\phi_{XY}(k)$ implies a high correlation between X and Y.

	$\phi_{\text{ABN, finetuned to convergence}}(5)$	$\phi_{\text{PBN, finetuned to convergence}}(5)$
75% FLOPs	0.883	0.465
62.5% FLOPs	0.910	0.758
50% FLOPs	0.757	0.398

Table 1: Positive correlation evaluation by newly introduced metric $\phi_{XY}(5)$

It is shown that, their is little correlation beteween PBN and the converged accuracy. The correlation is much stronger between our ABN and the converged accuarcy. AMC (He et al., 2018c) and Filter Pruning (Li et al., 2016) decide their pruning strategy base on PBN, which may harm their performances.

To evaluate the boundary of the correlation, we also conduct experiments under different FLOPs constraints(75%, 50% and 25% of the original model). For each constraints, we randomly sample 20 strategies, see Figure 4.

4.2 EFFICIENCY OF EVALUATION METHODS

To demostrate the efficiency of our evaluation method, we first show the number of iterations needed by our evaluation method. Then we compare it to others.

To evaluate a pruning strategy by adaptive-BN-based method, we first prune out half of the filters for each layer in MobileNetV1 (Howard et al., 2017) and re-caculate the BN statistics through iterations on ImageNet (Deng et al., 2009) training samples. Here we show the validation accuracy after each iteration. See Figure 4.2, as the number of iteration increase, the validation accuracy go up quickly and finally converge in less than 50 iterations.

We compare our adaptive-BN based method with other evaluation methods by their additional costs, including efforts for obtaining an evaluatoin module and perform one evaluation, see Table 2. ThiNet (Luo et al., 2017), NetAdapt (Yang et al., 2018b), Filter Pruning (Li et al., 2016) needs no less than 10^4 iterations. AMC (Yang et al., 2018b) and Meta-Pruning (Liu et al., 2019b) require to train an evaluation agent. But our method needs no more than 50 iterations.

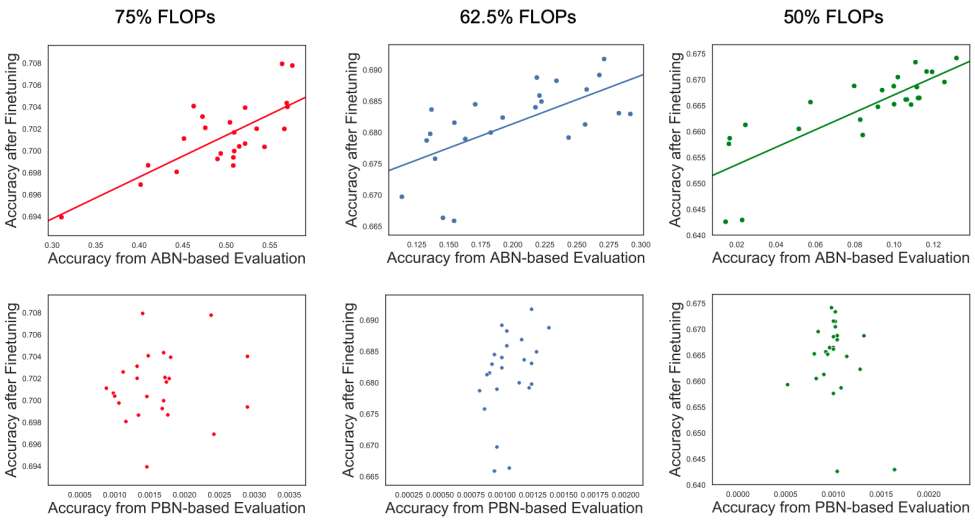

Figure 4: PBN vs. ABN: Correlation between post-evaluation accuracy and fine-tuning accuracy with different pruning ratios to operations (MobileNet V1 on ImageNet classification Top-1 results)

