Good Subnetworks Provably Exist: Pruning via Greedy Forward Selection

Mao Ye [1] Chengyue Gong [*] [1] Lizhen Nie [*] [2] Denny Zhou [3] Adam Klivans [1] Qiang Liu [1]

Abstract

Recent empirical works show that large deep neural networks are often highly redundant and one can find much smaller subnetworks without a significant drop of accuracy. However, most existing methods of network pruning are empirical and heuristic, leaving it open whether good subnetworks provably exist, how to find them efficiently, and if network pruning can be provably better than direct training using gradient descent. We answer these problems positively by proposing a simple greedy selection approach for finding good subnetworks, which starts from an empty network and greedily adds important neurons from the large network. This differs from the existing methods based on backward elimination, which remove redundant neurons from the large network. Theoretically, applying the greedy selection strategy on sufficiently large pre-trained networks guarantees to find small subnetworks with lower loss than networks directly trained with gradient descent. Our results also apply to pruning randomly weighted networks. Practically, we improve prior arts of network pruning on learning compact neural architectures on ImageNet, including ResNet, MobilenetV2/V3, and ProxylessNet. Our theory and empirical results on MobileNet suggest that we should fine-tune the pruned subnetworks to leverage the information from the large model, instead of re-training from new random initialization as suggested in Liu et al. (2019b).

1. Introduction

The last few years have witnessed the remarkable success of large-scale deep neural networks (DNNs) in achieving human-level accuracy on complex cognitive tasks, including image classification (e.g., He et al., 2016), speech recognition (e.g., Amodei et al., 2016) and machine translation (e.g., Wu et al., 2016). However, modern large-scale DNNs tend to suffer from slow inference speed and high energy cost, which form critical bottlenecks on edge devices such as mobile phones and Internet of Things (IoT) (Cai et al., 2019). It is of increasing importance to obtain DNNs with small sizes and low energy costs.

Network pruning has been shown to be a successful approach for learning small and energy-efficient neural networks (e.g., Han et al., 2016b). These methods start with a pre-trained large neural network and remove the redundant units (neurons or filters/channels) to obtain a much smaller subnetwork without significant drop of accuracy. See e.g., Zhuang et al. (2018); Luo et al. (2017); Liu et al. (2017; 2019b); He et al. (2019; 2018b) for examples of recent works.

However, despite the recent empirical successes, thorough theoretical understandings on why and how network pruning works are still largely missing. Our work is motivated by the following basic questions:

The Subnetwork Problems: *Given a pre-trained large (over-parameterized) neural network, does there exist a small subnetwork inside the large network that performs almost as well as the large network? How to find such a good subnetwork computationally efficiently? Does the small network pruned from the large network provably outperform the networks of same size but directly trained with gradient descent starting from scratch?*

We approach this problem by considering a simple greedy selection strategy, which starts from an empty network and constructs a good subnetwork by sequentially adding neurons from the pre-trained large network to yield the largest immediate decrease of the loss (see Figure 1(left)). This simple algorithm provides both strong theoretical guarantees and state-of-the-art empirical results, as summarized below.

Greedy Pruning Learns Good Subnetworks For two-layer neural networks, our analysis shows that our method yields a network of size n with a loss of $\mathcal{O}(1/n) + \mathcal{L}_N^*$, where $\mathcal{L}_N^*$ is the optimal loss we can achieve with all

[*]Equal contribution [1]Department of Computer Science, the University of Texas, Austin [2]Department of Statistics, the University of Chicago [3]Google Research. Correspondence to: Mao Ye <my21@cs.utexas.edu>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

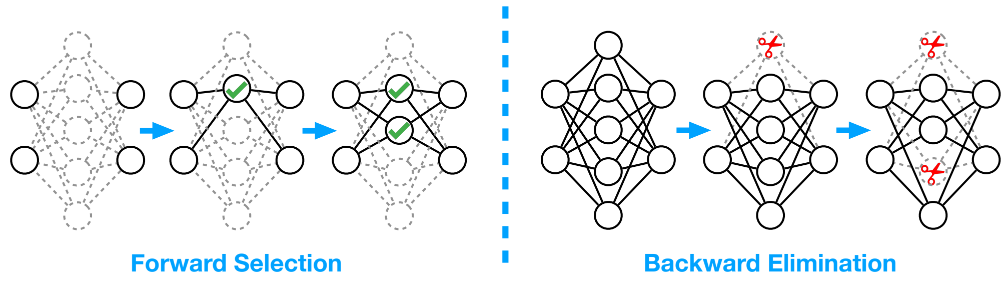

Figure 1. Left: Our method constructs good subnetworks by greedily adding the best neurons starting from an empty network. Right: Many existing methods of network pruning works by gradually removing the redundant neurons starting from the original large network.

the neurons in the pre-trained large network of size N. Further, if the pre-trained large network is sufficiently over-parametrized, we achieve a much smaller loss of $\mathcal{O}(1/n^2)$. Additionally, the $\mathcal{O}(1/n^2)$ rate holds even when the weights of the large network are drawn i.i.d. from a proper distribution.

In comparison, standard training of networks of size n by gradient descent yields a loss of $\mathcal{O}(1/n + \varepsilon)$ following the mean field analysis of Song et al. (2018); Mei et al. (2019), where ε is usually a small term involving the loss of training infinitely wide networks; see Section 3.3 for more details.

Therefore, our fast $\mathcal{O}(1/n^2)$ rate suggests that pruning from over-parameterized models guarantees to find more accurate small networks than direct training using gradient descent, providing a theoretical justification of the widely used network pruning paradigm.

Selection vs. Elimination Many of the existing methods of network pruning are based on *backward elimination* of the redundant neurons starting from the full large network following certain criterion (e.g., Luo et al., 2017; Liu et al., 2017). In contrast, our method is based on *forward selection*, progressively growing the small network by adding the neurons; see Figure 1 for an illustration. Our empirical results show that, our forward selection achieves better accuracy on pruning DNNs under fixed FLOPs constraints, e.g., ResNet (He et al., 2016), MobileNetV2 (Sandler et al., 2018), ProxylessNet (Cai et al., 2019) and MobileNetV3 (Howard et al., 2019) on ImageNet. In particular, our method outperforms all prior arts on pruning MobileNetV2 on ImageNet, achieving the best top1 accuracy under any FLOPs constraint.

