Lottery Ticket Preserves Weight Correlation: Is it Desirable or Not?

Ning Liu [1*] Geng Yuan [2*] Zhengping Che [3] Xuan Shen [2] Xiaolong Ma [2] Qing Jin [2] Jian Ren [4] Jian Tang [1†] Sijia Liu [5] Yanzhi Wang [2†]

Abstract

In deep model compression, the recent finding "Lottery Ticket Hypothesis" (LTH) (Frankle & Carbin, 2018) pointed out that there could exist a winning ticket (i.e., a properly pruned subnetwork together with original weight initialization) that can achieve competitive performance than the original dense network. However, it is not easy to observe such winning property in many scenarios, where for example, a relatively large learning rate is used even if it benefits training the original dense model. In this work, we investigate the underlying condition and rationale behind the winning property, and find that the underlying reason is largely attributed to the correlation between initialized weights and final-trained weights when the learning rate is not sufficiently large. Thus, the existence of winning property is correlated with an insufficient DNN pretraining, and is unlikely to occur for a well-trained DNN. To overcome this limitation, we propose the "pruning & fine-tuning" method that consistently outperforms lottery ticket sparse training under the same pruning algorithm and the same total training epochs. Extensive experiments over multiple deep models (VGG, ResNet, MobileNet-v2) on different datasets have been conducted to justify our proposals.

1. Introduction

Weight pruning has been widely studied and utilized to effectively remove the redundant weights in the over-parameterized deep neural networks (DNNs) while maintaining the accuracy performance (Han et al., 2015a; 2016; Wen et al., 2016; He et al., 2017; Min et al., 2018; He et al., 2019; Dai et al., 2019; Lin et al., 2020; He et al., 2020). The typical *pruning pipeline* has three main stages: 1) train an over-parameterized DNN, 2) prune the unimportant weights in the original DNN, and 3) fine-tune the pruned DNN to restore accuracy.

Many works have been proposed to investigate the behaviors on weight pruning (Tanaka et al., 2020; Ye et al., 2020; Renda et al., 2020; Malach et al., 2020). The Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2018) reveals that, inside a dense network with randomly initialized weights, a small sparse subnetwork, when trained in isolation using the identical initial weights, can reach a similar accuracy as the dense network. Such a sparse subnetwork with the initial weights is called the winning ticket.

For a more rigorous definition, let $f(x;\theta_0)$ be a given network *initialization*, where $\theta_0 \sim D_\theta$ denotes the initial weights. We then formalize pre-training, pruning and sparse training below. *Pre-training:* The network is trained for T epochs arriving at weights θ_T and network function $f(x;\theta_T)$. *Pruning:* Based on the pretrained weights θ_T, adopt certain pruning algorithm to generate a sparse mask $m \in \{0,1\}^{|\theta|}$. *Sparse Training:* The LTH paper considers two cases of sparse training. The first ("winning ticket") is the direct application of mask m to initial weights θ_0, resulting in weights $\theta_0 \odot m$ and network function $f(x;\theta_0 \odot m)$. The second (random reinitialization) is the application of mask m to a random initialization of weights $\theta'_0 \sim D_\theta$, resulting in weights $\theta'_0 \odot m$ and network function $f(x;\theta'_0 \odot m)$. The *winning property* has **two** aspects ①-② for identification: ① Training $f(x;\theta_0 \odot m)$ for T epochs (or fewer) will result in similar accuracy as that of the dense pre-trained network $f(x;\theta_T)$. ② There should be a notable accuracy gap between training $f(x;\theta_0 \odot m)$ for T epochs and training $f(x;\theta'_0 \odot m)$, and the former shall be higher.

In the standard LTH setup (Frankle & Carbin, 2018), the winning property can be observed in the case of low learning rate via the simple iterative magnitude pruning algorithm, but fails to occur at higher initial learning rates especially in deeper neural networks. For instance, the LTH work identifies the winning tickets on the CIFAR-10 dataset for the CONV-2/4/6 architectures (the down-scaled variants of VGG (Simonyan & Zisserman, 2014)), with the initial learning rate as low as 0.0001. For deeper networks such as

[*]Equal contribution. [1]Midea Group, Beijing, China [2]Northeastern University, Boston, MA, USA [3]Didi Chuxing, Beijing, China [4]Snap Inc., CA, USA [5]Michigan State University, MI, USA. [†]Corresponding author.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

ResNet-20 and VGG-19 on CIFAR-10, the winning tickets can be identified only in the case of low learning rate. At higher learning rates, additional warm up is needed to find the winning tickets. In Liu et al. (2018) (the latest ArXiv version), it revisits LTH and finds out that with a widely-adopted learning rate, the winning ticket has no accuracy advantage over the random reinitialization. This questions the second aspect of winning property on the accuracy gap between training $f(x; \theta_0 \odot m)$ and training $f(x; \theta_0' \odot m)$.

Further, the following work Frankle et al. (2019) proposes the iterative pruning with rewinding to stabilize identifying the winning tickets. Specifically, it resets the weights to θ_k in each pruning iteration, where θ_k denotes the weights trained from θ_0 for a small number of k epochs.