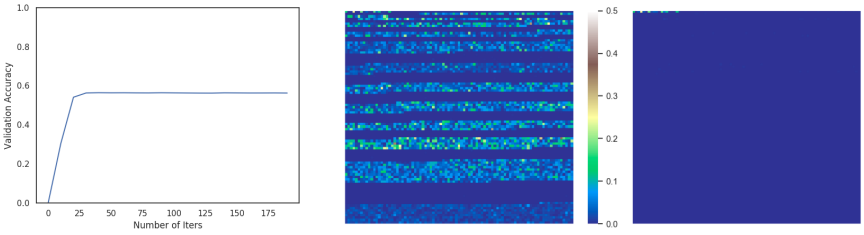

Figure 5: **Left:** Relation between number of sampling iterations and ABN-based evaluation results. **Right:** Distance to expected BN statistics. Left is the distance of mean values between PBN sampled from training data and expected BN sampled from validation data (true statistics). Right is the same distance but between ABN sampled from training data and the same expected BN sampled from validation data.

Method	Evaluation Cost
ThiNet (Luo et al., 2017)	2.5×10^5 iterations(batch size is 512)
NetAdapt (Yang et al., 2018b)	10^4 iterations (batch size is 96)
Filter Pruning (Li et al., 2016)	5×10^4 iterations(batch size is 512)
AMC (Yang et al., 2018b)	Training an evaluation agent
Meta-Pruning (Liu et al., 2019b)	Training an evaluation agent
FNNP	50 iterations(batch size is 512)

Table 2: Comparisions of evaluation costs

4.3 Comparisons with state-of-the-art methods

We compare our method with state-of-the-art methods on MobileNetV1 (Howard et al., 2017) and ResNet-50 (He et al., 2016) to prove its effectiveness.

ResNet We compare the top-1 accuracy of ResNet-50 (He et al., 2016) on ImageNet (Deng et al., 2009) under different FLOPs constraints. For each FLOPs constraint(3G, 2G and 1G), we first generate 1000 pruning strategies. Then we apply our ABN-based evaluating method for each candidates. We just fine-tune the top-2 candidates and return the best as delivered pruned model. It is shown that FNNP outerperforms 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 largely. Meta-Pruning (Liu et al., 2019b) trains a PruningNet to predict weights of the pruned model. But the predicted weights without fin-tuning may not be accurate to achieve high accuracy when testing. Moreover, this inaccuracy may illy

guide the searching of pruning strategies. The effectiveness of our ABN-based evaluation methods help us to find out the pruned model with optimal structure and weights.

Group by FLOPs	Method	FLOPs	Top1-Acc
3G	ThiNet-70 (Liu et al., 2019b)	2.9G	75.8%
	AutoSlim (Yu & Huang, 2019a)	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, 2019a)	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, 2019a)	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 We compare the top-1 accuracy of MobileNetV1 (Howard et al., 2017) on ImageNet (Deng et al., 2009) under same FLOPs constraint(about 280M FLOPs). We first generate 1500 pruning strategies that meet the FLOPs constraint. We then apply our ABN-based evaluating method for each candidates. After fine-tuning the top-2 candidates, we return the best of them as the compressed model.

AMC (He et al., 2018c) trains their pruning strategy decision agent based on pruned model without fine-tuning, which illy affect the decision. 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 a lso achieve state-of-the-art, see Table 3.

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	**70.7%**

Table 4: Comparisions of MobileNetV1 and other pruning methods

5 Discussion and Conclusions

We presented our FNNP framework, in which a fast and accurate sub-net evaluation module called ABN-based evaluation. To quantitatively show the advantages of this module over other inaccurate evaluation methods, a correlation coefficient is proposed and analysed.

Apart from the study shown in this work, we believe our ABN-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 replaced by our ABN-based evaluation module for a faster sub-net selection. (He et al., 2018c) can also efficiently train its reinforcement learning agent with our generated wining pruning candidates. On the other side, our pipeline can also be upgraded by existing methods for better performance. For example, the evolutionary algorithm used in (Liu et al., 2019b) can be used to improve the basic random strategy in our strategy generation module.

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.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4943–4953, 2019.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1586–1595, 2018.

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.

Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

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.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

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.

Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

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.

Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 7794–7803, 2018.

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*, 2019a.

Jiahui Yu and Thomas Huang. Universally slimmable networks and improved training techniques. *arXiv preprint arXiv:1903.05134*, 2019b.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

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.