Additionally, we draw thorough comparison between the forward selection strategy with the backward elimination in Appendix 11, and demonstrate the advantages of forward selection from both theoretical and empirical perspectives.

Rethinking the Value of Network Pruning Both our theoretical and empirical discoveries highlight the benefits of using *large, over-parameterized* models to learn small models that *inherit the weights* of the large network. This implies that in practice, we should *finetune* the pruned network to leverage the valuable information of both the structures and parameters in the large pre-trained model.

However, these observations are different from the recent findings of Liu et al. (2019b), whose empirical results suggest that training a large, over-parameterized network is often not necessary for obtaining an efficient small network and finetuning the pruned subnetwork is no better than re-training it starting from a new random initialization.

We think the apparent inconsistency happens because, different from our method, the pruning algorithms tested in Liu et al. (2019b) are not able to make the pruned network efficiently use the information in the weight of the original network. To confirm our findings, we perform tests on compact networks on mobile settings such as MobileNetV2 (Sandler et al., 2018) and MobileNetV3 (Howard et al., 2019), and find that finetuning a pruned MobileNetV2/MobileNetV3 gives much better performance than re-training it from a new random initialization, which violates the conclusion of Liu et al. (2019b). Besides, we observe that increasing the size of pre-trained large models yields better pruned subnetwork as predicted by our theory. See Section 4.2 and 4.3 for a thorough discussion.

Notation We use notation $[N] := \{1, \dots, N\}$ for the set of the first N positive integers. All the vector norms $\|\cdot\|$ are assumed to be ℓ_2 norm. $\|\cdot\|_{\mathrm{Lip}}$ and $\|\cdot\|_\infty$ denote Lipschitz and ℓ_∞ norm for functions. We denote $\mathrm{supp}(\rho)$ as the support of distribution ρ.

2. Problem and Method

We focus on two-layer networks for analysis. Assume we are given a pre-trained large neural network consisting of N neurons,

$$f_{[N]}(\boldsymbol{x}) = \sum_{i=1}^{N} \sigma(\boldsymbol{x}; \boldsymbol{\theta}_i)/N,$$

where $\sigma(\boldsymbol{x};\boldsymbol{\theta}_i)$ denotes the i-th neuron with parameter $\boldsymbol{\theta}_i \in \mathbb{R}^d$ and input $\boldsymbol{x}$. In this work, we consider

$$\sigma(\boldsymbol{x};\boldsymbol{\theta}_i) = b_i \sigma_+(\boldsymbol{a}_i^\top \boldsymbol{x}),$$

where $\boldsymbol{\theta}_i = [\boldsymbol{a}_i, b_i]$ and $\sigma_+(\cdot)$ is an activation function such as Tanh and ReLU. But our algorithm works for general forms of $\sigma(\boldsymbol{x};\boldsymbol{\theta}_i)$. Given an observed dataset $\mathcal{D}_m := (\boldsymbol{x}^{(i)}, y^{(i)})_{i=1}^m$ with m data points, we consider the following regression loss of network f:

$$\mathcal{L}[f] \quad = \quad \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_m}[(f(\boldsymbol{x})-y)^2]/2.$$

We are interested in finding a subset S of n neurons ($n < N$) from the large network, which minimizes the loss of the subnetwork $f_S(\boldsymbol{x}) = \sum_{i\in S}\sigma(\boldsymbol{x};\boldsymbol{\theta}_i)/|S|$, i.e.,

$$\min_{S\subseteq[N]} \mathcal{L}[f_S] \quad s.t. \quad |S| \le n. \tag{1}$$

Here we allow the set S to contain repeated elements. This is a challenging combinatorial optimization problem. We propose a greedy forward selection strategy, which starts from an empty network and gradually adds the neuron that yields the best immediate decrease on loss. Specifically, starting from $S_0 = \emptyset$, we sequentially add neurons via

$$S_{n+1} \leftarrow S_n \cup i_n^* \quad \text{where} \quad i_n^* = \arg\min_{i\in[N]} \mathcal{L}[f_{S_n \cup i}]. \tag{2}$$

Notice that the constructed subnetwork inherits the weights of the large network and in practice we may further finetune the subnetwork with training data. More details of the practical algorithm and its extension to deep neural networks are in Section 4.

3. Theoretical Analysis

The simple greedy procedure yields strong theoretical guarantees, which, as a byproduct, also implies the existence of small and accurate subnetworks. Our results are two fold:

i) Under mild conditions, the selected subnetwork of size n achieves $\mathcal{L}[f_{S_n}] = \mathcal{O}(1/n) + \mathcal{L}_N^*$, where $\mathcal{L}_N^*$ is the best possible loss achievable by convex combinations of all the N neurons in $f_{[N]}$.

ii) We achieve a faster rate of $\mathcal{L}[f_{S_n}] = \mathcal{O}(1/n^2)$ if the large network $f_{[N]}$ is sufficiently over-parameterized and can overfit the training data subject to small perturbation (see Assumption 2).

In comparison, the mean field analysis of Song et al. (2018); Mei et al. (2019) shows that:

iii) Training a network of size n using (continuous time) gradient descent starting from random initialization gives an $\mathcal{O}(1/n + \varepsilon)$ loss, where ε is a (typically small) term involving the loss of infinitely wide networks trained with gradient dynamics. See Song et al. (2018); Mei et al. (2019) for details.

Our fast $\mathcal{O}(1/n^2)$ rate shows that subnetwork selection from *large, over-parameterized* models yields provably better results than training small networks of the same size starting from scratch using gradient descent. This provides the first theoretical justification of the empirical successes of the popular network pruning paradigm.

We now introduce the theory in depth. We start with the general $\mathcal{O}(1/n)$ rate in Section 3.1, and then establish and discuss the faster $\mathcal{O}(1/n^2)$ rate in Section 3.2 and 3.3.

3.1. General Convergence Rate

Let $\mathcal{L}_N^*$ be the minimal loss achieved by the best convex combination of all the N neurons in $f_{[N]}$, that is,

$$\mathcal{L}_N^* = \min_{\boldsymbol{\alpha}=[\alpha_1,\ldots,\alpha_N]} \left\{ \mathcal{L}[f_{\boldsymbol{\alpha}}] : \alpha_i \ge 0, \sum_{i=1}^N \alpha_i = 1 \right\}, \tag{3}$$

where $f_{\boldsymbol{\alpha}} = \sum_{i=1}^N \alpha_i \sigma(\boldsymbol{\theta}_i, \boldsymbol{x})$. It is obvious that $\mathcal{L}_N^* \le \mathcal{L}[f_{[N]}]$. We can establish the general $\mathcal{O}(1/n)$ rate with the following mild regularity conditions.