In this paper, we investigate the underlying condition and rationale behind winning property. We ask if such a property is a natural characteristic of DNNs across their architectures and/or applications. We revisit LTH via extensive experiments built upon various representative DNN models and datasets, and have confirmed that the winning property only exists at a low learning rate. In fact, such a "low learning rate" (e.g., 0.01 for ResNet-20 and 0.0001 for CONV-2/4/6 architectures on CIFAR-10) is already significantly deviated from the standard learning rate, and results in notable accuracy degradation in the pretrained DNN. Besides, training from the "winning ticket" at such a low learning rate can only restore the accuracy of the pretrained DNN under the same insufficient learning rate, instead of that under the desirable learning rate. By introducing a *correlation indicator* for quantitative analysis, we found that the underlying reason is largely attributed to the correlation between initialized weights and final-trained weights when the learning rate is not sufficiently large. We draw the following conclusions:

- As a result of low learning rate, such weight correlation results in low accuracy in DNN pretraining.
- Such weight correlation is also a key condition of winning property, concluded through a detailed analysis of the cause of winning property.
- Thus, the existence of winning property is correlated with an insufficient DNN pretraining, i.e., it is unlikely to occur for a well-trained DNN.

Different from sparse training under lottery ticket setting, we propose the "pruning & fine-tuning" method, i.e., apply mask m to pre-trained weights θ_T and perform fine-tuning for T epochs. The generated sparse subnetwork can largely achieve the accuracy of the pretrained dense DNN. Through comprehensive experiments and analysis we draw the following conclusions:

- "Pruning & fine-tuning" consistently outperforms lottery ticket setting under the same pruning algorithm for mask generation, and the same total training epochs.
- The pruning algorithm responsible for mask generation plays an important role in the quality of generated sparse subnetwork.
- Thus, if one wants to optimize the accuracy of sparse subnetwork and restore the accuracy of the pretrained dense DNN, we suggest adopting the pruning & fine-tuning method instead of lottery ticket setting.

2. Related Work

2.1. DNN Weight Pruning

DNN weight pruning as a model compression technique can effectively remove the redundant weights in DNN models and hence reduce both storage and computation costs. The general flow of weight pruning consists of three steps: (1) train the neural network first; (2) derive a sub-network structure (i.e., removing unimportant weights) using a certain pruning algorithm; and (3) fine-tune the remaining weights in the sub-network to restore accuracy. Different pruning algorithms will deliver different capabilities to search for the best-suited sparse sub-network and lead to different final accuracies.

The most straightforward method is the magnitude-based *one-shot pruning*, which will directly zero-out a given percentage of trained weights with the smallest magnitude. However, this method usually leads to a severe accuracy drop under a relatively high pruning rate. *Iterative magnitude pruning* is proposed in (Han et al., 2015b), which removes the weights with the smallest magnitude in an iterative manner. It repeats step (1) and step (2) multiple times until reaching the target pruning rate. In (Frankle & Carbin, 2018), iterative pruning is adopted to find the sparse sub-network (i.e., winning ticket). The iterative pruning process is still a greedy search, and has been extended in (Zhu & Gupta, 2017; Tan & Motani, 2020; Liu et al., 2020) to derive better sub-network structures.

To overcome the greedy nature in the heuristic pruning methods, the more mathematics-oriented regularization-based algorithm (Wen et al., 2016; He et al., 2017) has been proposed, to generate sparsity by incorporating ℓ_1 or ℓ_2 structured regularization in the loss function. However, this method directly applies fixed regularization terms that penalize all weights equally and will lead to a potential accuracy drop. Later work (Zhang et al., 2018; Ren et al., 2019) incorporate Alternating Direction Methods of Multipliers (ADMM) (Boyd et al., 2011; Ouyang et al., 2013) to solve the pruning problem as an optimization problem, which adopts dynamic regularization penalties and maintains high accuracy.

2.2. Lottery Ticket Hypothesis

2.2.1. The Origin and Controversy of Lottery Ticket Hypothesis

The recent work (Frankle & Carbin, 2018) reveals that, inside a dense network with randomly initialized weights, a small sparse subnetwork can reach a similar test accuracy when trained in isolation using the identical initial weights as training the dense network. Such sparse subnetwork is called the winning ticket and can be found by pruning the pre-trained dense network under a non-trivial pruning ratio.

As demonstrated in (Frankle & Carbin, 2018), winning tickets can be found in small networks and small dataset when using relatively low learning rates (e.g., 0.01 for SGD). The work from the same period (Liu et al., 2018) finds that, when using a relatively large learning rate (e.g., 0.1 for SGD), training a "winning ticket" with identical initialized weights will not provide any unique advantage in accuracy compared to training with randomly initialized weights. The following work (Frankle et al., 2019; Renda et al., 2020) also confirms that, for deeper networks and using relatively large learning rates, the winning property can hardly be observed. They propose a weight rewinding technique to identify small subnetworks, which can be trained in isolation to achieve competitive accuracy as the dense pretrained network.

2.2.2. Other Aspects and Applications

Later work (Chen et al., 2020) further extends the lottery ticket hypothesis to a pre-trained BERT model to evaluate the transferability of the sparse subnetworks among different downstream NLP tasks. Recent works (Morcos et al., 2019; Chen et al., 2020) have studied the lottery ticket hypothesis in computer vision tasks and in unsupervised learning.

The potential of sparse training suggested by the lottery ticket hypothesis has motivated the study of deriving the "winning tickets" at an early stage of training, thereby accelerating training process. There is a number of work in this direction (Frankle et al., 2020a; You et al., 2019; Frankle et al., 2020b), which are orthogonal to the discussions in this paper.

3. Notations in this Paper

In this paper, we follow the notations from (Frankle & Carbin, 2018) and generalize to the "pruning & fine-tuning" setup. Detailed notations (as shown in Figure 1) are illustrated as follows:

- *Initialization*: Given a network $f(x;\theta_0)$, where $\theta_0 \sim D_\theta$ denotes the initial weights.
- *Pre-training*: Train the network for T epochs arriving at weights θ_T and network function $f(x;\theta_T)$.
- *Pruning*: Based on the trained weights θ_T, adopt certain algorithm to generate a pruning mask $m \in \{0,1\}^{|\theta|}$. The LTH paper (Frankle & Carbin, 2018) uses the iterative pruning algorithm. We start from this algorithm for a fair evaluation, but are not restricted to it. Other algorithms, e.g., one-shot pruning and ADMM-based pruning are also employed to evaluate the impact on sparse training and pruning & fine-tuning, as shown in Section 5.
- *Sparse Training (Lottery Ticket Setting):* The LTH paper considers two cases in the sparse training setup. The first is the direct application of mask m to initial weights θ_0, resulting in weights $\theta_0 \odot m$ and network function $f(x;\theta_0 \odot m)$. The LTH paper termed this case the "winning tickets"[1]. The second is the application of mask m to a random initialization of weights $\theta_0' \sim D_\theta$, resulting in weights $\theta_0' \odot m$ (network function $f(x;\theta_0' \odot m)$). This case is termed "random reinitialization" in the LTH paper. The weights after training $f(x;\theta_0 \odot m)$ for T epochs are denoted by $(\theta_0 \odot m)_T$, while the weights after training $f(x;\theta_0' \odot m)$ for T epochs are denoted by $(\theta_0' \odot m)_T$. Please note that the mask m is kept through this training process.
- *Pruning & fine-tuning:* After generating the mask m, we directly apply it to the trained weights θ_T, resulting in weights $\theta_T \odot m$, and perform fine-tuning (retraining) for another T' epochs. The final weights are denoted by $(\theta_T \odot m)_{T'}$. To maintain the same number of total epochs as the lottery ticket setting, we set $T' = T$. Please note that the mask m is kept through this fine-tuning process.

The *winning property* has twofold meaning: First, training $f(x;\theta_0 \odot m)$ for T epochs (or fewer) will result in similar accuracy as $f(x;\theta_T)$ (pre-training result of the dense network), under a non-trivial pruning rate. Second, there should be a notable accuracy gap between training $f(x;\theta_0 \odot m)$ for T epochs and training $f(x;\theta_0' \odot m)$, and the former shall be higher.

4. Why Lottery Ticket Exists? An Analysis from the Weight Correlation Perspective

4.1. Revisiting Lottery Ticket: When does this winning property exist?

We revisit the lottery ticket experiments on various DNN architectures including VGG-11, ResNet-20, and MobileNet-V2 on the CIFAR-10 and CIFAR-100 datasets. Our goal is

[1]We inherit this terminology, although it does not result in the winning property in many of our testing results.

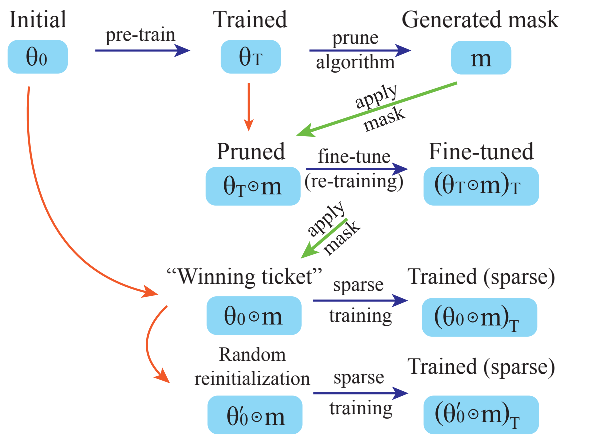
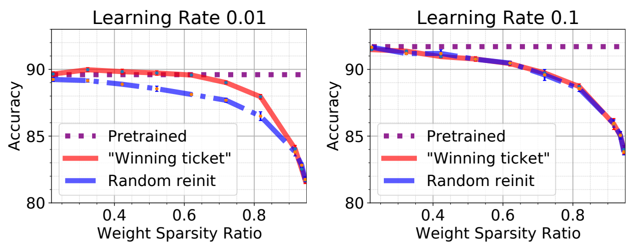


Figure 1. Illustration of notations: "pre-training", "pruning" (mask generation), "sparse training", and "pruning & fine-tuning".

to investigate the precise condition when winning property exists. We explore two different initial learning rates. The pruning approach for deriving masks follows the iterative pruning in Frankle & Carbin (2018). Namely, iteratively remove a percentage of the weights with the least magnitudes in each layer. In each iterative pruning round, reset the weights to the initial weight θ_0. We use the uniform per-layer pruning rate. Note the first convolutional layer is not pruned for all DNNs in this work.