Assumption 1 (Boundedness and Smoothness). *Suppose that $||\boldsymbol{x}^{(i)}|| \le c_1$, $|y^{(i)}| \le c_1$ for every $i \in [m]$, and $\|\sigma_+\|_{Lip} \le c_1$, $\|\sigma_+\|_\infty \le c_1$ for some $c_1 < \infty$.*

Proposition 1. *Under Assumption 1, if S_n is constructed by (2), we have $\mathcal{L}[f_{S_n}] = \mathcal{O}(1/n) + \mathcal{L}_N^*$, for $\forall n \in [N]$.*

Remark Notice that at iteration n, the number of neurons in set S_n is no more than n since in each iteration, we at most increase the number of neurons by one. Also, as we allow select one neuron multiple times the number of neurons in S_n can be smaller than n.

Note that the condition of Proposition 1 is very mild. It holds for any original network $f_{[N]}$ of any size, although it is favorable to make N large to obtain a small $\mathcal{L}_N^*$. In the sequel, we show that a faster $\mathcal{O}(1/n^2)$ rate can be achieved, if $f_{[N]}$ is sufficiently large and can "overfit" the training data in a proper sense.

3.2. Faster Rate With Over-parameterized Networks

We now establish the faster rate $\mathcal{L}[f_{S_n}] = \mathcal{O}(1/n^2)$ when the large network is properly over-paramterized, which outperforms the $\mathcal{O}(1/n)$ rate achieved by standard gradient descent. This provides a theoretical foundation for the widely used approach of learning small networks by pruning from large networks.

Specifically, our result requires that N is sufficiently large and the neurons in $f_{[N]}$ are independent and diverse enough such that we can use a convex combination of N neurons to perfectly fit the data $\mathcal{D}_m$, even when subject to arbitrary perturbations on the labels with bounded magnitude.

Assumption 2 (**Over-parameterization**). *There exists a constant $\gamma > 0$ such that for any $\boldsymbol{\epsilon} = [\epsilon^{(1)}, ..., \epsilon^{(m)}] \in \mathbb{R}^m$ with $||\boldsymbol{\epsilon}|| \leq \gamma$, there exists $[\alpha_1, ..., \alpha_N] \in \mathbb{R}^N$ (which may depends on $\boldsymbol{\epsilon}$) with $\alpha_i \in [0, 1]$ and $\sum_{i=1}^N \alpha_i = 1$ such that for all $(\boldsymbol{x}^{(i)}, y^{(i)})$, $i \in [m]$,*

$$\sum_{j=1}^N \alpha_i \sigma(\boldsymbol{\theta}_j, \boldsymbol{x}^{(i)}) = y^{(i)} + \epsilon^{(i)}.$$

Note that this implies that $\mathcal{L}_N^* = 0$.

This roughly requires that the original large network should be sufficiently over-parametrized to have more independent neurons than data points to overfit arbitrarily perturbed labels (with a bounded magnitude). As we discuss in Appendix 9, Assumption 2 can be shown to be equivalent to the interior point condition of Frank-Wolfe algorithm (Bach et al., 2012; Lacoste-Julien, 2016; Chen et al., 2012).

Theorem 2 (**Faster Rate**). *Under assumption 1 and 2, for S_n defined in (2), we have*

$$\mathcal{L}[f_{S_n}] = \mathcal{O}(1/(\min(1, \gamma)n)^2). \qquad (4)$$

3.3. Assumption 2 Under Gradient Descent

In this subsection, we show that Assumption 2 holds with high probability when N is sufficiently large and the large network $f_{[N]}$ is trained using gradient descent with a proper random initialization. Our analysis builds on the mean field analysis of neural networks (Song et al., 2018; Mei et al., 2019). We introduce the background before we proceed.

Gradient Dynamics Assume the parameters $\{\boldsymbol{\theta}_i\}_{i=1}^N$ of $f_{[N]}$ are trained using a continuous-time gradient descent (which can be viewed as gradient descent with infinitesimal step size), with a random initialization:

$$\frac{d}{dt}\boldsymbol{\vartheta}_i(t) = \boldsymbol{g}_i(\boldsymbol{\vartheta}(t)), \quad \boldsymbol{\vartheta}_i(0) \overset{\text{i.i.d.}}{\sim} \rho_0, \quad \forall i \in [N], \qquad (5)$$

where $\boldsymbol{g}_i(\boldsymbol{\vartheta})$ denotes the negative gradient of loss w.r.t. $\boldsymbol{\vartheta}_i$,

$$\boldsymbol{g}_i(\boldsymbol{\vartheta}(t)) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_m}[(y - f(\boldsymbol{x};\ \boldsymbol{\vartheta}(t))\nabla_{\boldsymbol{\vartheta}_i}\sigma(\boldsymbol{x}, \boldsymbol{\vartheta}_i(t))],$$

and $f(\boldsymbol{x};\ \boldsymbol{\vartheta}) = \sum_{i=1}^N \sigma(\boldsymbol{x}, \boldsymbol{\vartheta}_i)/N$. Here we initialize $\boldsymbol{\vartheta}_i(0)$ by drawing i.i.d. samples from some distribution ρ_0.

Assumption 3. *Assume ρ_0 is an absolute continuous distribution on $\mathbb{R}^d$ with a bounded support. Assume the parameters $\{\boldsymbol{\theta}_i\}$ in $f_{[N]}$ are obtained by running (5) for some finite time T, that is, $\boldsymbol{\theta}_i = \boldsymbol{\vartheta}_i(T)$, $\forall i \in [N]$.*

Mean Field Limit We can represent a neural network using the empirical distribution of the parameters. Let ρ_t^N be the empirical measure of $\{\boldsymbol{\vartheta}_i(t)\}_{i=1}^N$ at time t, i.e., $\rho_t^N := \sum_{i=1}^N \delta_{\boldsymbol{\vartheta}_i(t)}/N$ where $\delta_{\boldsymbol{\vartheta}_i}$ is Dirac measure at $\boldsymbol{\vartheta}_i$. We can represent the network $f(\boldsymbol{x}; \boldsymbol{\vartheta}(t))$ by $f_{\rho_t^N} := \mathbb{E}_{\boldsymbol{\vartheta}\sim\rho_t^N}[\sigma(\boldsymbol{\vartheta}, \boldsymbol{x})]$. Also, $f_{[N]} = f_{\rho_T^N}$ under Assumption 3.

The mean field analysis amounts to study the limit behavior of the neural network with an infinite number of neurons. Specifically, as $N \to \infty$, it can be shown that ρ_t^N weakly converges to a limit distribution ρ_t^∞, and $f_{\rho_t^\infty}$ can be viewed as the network with infinite number of neurons at training time t. It is shown that ρ_t^∞ is characterized by a partial differential equation (PDE) (Song et al., 2018; Mei et al., 2019):

$$\frac{d}{dt}\rho_t^\infty = \nabla \cdot (\rho_t^\infty \boldsymbol{g}[\rho_t^\infty]), \qquad \rho_0^\infty = \rho_0, \qquad (6)$$

where $\boldsymbol{g}[\rho_t^\infty](\boldsymbol{\vartheta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_m}[(y - f_\rho(\boldsymbol{x}))\nabla_{\boldsymbol{\vartheta}}\sigma(\boldsymbol{x}, \boldsymbol{\vartheta})]$, $f_\rho(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\vartheta}\sim\rho}[\sigma(\boldsymbol{x};\ \boldsymbol{\vartheta})]$, and $\nabla \cdot \boldsymbol{g}$ is the divergence operator.

The mean field theory needs the following smoothness condition on activation to make sure the PDE (6) is well defined (Song et al., 2018; Mei et al., 2019).

Assumption 4. *The derivative of activation function is Lipschitz continuous, i.e., $||\sigma'_+||_{Lip} < \infty$.*

It is noticeable that Assumption 4 does not hold for ReLU. However, as shown in Song et al. (2018), empirically, ReLU networks behave very similarly to networks with smooth activation.

A key result of the mean field theory says that ρ_T^N weakly converges to ρ_T^∞ when $N \to \infty$. It implies that $\mathcal{L}[f_{\rho_T^n}] = \mathcal{O}(1/n + \varepsilon)$, with $\varepsilon = \mathcal{L}[f_{\rho_T^\infty}]$. As shown in Song et al. (2018), $\mathcal{L}[f_{\rho_T^\infty}]$ is usually a small term giving that the training time T is sufficiently large, under some regularity conditions.

Over-parameterization of Mean Field Limit The key idea of our analysis is: if the infinitely wide limit network $f_{\rho_T^\infty}$ can overfit any noise with bounded magnitude (as defined in Assumption 2), then Assumption 2 holds for $f_{\rho_T^N}$ with high probability when N is sufficiently large.

Assumption 5. *There exists $\gamma^* > 0$, such that for any noise vector $\boldsymbol{\epsilon} = [\epsilon_i]_{i=1}^m$ with $\|\boldsymbol{\epsilon}\| \leq \gamma^*$, there exists a positive integer M, and $[\alpha_1, ..., \alpha_M] \in \mathbb{R}^M$ with $\alpha_j \in [0, 1]$ and $\sum_{j=1}^M \alpha_j = 1$ and $\bar{\boldsymbol{\theta}}_j \in supp(\rho_T^\infty)$, $j \in [M]$ such that*

$$\sum_{j=1}^M \alpha_j \sigma(\bar{\boldsymbol{\theta}}_j, \boldsymbol{x}^{(i)}) = y^{(i)} + \epsilon^{(i)},$$

holds for any $i \in [m]$. Here M, $\{\alpha_j, \bar{\boldsymbol{\theta}}_j\}$ may depend on $\boldsymbol{\epsilon}$.

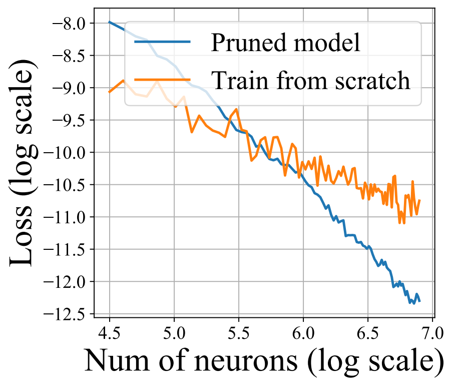

Figure 2. Comparison of loss of the pruned network and train-from-scratch network with varying sizes. Both the loss and number of neurons are in logarithm scale.

Assumption 5 can be viewed as an infinite variant of Assumption 2. It is very mild because supp(ρ_T^∞) contains infinitely many neurons and given any $\boldsymbol{\epsilon}$, we can pick an arbitrarily large number of neurons from supp(ρ_T^∞) and reweight them to fit the perturbed data. Also, assumption 5 implicitly requires a sufficient training time T in order to make the limit network fit the data well.

Assumption 6 (**Density Regularity**). *For* $\forall r_0 \in (0, \gamma^*]$, *there exists* p_0 *that depends on* r_0, *such that for every* $\boldsymbol{\theta} \in$ *supp*(ρ_T^∞), *we have* $\mathbb{P}_{\boldsymbol{\theta}' \sim \rho_T^\infty}\left(\left\|\boldsymbol{\theta}' - \boldsymbol{\theta}\right\| \le r_0\right) \ge p_0$.

Theorem 3. *Suppose Assumption 1, 3, 4, 5 and 6 hold, then for any* $\delta > 0$, *when* N *is sufficiently large, assumption 2 holds for any* $\gamma \le \frac{1}{2}\gamma^*$ *with probability at least* $1-\delta$, *which gives that* $\mathcal{L}[f_{S_n}] = \mathcal{O}(1/(\min(1,\gamma)n)^2)$.

Theorem 3 shows that if the pre-trained network is sufficiently large, the loss of the pruned network decays at a faster rate. Compared with Proposition 1, it highlights the importance of using a large pre-trained network for pruning.

Pruning vs. GD: Numerical Verification of the Rates We numerically verify the fast $\mathcal{O}(1/n^2)$ rate in (4) and the $\mathcal{O}(1/n)$ rate of gradient descent by Song et al. (2018); Mei et al. (2019) (when ε term is very small). Given some simulated data, we first train a large network $f_{[N]}$ with $N = 1000$ neurons by gradient descent with random initialization. We then apply our greedy selection algorithm to find subnetworks with different sizes n. We also directly train networks of size n with gradient descent. See Appendix 7 for more details. Figure 2 plots the the loss $\mathcal{L}[f]$ and the number of neurons n of the pruned network and the network trained from scratch. This empirical result matches our $\mathcal{O}(1/n^2)$ rate in Theorem 3, and the $\mathcal{O}(1/n)$ rate of the gradient descent.