(a) Iterative pruning at learning rate of 0.01. (b) Iterative pruning at learning rate of 0.1.

Figure 2. Illustration of random reinitialization and "winning tickets" for ResNet-20 on CIFAR-10 at learning rates 0.01 and 0.1.

Figure 2 illustrates the experiments of accuracy comparison between random reinitialization and "winning ticket" (both sparse training) for ResNet-20 on CIFAR-10 at learning rates 0.01 and 0.1 over a range of different sparsity ratios (Frankle & Carbin (2018) uses the low learning rate 0.01). We conduct each experiment *five* times (result variation shown in the figure). We set the same training epochs 150 rounds for training the original DNN with initial weights $f(x;\theta_0)$ (i.e., pretraining), training from randomly reinitialized weights with the mask $f(x;\theta_0' \odot m)$ (random reinitialization), and training from the initial weights with the mask $f(x;\theta_0 \odot m)$ ("winning ticket"). The hyperparameters used are the same as (Frankle & Carbin, 2018): SGD with momentum (0.9), and the learning rates decrease by a factor of 10 after 80 and 120 epochs. The batch size is 128. No additional training tricks are utilized throughout the paper for fairness in comparison.

In the case of the initial learning rate of 0.01, the pre-trained DNN's accuracy is 89.62%. The "winning tickets" consistently outperform the random reinitialization over different sparsity ratios. It achieves the highest accuracy 90.04% (higher than the pre-trained DNN) at sparsity ratio of 62%. This is similar to the observations found in (Frankle & Carbin, 2018) on the same network and dataset. On the other hand, in the case of the initial learning rate of 0.1, the pre-trained DNN's accuracy is 91.7%. In this case, the "winning ticket" has a similar accuracy performance as the random reinitialization, and cannot achieve the accuracy close to the pre-trained DNN with a reasonable sparsity ratio (say 50% or beyond). Thus no winning property is satisfied. Similar results can be found in the experiments of ResNet-20, VGG-11 and MobileNet-v2 on both CIFAR-10 and CIFAR-100, while the results of the rest of the experiments are detailed in Appendix A.

From these experiments, the winning property exists at a low learning rate but does not exist at a relatively high learning rate, which is also observed in (Liu et al., 2018). However, we would like to point out that the relatively high learning rate 0.1 (which is in fact the standard learning rate on these datasets) results in a *notably higher accuracy* in the pre-trained DNN (91.7% vs. 89.6%) than the low learning rate[2]. The associated sparse training results ("winning ticket", random reinitialization) in the lottery ticket setting are also higher with the learning rate 0.1. This point is largely missing in the previous discussions. Now the key **question** is: Are the above two observations correlated? If the answer is yes, it means that the winning property is not universal to DNNs, nor is it a natural characteristic of DNN architecture or application. Rather, it indicates that the learning rate is not sufficiently large, and the original, pretrained DNN is not well-trained.

Our hypothesis is that the above observations are correlated, and this is largely attributed to the correlation between initialized weights and final-trained weights when the learning rate is not sufficiently large. Before validating our hypothesis, we will introduce a *correlation indicator* (CI) for quantitative analysis.

4.2. Weight Correlation Indicator

Consider a DNN with two collections of weights θ and θ'. Note that this is a general definition that applies to both the original DNN and sparse DNN (when the mask is

[2]As CIFAR-10 is a relatively small dataset, 2% accuracy is a notable accuracy difference that should not be ignored.

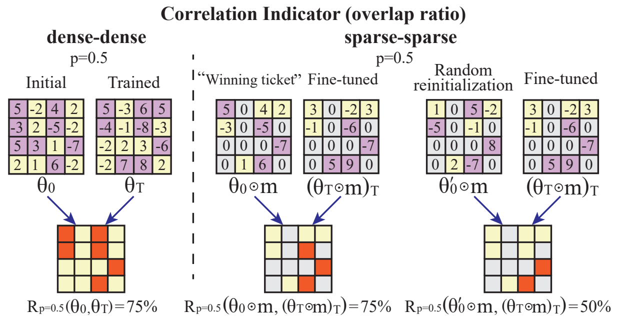


Figure 3. Scenarios for quantitative analysis of the weight correlation with an example of $sparsity\ ratio = 50\%$ and $p = 0.5$. This example is one DNN layer, while our actual computation is on the whole DNN.