3.4. Pruning Randomly Weighted Networks

A line of recent empirical works (e.g., Frankle & Carbin, 2019; Ramanujan et al., 2019) shows that it is possible to find a subnetwork with good accuracy inside a large network with random weights without pretraining. Our analysis is also applicable to this case. Specifically, the $\mathcal{L}[f_{S_n}] = \mathcal{O}(1/n^2)$ bound in Theorem 3 holds even when the weights $\{\boldsymbol{\theta}_i\}$ of the large network is i.i.d. drawn from the initial distribution ρ_0, without further training. This is because Theorem 3 applies to any training time T, including $T = 0$ (no training). See Appendix 10 for a more thorough discussion.

3.5. Greedy Backward Elimination

To better illustrate the advantages of the forward selection approach over backward elimination (see Figure 1), it is useful to consider the backward elimination counterpart of our method which minimizes the same loss as our method, but from the opposite direction. That is, it starts from the full network $S_0^{\text{B}} := [N]$, and sequentially deletes neurons via

$$S_{n+1}^{\text{B}} \leftarrow S_n^{\text{B}} \setminus \{i_n^*\}, \quad \text{where } i_n^* = \underset{i \in S_n^{\text{B}}}{\arg\min} \mathcal{L}[f_{S_n^{\text{B}} \setminus \{i\}}].$$

As shown in Appendix 11, this backward elimination does not enjoy similar $\mathcal{O}(1/n)$ or $\mathcal{O}(1/n^2)$ rates as forward selection and simple counter examples can be constructed easily. Additionally, Table 5 in Appendix 11 shows that the forward selection outperforms this backward elimination on both ResNet34 and MobileNetV2 for ImageNet.

3.6. Further Discussion

To the best of our knowledge, our work provides the first rigorous theoretical justification that pruning from over-parameterized models *outperforms* direct training of small networks from scratch using gradient descent, under rather practical assumptions. However, there still remain gaps between theory and practice that deserve further investigation in future works. Firstly, we only analyze the simple two-layer networks but we believe our theory can be generalized to deep networks with refined analysis and more complex theoretical framework such as deep mean field theory (Araújo et al., 2019; Nguyen & Pham, 2020). We conjecture that pruning deep network gives $\mathcal{O}(1/n^2)$ rate with the constant depending on the Lipschitz constant of the mapping from feature map to output. Secondly, as we only analyze the two-layer networks, our theory cannot characterize whether pruning finds good structure of deep networks, as discussed in Liu et al. (2019b). Indeed, theoretical works on how network architecture influences the performance are still largely missing. Finally, some of our analysis is built on the mean field theory, which is a special parameterization of network. It is also of interest to generalize our

Algorithm 1 Layer-wise Greedy Subnetwork Selection

```
**Goal**: Given a pretrained network $f_{\text{Large}}$ with $H$ layers,
find a subnetwork $f$ with high accuracy.
Set $f = f_{\text{Large}}$.
**for** Layer $h \in [H]$ (From input layer to output layer) **do**
  Set $S = \emptyset$
  **while** Convergence criterion is not met **do**
    Randomly sample a mini-batch data $\hat{\mathcal{D}}$
    **for** filter (or neuron) $k \in [N_h]$ **do**
      $S'_k \leftarrow S \cup \{k\}$
      Replace layer $h$ of $f$ by $\sum_{j \in [S'_k]} \sigma(\boldsymbol{\theta}_j, \boldsymbol{z}^{\text{in}}) / |S'_k|$
      Calculate its loss $\ell_k$ on mini-batch data $\hat{\mathcal{D}}$.
    **end for**
    $S \leftarrow S \cup \{k^*\}$, where $k^* = \arg\min_{k \in [N_h]} \ell_k$
  **end while**
  Replace layer $h$ of $f$ by $\sum_{j \in [S]} \sigma(\boldsymbol{\theta}_j, \boldsymbol{z}^{\text{in}}) / |S|$
**end for**
Finetune the subnetwork $f$.
```

theory to other parameterizations, such as these based on neural tangent kernel (Jacot et al., 2018; Du et al., 2019b).

4. Practical Algorithm and Experiments

Practical Algorithm We propose to apply the greedy selection strategy in a layer-wise fashion in order to prune neural networks with multiple layers. Assume we have a pretrained deep neural network with H layers, whose h-th layer contains N_h neurons and defines a mapping $\sum_{j \in [N_h]} \sigma(\boldsymbol{\theta}_j, \boldsymbol{z}^{\text{in}})/N_h$, where $\boldsymbol{z}^{\text{in}}$ denotes the input of this layer. To extend the greedy subnetwork selection to deep networks, we propose to prune the layers sequentially, from the input layer to the output layer. For each layer, we first remove all the neurons in that layer, and gradually add the best neuron back that yields the largest decrease of the loss, similar to the updates in (2). After finding the subnetwork for all the layers, we further finetune the pruned network, training it with stochastic gradient descent using the weight of original network as initialization. This allows us to inherit the accuracy and information in the pruned subnetwork, because finetuning can only decrease the loss over the initialization. We summarize the detailed procedure of our method in Algorithm 1.

Empirical Results We first apply the proposed algorithm to prune various models, e.g. ResNet (He et al., 2016), MobileNetV2 (Sandler et al., 2018), MobileNetV3 (Howard et al., 2019) and ProxylessNet (Cai et al., 2019) for ImageNet (Deng et al., 2009) classification. We also show the experimental results on CIFAR-10/100 in the appendix. Code is available at `https://github.com/lushleaf/Network-Pruning-Greedy-Forward-Selection`. Our results are summarized as follows:

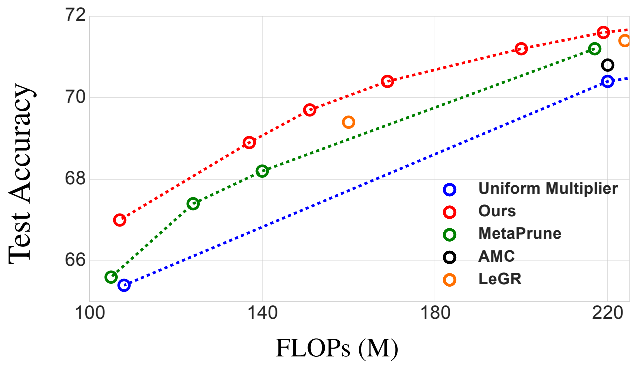

Figure 3. After applying different pruning algorithms to MobileNetV2 on ImageNet, we display the top1 accuracy of different methods. It is obvious that our algorithm can consistently outperform all the others under any FLOPs.

i) Our greedy selection method consistently outperforms the prior arts on network pruning on learning small and accurate networks with high computational efficiency.

ii) Finetuning pruned subnetworks of neural architectures (e.g., MobileNetV2/V3) consistently outperforms re-training them from new random initialization, violating the results of Liu et al. (2019b).

iii) Increasing the size of the pre-trained large networks improves the performance of the pruned subnetworks, highlighting the importance of pruning from *large* models.

4.1. Finding Subnetworks on ImageNet

We use ILSVRC2012, a subset of ImageNet (Deng et al., 2009) which consists of about 1.28 million training images and 50,000 validation images with 1,000 different classes.

Training Details We evaluate each neuron using a mini-batch of training data to select the next one to add, as shown in Algorithm 1. We stop adding new neurons when the gap between the current loss and the loss of the original pre-trained model is smaller than ϵ. We vary ϵ to get pruned models with different sizes.

During finetuning, we use the standard SGD optimizer with Nesterov momentum 0.9 and weight decay 5×10^{-5}. For ResNet, we use a fixed learning rate 2.5×10^{-4}. For the other architectures, following the original settings (Cai et al., 2019; Sandler et al., 2018), we decay learning rate using cosine schedule (Loshchilov & Hutter, 2017) starting from 0.01. We finetune subnetwork for 150 epochs with batch size 512 on 4 GPUs. We resize images to 224×224 resolution and adopt the standard data augmen-

Model	Method	Top-1 Acc	Size (M)	FLOPS
ResNet34	Full Model (He et al., 2016)	73.4	21.8	3.68G
	Li et al. (2017)	72.1	-	2.79G
	Liu et al. (2019b)	72.9	-	2.79G
	Dong et al. (2017)	73.0	-	2.75G
	Ours	**73.5**	**17.2**	**2.64G**
	SFP (He et al., 2018a)	71.8	-	2.17G
	FPGM (He et al., 2019)	72.5	-	2.16G
	Ours	**72.9**	**14.7**	**2.07G**
MobileNetV2	Full Model (Sandler et al., 2018)	72.0	3.5	314M
	Ours	**71.9**	**3.2**	**258M**
	LeGR (Chin et al., 2019)	71.4	-	224M
	Uniform (Sandler et al., 2018)	70.4	2.9	220M
	AMC (He et al., 2018b)	70.8	2.9	220M
	Ours	**71.6**	**2.9**	**220M**
	Meta Pruning (Liu et al., 2019a)	71.2	-	217M
	Ours	**71.2**	**2.7**	**201M**
	ThiNet (Luo et al., 2017)	68.6	-	175M
	DPL (Zhuang et al., 2018)	68.9	-	175M
	Ours	**70.4**	**2.3**	**170M**
	LeGR (Chin et al., 2019)	69.4	-	160M
	Ours	**69.7**	**2.2**	**152M**
	Meta Pruning (Liu et al., 2019a)	68.2	-	140M
	Ours	**68.8**	**2.0**	**138M**
	Uniform (Sandler et al., 2018)	65.4	-	106M
	Meta Pruning (Liu et al., 2019a)	65.0	-	105M
	Ours	**66.9**	**1.9**	**107M**
MobileNetV3-Small	Full Model (Howard et al., 2019)	67.5	2.5	64M
	Uniform (Howard et al., 2019)	65.4	2.0	47M
	Ours	**65.8**	2.0	**49M**
ProxylessNet-Mobile	Full Model (Cai et al., 2019)	74.6	4.1	324M
	Uniform (Cai et al., 2019)	72.9	3.6	240M
	Ours	**74.0**	3.4	**232M**

Table 1. Top1 accuracies for different benchmark models, e.g. ResNets (He et al., 2016), MobileNetV2 (Sandler et al., 2018), MobileNetV3-small (Howard et al., 2019) and ProxylessNet (Cai et al., 2019) on ImageNet2012 (Deng et al., 2009).

tation scheme (mirroring and shifting).

Results Table 1 reports the top1 accuracy, FLOPs and model size [1] of subnetworks pruned from the full networks. We first test our algorithm on two standard benchmark models, ResNet-34 and MobileNetV2. We further apply our algorithm to several recent proposed models e.g., ProxylessNet, MobileNetV3-Small.

ResNet-34 Our algorithm outperforms all the prior results on ResNet-34. We obtain an even better top1 accuracy (73.4% vs. 73.5%) than the full-size network while reducing the FLOPs from 3.68G to 2.64G. We also obtain a model with 72.9% top1 accuracy and 2.07G FLOPs, which has higher accuracy but lower FLOPs than previous works.

MobileNetV2 Different from ResNet and other standard structures, MobileNetV2 on ImageNet is known to be hard to prune using most traditional pruning algorithms (Chin et al., 2019). As shown in Table 1, compared with the 'uniform baseline', which uniformly reduces the number of channels on each layer, most popular algorithms fail to improve the performance by a large margin. In comparison, our algorithm improves the performance of small-size networks by a significant margin. As shown in Table 1, the subnetwork with 245M FLOPs obtains 71.9% top1 accuracy, which matches closely with the 72.0% accuracy of the full-size network. Our subnetwork with 151M FLOPs achieves 69.7% top1 accuracy, improving the previous state-of-the-art of 69.4% top1 accuracy with 160M FLOPs. As shown in Figure 3, our algorithm consistently

[1] All the FLOPS and model size reported in this paper is calculated by `https://pypi.org/project/ptflops`.

outperforms all the other baselines under all FLOPs. The improvement of our method on the low FLOPs region is particularly significantly. For example, when limited to 106M FLOPs, we improve the 65.0% top1 accuracy of Meta Pruning to 66.9%.