applied and a portion of weights eliminated). We define the *correlation indicator* to quantify the amount of overlapped indices of large-magnitude weights between θ and θ'. More specifically, given a DNN with L layers, where the l-th layer has N_l weights, the *weight index set* $T_p\big((\theta)^l\big)$ ($p \in [0,1]$) is the top-$p \cdot 100\%$ largest-magnitude weights in the l-layer. Similarly, we define $T_p\big((\theta')^l\big)$. Please note that for a sparse DNN, the portion p is defined with respect to the number of remaining weights in the sparse (sub-)network[3]. The intersection of these two sets includes those weights that are large (top-$p \cdot 100\%$) in magnitude in both θ and θ', and $\textbf{card}\Big(T_p\big((\theta)^l\big) \cap T_p\big((\theta')^l\big)\Big)$ denotes the number of such weights in layer l. The correlation indicator (overlap ratio) between θ and θ' is finally defined as:

$$R_p(\theta, \theta') = \frac{\sum_l \textbf{card}\Big(T_p\big((\theta)^l\big) \cap T_p\big((\theta')^l\big)\Big)}{p \cdot \sum_l N_l} \tag{1}$$

When $R_p(\theta, \theta') \approx p$, the top-$p \cdot 100\%$ largest-magnitude weights in θ and θ' are largely independent. In this case the correlation is relatively weak[4]. On the other hand, if there is a large deviation of $R_p(\theta, \theta')$ from p, then there is a strong correlation. Especially when $R_p(\theta, \theta') > p$, the weights that are large in magnitude in θ are likely to also be large in θ', indicating a positive correlation. Otherwise, when $R_p(\theta, \theta') < p$, it implies a negative correlation.

As shown in Figure 3, the above correlation indicator will be utilized to quantify the correlation between a dense DNN and a dense DNN, i.e., $R_p(\theta_0, \theta_T)$ for DNN pre-training, and between a sparse DNN and a sparse DNN, i.e., $R_p(\theta_0 \odot m, (\theta_T \odot m)_T)$ and $R_p(\theta_0' \odot m, (\theta_T \odot m)_T)$ for the cases of "winning ticket" and random reinitialization under lottery ticket setting. Next, we will use the former to demonstrate the effect of different learning rates in DNN pre-training and the latter to demonstrate the rationale and condition of winning property.

[3]In this way the formula can be unified for dense and sparse DNNs.

[4]We cannot say that there is no correlation here because $R_p(\theta, \theta') \approx p$ is only a necessary condition.

4.3. Weight Correlation in DNN Pre-Training

Intuitively, the *weight correlation* means that if a weight is large in magnitude at initialization, it is likely to be large after training. The reason for such correlation is that the learning rate is too low and weight updating is slow. Such weight correlation is not desirable for DNN training and typically results in lower accuracy, as weights in a well-trained DNN should depend more on the location of those weights instead of initialization (Liu et al., 2018). Thus when such weight correlation is strong, the DNN accuracy will be lower, i.e., not well-trained.

To validate the above statement, we have performed experiments to derive $R_p(\theta_0, \theta_T)$ on DNN pretraining with different initial learning rates. Using ResNet-20 on CIFAR-10 dataset as an illustrative example. Figure 4 illustrates the correlation indicator between the initial weights θ_0 and the trained weights θ_T from DNN pretraining at learning rates of 0.01 and 0.1, respectively. We use $T = 150$ the same as Section 4.1, also the same other hyperparameters and no additional training tricks. We can observe that $R_p(\theta_0, \theta_T)$ at learning rate 0.01 has a notably higher correlation compared to the case of learning rate 0.1. This observation indicates that the large-magnitude weights of θ_0 are not fully updated at a low learning rate 0.01, indicating that the pre-trained DNN is not well-trained. In the case of learning rate 0.1, the weights are sufficiently updated thus largely independent from the initial weights ($R_p(\theta_0, \theta_T) \approx p$, where $p = 10\%, 20\%, 30\%, 40\%, 50\%$), indicating a well-trained DNN. Results on other DNN models and datasets are provided in Appendix B, and a similar conclusion can be drawn.

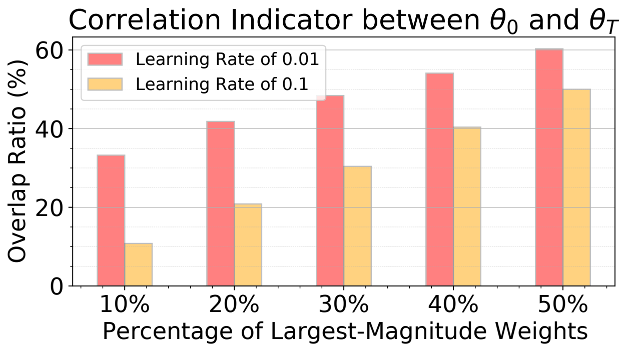


Figure 4. The overlap ratios (when $p = 10\%$, 20%, 30%, 40% and 50%) between the initial weights θ_0 and the pretrained weights θ_T at learning rate of 0.01 and 0.1.

As shown in the result discussions, learning rates 0.1 and 0.01 (for ResNet-20) are not merely two candidate hyperparameter values. Rather, they result in a well-trained DNN (so a desirable learning rate) and a not well-trained DNN (so a not-so-good learning rate), respectively. We shall not rely on the conclusion drawn from the latter that results in an insufficient DNN pre-training.

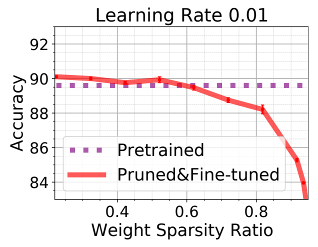
4.4. Cause and Condition of the Winning Property

Weight Correlation under Lottery Ticket Setting: In this subsection, our goal is to understand the different accuracy from training $f(x; \theta_0 \odot m)$ ("winning ticket") and training $f(x; \theta_0' \odot m)$ (random reinitialization) when the learning rate is low, thereby revealing the cause and condition of winning property. We will achieve this goal by studying the weight correlation.

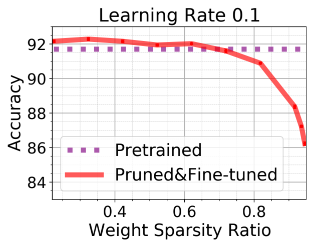
Consider the "pruning & fine-tuning" case formally defined in Section 3, in which we apply mask m on the trained weights θ_T from DNN pretraining, and then perform fine-tuning for another T epochs. The final weights are denoted by $(\theta_T \odot m)_T$. Using ResNet-20 on CIFAR-10 as an illustrative example. Figure 5(a) and 5(b) show the accuracy of the "pruning & fine-tuning" result $f\big(x; (\theta_T \odot m)_T\big)$ at different sparsity ratios, with learning rates 0.01 and 0.1, respectively. Again we use $T = 150$ epochs and the same other hyperparameters. The accuracies of the pretrained DNN with corresponding learning rates are also provided. One can observe that $f\big(x; (\theta_T \odot m)_T\big)$ achieves relatively high accuracy, close to or higher than the accuracy of the pretrained DNN at the same learning rate (even at the desirable learning rate 0.1)[5]. Results on other DNN models and datasets are provided in Appendix C, and a similar conclusion can be drawn.

We study the correlation between $\theta_0 \odot m$ ($\theta_0' \odot m$) and $(\theta_T \odot m)_T$ to shed some light on the cause of winning property. Again use ResNet-20 on CIFAR-10 as an illustrative example, while the results on other DNN models and datasets are provided in Appendix D with similar conclusion. Figure 5(c) shows the correlation indicator between $\theta_0 \odot m$ ("winning ticket") and $(\theta_T \odot m)_T$, and between $\theta_0' \odot m$ (random reinitialization) and $(\theta_T \odot m)_T$, under the insufficient learning rate 0.01. While Figure 5(d) shows the correlation indicator results under the desirable learning rate 0.1. One can observe the positive correlation between $\theta_0 \odot m$ and $(\theta_T \odot m)_T$ at the low learning rate, when the winning property exists. Such correlation is minor in the other cases.

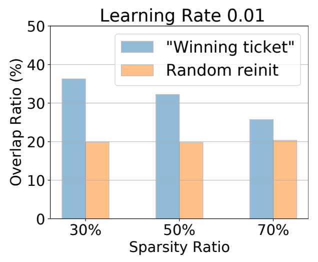
Analysis of Weight Correlation and Condition of Winning Property: Let us investigate the **cause** of correlation between $\theta_0 \odot m$ and $(\theta_T \odot m)_T$ at low learning rate. As



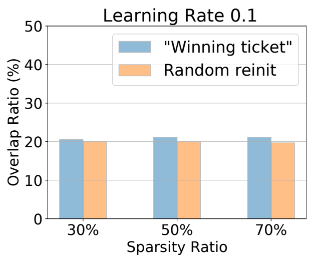
(a) Pruning&fine-tuning at learning rate of 0.01.



(b) Pruning&fine-tuning at learning rate of 0.1.



(c) The overlap ratio comparison at learning rate 0.01.



(d) The overlap ratio comparison at learning rate 0.1.

Figure 5. (a), (b): Accuracies of $f\big(x; (\theta_T \odot m)_T\big)$ ("pruning & fine-tuning") at different sparsity ratios with masks generated by iterative pruning. (c), (d): The weight correlation (overlap ratio) comparison at $p = 0.2$, between $\theta_0 \odot m$ ("winning ticket") and $(\theta_T \odot m)_T$ (pruned&fine-tuned weights), and between $\theta_0' \odot m$ (random reinitialization) and $(\theta_T \odot m)_T$ (pruned&fine-tuned weights) under 0.3, 0.5, 0.7 sparsity ratios.

shown in Section 4.3, there is a correlation between θ_0 and θ_T at the insufficient learning rate. Then there is also a correlation between $\theta_0 \odot m$ and $\theta_T \odot m$ (both applied the same mask). As $\theta_T \odot m$ includes the pretrained weights and $(\theta_T \odot m)_T$ only applies additional fine-tuning, there will be positive correlation between $\theta_T \odot m$ and $(\theta_T \odot m)_T$. Combining the above two statements will yield the correlation between $\theta_0 \odot m$ and $(\theta_T \odot m)_T$. When we consider random reinitialization, there is no correlation between θ_0' and θ_T as a reinitialization is applied. So there is no correlation between $\theta_0' \odot m$ and $\theta_T \odot m$, or between $\theta_0' \odot m$ and $(\theta_T \odot m)_T$.