ProxylessNet-Mobile and MobileNetV3-Small We further experiment on two recently-proposed architectures, ProxylessNet-Mobile and MobileNetV3-Small. As shown in Table 1, we consistently outperform the 'uniform baseline'. For MobileNetV3-Small, we improve the 65.4% top1 accuracy to 65.8% when the FLOPs is less than 50M FLOPs. For ProxylessNet-Mobile, we enhance the 72.9% top1 accuracy to 74.0% when the FLOPs is under 240M.

4.2. Rethinking the Value of Finetuning

Recently, Liu et al. (2019b) finds that for ResNet, VGG and other standard structures on ImageNet, re-training the weights of the pruned structure from new random initialization can achieve better performance than finetuning. However, we find that this claim does not hold for mobile models, such as MobileNetV2 and MobileNetV3. In our experiments, we use the same setting of Liu et al. (2019b) for re-training from random initialization.

Models	FLOPs	Re-training (%)	Finetune (%)
MobileNetV2	220M	70.8	71.6
MobileNetV2	170M	69.0	70.4
MobileNetV3	49M	63.2	65.8

Table 2. Top1 accuracy on MobileNetV2 and MobileNetV3-small on ImageNet. "Scratch" denotes training the pruned model from scratch. We use the Scratch-B setting in Liu et al. (2019b) for training from scratch.

We compare finetuning and re-training of the pruned MobileNetV2 with 219M and 169M Flops. As shown in Table 2, finetuning outperforms re-training by a large margin. For example, for the 169M FLOPs model, re-training decreases the top1 accuracy from 70.4% to 69.0%. This empirical evidence demonstrates the importance of using the weights learned by the large model to initialize the pruned model.

We conjecture that the difference between our findings and Liu et al. (2019b) might come from several reasons. Firstly, for large architecture such as VGG and ResNet, the pruned model is still large enough (e.g. as shown in Table 1, FLOPs > 2G) to be optimized from scratch. However, this does not hold for the pruned mobile models, which is much smaller. Secondly, Liu et al. (2019b) mainly focus on sparse regularization based pruning methods such as Han et al. (2015); Li et al. (2017); He et al. (2017). In those methods, the loss used for training the large network has an extra strong regularization term, e.g., channel-wise L_p penalty. However, when re-training the pruned small network, the penalty is excluded. This gives inconsistent loss functions. As a consequence, the weights of the large pre-trained network may not be suitable for finetuning the pruned model. In comparison, our method uses the same loss for training the re-trained large model and the pruned small network, both without regularization term. A more comprehensive understanding of this issue is valuable to the community, which we leave as a future work.

However, we believe that a more comprehensive understanding of finetuning is valuable to the community, which we leave as a future work.

	Large N $\longrightarrow$ Small N		
Original FLOPs (M)	320	220	108
Pruned FLOPs (M)	96	96	97
Top1 Accuracy (%)	66.2	65.6	64.9

Table 3. We apply our algorithm to get three pruned models with similar FLOPs from full-size MobileNetV2, MobileNetV2$\times$0.75 and MobileNetV2$\times$0.5. We compare their top1 accuracy on Imagenet.

4.3. On the Value of Pruning from Large Networks

Our theory suggests it is better to prune from a larger model, as discussed in Section 3. To verify, we apply our method to MobileNetV2 with different sizes, including MobileNetV2 (full size), MobileNetV2$\times$0.75 and MobileNetV2$\times$0.5 (Sandler et al., 2018). We keep the FLOPs of the pruned models almost the same and compare their performance. As shown in Table 3, the pruned models from larger original models give better performance. For example, the 96M FLOPs pruned model from the full-size MobileNetV2 obtains a top1 accuracy of 66.2% while the one pruned from MobileNetV2$\times$0.5 only has 64.9%.

5. Related Works

Structured Pruning A vast literature exists on *structured* pruning (e.g., Han et al., 2016a), which prunes neurons, channels or other units of neural networks. Compared with weight pruning (e.g., Han et al., 2016b), which specifies the connectivity of neural networks, structured pruning is more realistic as it can compress neural networks without dedicated hardware or libraries. Existing methods prune the redundant neurons based on different criterion, including the norm of the weights (e.g., Liu et al., 2017; Zhuang et al., 2018; Li et al., 2017), feature reconstruction error of the next or final layers (e.g., He et al., 2017; Yu et al., 2018; Luo et al., 2017), or gradient-based sensitivity measures (e.g., Baykal et al., 2019b; Zhuang et al., 2018). Our method is designed to directly minimize the final loss, and yields both better practical performance and theoretical guarantees.

Forward Selection vs. Backward Elimination Many of the popular conventional network pruning methods are based on backward elimination of redundant neurons (e.g. Liu et al., 2017; Li et al., 2017; Yu et al., 2018), and fewer algorithms are based forward selection like our method (e.g. Zhuang et al., 2018). Among the few exceptions, Zhuang et al. (2018) propose a greedy channel selection algorithm similar to ours, but their method is based on minimizing a gradient-norm based sensitivity measure (instead of the actual loss like us), and yield no theoretical guarantees. Appendix 11 discusses the theoretical and empirical advantages of forward selection over backward elimination.

Sampling-based Pruning Recently, a number of works (Baykal et al., 2019a; Liebenwein et al., 2019; Baykal et al., 2019b; Mussay et al., 2020) proposed to prune networks based on variants of (iterative) random sampling according to certain sensitivity score. These methods can provide concentration bounds on the difference of output between the pruned networks and the full networks, which may yield a bound of $\mathcal{O}(1/n + \mathcal{L}[f_{[N]}])$ with a simple derivation. Our method uses a simpler greedy deterministic selection strategy and achieves better rate than random sampling in the overparameterized cases In contrast, sampling-based pruning may not yield the fast $\mathcal{O}(1/n^2)$ rate even with overparameterized models. Unlike our method, these works do not justify the advantage of pruning from large models over direct gradient training.

Lottery Ticket; Re-train After Pruning Frankle & Carbin (2019) proposed the Lottery Ticket Hypothesis, claiming the existence of winning subnetworks inside large models. Liu et al. (2019b) regards pruning as a kind for neural architecture search. A key difference between our work and Frankle & Carbin (2019) and Liu et al. (2019b) is how the parameters of the subnetwork are trained:

i) We finetune the parameters of the subnetworks starting from the weights of the pre-trained large model, hence inheriting the information the large model.

ii) Liu et al. (2019b) proposes to re-train the parameters of the pruned subnetwork starting from new random initialization.

iii) Frankle & Carbin (2019) proposes to re-train the pruned subnetwork starting from the same initialization and random seed used to train the pre-trained model.