At a desirable learning rate 0.1, there is a minor (or no) correlation between θ_0 and θ_T as shown in Section 4.3. As a result, there is minor (or no) correlation between $\theta_0 \odot m$ and $\theta_T \odot m$, or between $\theta_0 \odot m$ and $(\theta_T \odot m)_T$. From the above analysis, one can observe that the correlation between θ_0 and θ_T is the **key condition** in weight correlation analysis.

The positive correlation between $\theta_0 \odot m$ and $(\theta_T \odot m)_T$ helps to explain the winning property at low learning rate. Compared with random reinitialization $\theta_0' \odot m$, the "winning ticket" $\theta_0 \odot m$ is "closer" to a reasonably accurate solution. As the weight upscaling is slow (learning rate insufficient),

[5]In fact, the relatively high accuracy of $f\big(x; (\theta_T \odot m)_T\big)$ is one major reason for us to explore the correlation between $\theta_0 \odot m$ ($\theta_0' \odot m$) and $(\theta_T \odot m)_T$. In Section 5 we will generalize to the conclusion that "pruning & fine-tuning" results in higher accuracy in general than sparse training (the lottery ticket setting).

it takes less effort to reach a higher accuracy starting from $\theta_0 \odot m$ compared with starting from $\theta_0' \odot m$. Besides, as pointed out in Section 4.3, the insufficient learning rate (and correlation between θ_0 and θ_T) results in a low accuracy in the pre-trained DNN, which makes it easier for sparse training to reach its accuracy. On the other hand, at a sufficient learning rate, such correlations do not exist (or are very minor), and then the winning property does not exist.

Remarks: From the above analysis, we conclude that a key condition of winning property is the correlation between θ_0 and θ_T. However, as already demonstrated in Section 4.3, under the same condition the pretrained network will not be well-trained, as weights in a well-trained DNN should depend more on the location of those weights instead of initialization. In fact, as shown in Figure 2 and Appendix A, the "winning ticket" can only restore the accuracy of the pretrained DNN under the same insufficient learning rate, instead of reaching the pretrained DNN accuracy at a desirable learning rate. This makes the value of the winning property questionable.

4.5. Takeaway

As discussed above, the existence of winning property is correlated with an insufficient DNN pretraining. It seems that winning property is not a natural characteristic of DNN architecture or application, and is unlikely to occur for a well-trained DNN (with a desirable learning rate). As a result, we do not suggest investigating the winning property under an insufficient learning rate.

5. Pruning & Fine-tuning – A Better Way to Restore Accuracy under Sparsity

As concluded from the above discussions, it is difficult for sparse training to restore the accuracy of the pre-trained dense DNN, when a desirable learning rate is applied. On the other hand, as already hinted in Section 4.4, the "pruning & fine-tuning" (i.e., fine-tuning from $(\theta_T \odot m)_T$) exhibits a higher capability in achieving the accuracy of a pre-trained DNN, no matter what the learning rate is. Compared with the lottery ticket setting, the only difference in "pruning & fine-tuning" is that the mask m is applied to the pretrained weights θ_T. Is this the key reason for the high accuracy? Is this a universal property for different DNN architectures and applications? If the answer is yes, what is the underlying reason? We aim to answer these questions.

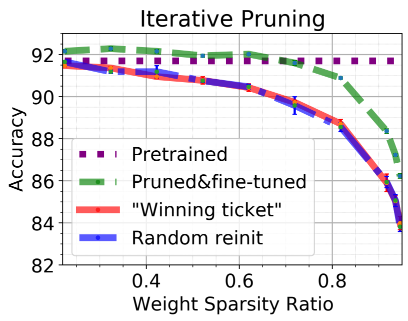
In this section, we only consider the **desirable learning rate** (e.g., 0.1 for ResNet-20 on CIFAR-10 dataset) and sufficient DNN pre-training, as the associated conclusions will be more meaningful.

Fair Comparison with Lottery Ticket Setting: We claim that under the same T epochs for fine-tuning and sparse training, it is a fair comparison. The generation of mask m is the same. The only difference is that pruning & fine-tuning applies mask m to the pre-trained weights θ_T, while sparse training applies m to θ_0. Note that θ_T is available before m as the latter is derived based on θ_T using pruning algorithm. Thus pruning & fine-tuning will have **no additional training epochs** compared with sparse training.

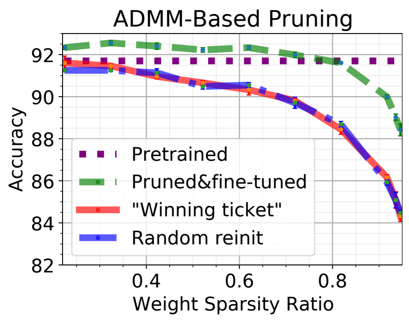
Comparison between Pruning & Fine-tuning and Sparse Training: We use ResNet-20 on CIFAR-10 dataset as an illustrative example, and the rest of results are provided in Appendix E (with the similar conclusion). We use the desirable learning rate 0.1, $T = 150$ epochs, and the same as Section 4.1 for the rest of hyperparameters. Figure 6(a) shows the accuracy comparison between pruning & fine-tuning (i.e., training (fine-tuning) from $\theta_T \odot m$) and the two sparse training scenarios "winning ticket" (i.e., training from $\theta_0 \odot m$) and random reinitialization (i.e., training from $\theta_0' \odot m$) at different sparsity ratios. Iterative pruning algorithm is used to derive mask m here. One can clearly observe the accuracy gap between pruning & fine-tuning and the two sparse training cases (lottery ticket setting). In fact, the pruning & fine-tuning scheme can consistently outperform the pretrained dense DNN up to sparsity ratio 70%. Again, there is no accuracy difference between the two sparse training cases.