Obviously, the different parameter training of subnetworks should be combined with different network pruning strategies to achieve the best results. Our algorithmic and theoretical framework naturally justifies the finetuning approach. Different theoretical frameworks for justifying the proposals of Liu et al. (2019b) and Frankle & Carbin (2019) (equipped with their corresponding subnetwork selection methods) are of great interest.

More recently, a concurrent work Malach et al. (2020) discussed a stronger form of lottery ticket hypothesis that shows the existence of winning subnetworks in large networks with random weights (without pre-training), which corroborates the empirical observations in (Wang et al., 2019; Ramanujan et al., 2019). However, the result of Malach et al. (2020) does not yield fast rate as our framework for justifying the advantage of network pruning over training from scratch, and does not motivate practical algorithms for finding good subnetworks in practice.

Frank-Wolfe Algorithm As suggested in Bach (2017), Frank-Wolfe (Frank & Wolfe, 1956) can be applied to learn neural networks, which yields an algorithm that greedily adds neurons to progressively construct a network. However, each step of Frank-Wolfe leads to a challenging global optimization problem, which can not be solved in practice.

Compared with Bach (2017), our subnetwork selection approach can be viewed as constraining the global optimization the discretized search space constructed using over-parameterized large networks pre-trained using gradient descent. Because gradient descent on over-parameterized networks is shown to be nearly optimal (e.g., Song et al., 2018; Mei et al., 2019; Du et al., 2019b;a; Jacot et al., 2018), selecting neurons inside the pre-trained models can provide a good approximation to the original non-convex problem.

Sub-modular Optimization An alternative general framework for analyzing greedy selection algorithms is based on sub-modular optimization (Nemhauser et al., 1978). However, our problem 1 is not sub-modular, and the (weak) sub-modular analysis (Das & Kempe, 2011) can only bound the ratio between $\mathcal{L}[f_{S_n}]$ and the best loss of subnetworks of size n achieved by (1), not the best loss $\mathcal{L}^*_N$ achieved by the best convex combination of all the N neurons in the large model.

6. Conclusion

We propose a simple and efficient greedy selection algorithm for constructing subnetworks from pretrained large networks. Our theory provably justifies the advantage of pruning from large models over training small networks from scratch. The importance of using sufficiently large, over-parameterized models and finetuning (instead of re-training) the selected subnetworks are emphasized. Empirically, our experiments verify our theory and show that our method improves the prior arts on pruning various models such as ResNet-34 and MobileNetV2 on Imagenet.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

Araújo, D., Oliveira, R. I., and Yukimura, D. A mean-field limit for certain deep neural networks. *arXiv preprint arXiv:1906.00193*, 2019.

Bach, F. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Bach, F., Lacoste-Julien, S., and Obozinski, G. On the equivalence between herding and conditional gradient algorithms. *International Conference on Machine*, 2012.

Baykal, C., Liebenwein, L., Gilitschenski, I., Feldman, D., and Rus, D. Data-dependent coresets for compressing neural networks with applications to generalization bounds. *The International Conference on Learning Representations*, 2019a.

Baykal, C., Liebenwein, L., Gilitschenski, I., Feldman, D., and Rus, D. Sipping neural networks: Sensitivity-informed provable pruning of neural networks. *arXiv preprint arXiv:1910.05422*, 2019b.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *The International Conference on Learning Representations*, 2019.

Chaudhuri, K. and Dasgupta, S. Rates of convergence for the cluster tree. In *Advances in Neural Information Processing Systems*, pp. 343–351, 2010.

Chen, Y., Welling, M., and Smola, A. Super-samples from kernel herding. *arXiv preprint arXiv:1203.3472*, 2012.

Chin, T.-W., Ding, R., Zhang, C., and Marculescu, D. Legr: Filter pruning via learned global ranking. *arXiv preprint arXiv:1904.12368*, 2019.

Das, A. and Kempe, D. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 1057–1064, 2011.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. IEEE, 2009.

Dong, X., Huang, J., Yang, Y., and Yan, S. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5840–5848, 2017.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. *International Conference on Machine*, 2019a.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *International Conference of Learning Representations*, 2019b.

Dudley, R. M. Balls in rk do not cut all subsets of k+ 2 points. *Advances in Mathematics*, 31(3):306–308, 1979.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2): 95–110, 1956.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *The International Conference on Learning Representations*, 2019.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., and Dally, W. J. Eie: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016a.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *The International Conference on Learning Representations*, 2016b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. *International Joint Conference on Artificial Intelligence*, 2018a.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European*

Conference on Computer Vision (ECCV), pp. 784–800, 2018b.

He, Y., Liu, P., Wang, Z., Hu, Z., and Yang, Y. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314–1324, 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Lacoste-Julien, S. Convergence rate of frank-wolfe for non-convex objectives. *NIPS*, 2016.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *The International Conference on Learning Representations*, 2017.

Liebenwein, L., Baykal, C., Lang, H., Feldman, D., and Rus, D. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.-T., and Sun, J. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3296–3305, 2019a.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. *The International Conference on Learning Representations*, 2019b.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *The International Conference on Learning Representations*, 2017.

Luo, J.-H., Wu, J., and Lin, W. 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.

Malach, E., Yehudai, G., Shalev-Shwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need, 2020.

Mei, S., Misiakiewicz, T., and Montanari, A. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv preprint arXiv:1902.06015*, 2019.

Mussay, B., Osadchy, M., Braverman, V., Zhou, S., and Feldman, D. Data-independent neural pruning via coresets. In *The International Conference on Learning Representations*, 2020.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functionsi. *Mathematical programming*, 14(1):265–294, 1978.

Nguyen, P.-M. and Pham, H. T. A rigorous framework for the mean field limit of multilayer neural networks, 2020.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? *arXiv preprint arXiv:1911.13299*, 2019.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Song, M., Montanari, A., and Nguyen, P. A mean field view of the landscape of two-layers neural networks. *Proceedings of the National Academy of Sciences*, 115:E7665–E7671, 2018.

Wang, Y., Zhang, X., Xie, L., Zhou, J., Su, H., Zhang, B., and Hu, X. Pruning from scratch. *arXiv preprint arXiv:1909.12579*, 2019.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018.

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., and Zhu, J. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 875–886, 2018.