Furthermore, we consider other two candidate pruning algorithms, ADMM-based pruning (Zhang et al., 2018) and one-shot pruning, for pruning mask generation. Figure 6(b) and Figure 6(c) demonstrate the corresponding accuracy comparison results between pruning & fine-tuning and the two sparsity training scenarios. Again one can observe the notable advantage of pruning & fine-tuning over the lottery ticket setting, even with a weak one-shot pruning algorithm for mask generation. In fact, pruning & fine-tuning under ADMM-based pruning can restore the accuracy of pretrained DNN with 80% sparsity. The property is not found under any of these pruning algorithms. Clearly, the consistent advantage of pruning & fine-tuning is attributed to the fact that mask m is applied to pretrained weights θ_T instead of the initialized weights θ_0. In fact, information in θ_T is important for the sparse subnetwork to maintain accuracy of the pretrained dense network. Or in other words, weights in the desirable sparse subnetwork should have correlation with θ_T instead of θ_0.

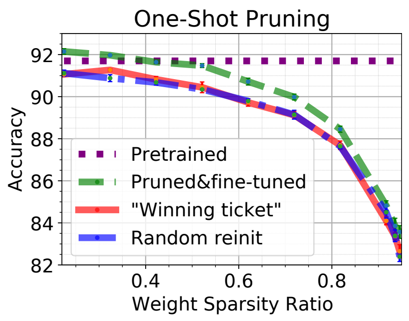
Effect of Different Pruning Algorithms – Towards a Better Mask Generation: We have tested three pruning algorithms for mask generation. How to evaluate their relative performance? Figure 7 combines the above results and demonstrates the accuracy performances of pruning & fine-tuning and sparse training ("winning ticket" case), under all three pruning algorithms. The rest of results are in Appendix E. One can observe the order in the accuracy perfor-



(a) The iterative pruning at learning rate of 0.1.



(b) The ADMM-based pruning at learning rate of 0.1.



(c) The least magnitude one-shot pruning at learning rate of 0.1.

Figure 6. Accuracy performance of pruning & fine-tuning vs. two sparse training cases ("winning ticket" and random reinitialization). Three pruning algorithms are utilized for mask generation: iterative pruning (a), ADMM-based pruning (Zhang et al., 2018) (b), and one-shot pruning (c).

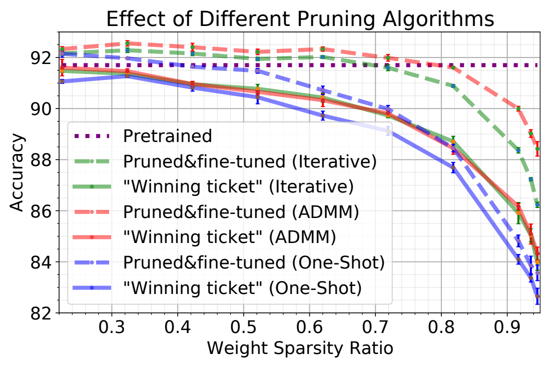


Figure 7. Accuracy performances of pruning & fine-tuning and sparse training ("winning ticket" case), under all three pruning algorithms (iterative pruning, ADMM-based pruning, and one-shot pruning) for mask generation.

mance: ADMM-based pruning on top, iterative pruning in the middle, and one-shot pruning the lowest. This order is the same for pruning & fine-tuning and sparse training. Note that the pruning algorithm is utilized to generate mask m, while the other conditions are the same (i.e., θ_T, fine-tuning T epochs on $\theta_T \odot m$, or sparse training on $\theta_0 \odot m$). Hence, the relative performance is attributed to the quality in mask generation. One can conclude that the selection of pruning algorithm is critical in generating the sparse subnetwork as the quality of mask generation plays a key role here.

An Analysis from Weight Correlation Perspective: We conduct a weight correlation analysis of pruning & fine-tuning results that can largely restore the accuracy of pre-trained, dense DNN, between the final weights $(\theta_T \odot m)_T$ and the initialization θ_0. Detailed results and discussions are provided in Appendix F. The major conclusion is that there is a lack of correlation between $(\theta_T \odot m)_T$ and θ_0, but there is a correlation between $(\theta_T \odot m)_T$ and θ_T. It further strengthens the conclusion that it is not desirable to have the weight correlation between final-trained weights and weight initialization.

Comparison with Frankle et al. (2019): The work Frankle et al. (2019) suggests applying mask m to θ_k and then apply sparse training, where θ_k denotes the weights trained from θ_0 for a small number of k epochs. This technique is training from $\theta_k \odot m$, and is in between sparse training (training from $\theta_0 \odot m$) and pruning & fine-tuning (training from $\theta_T \odot m$). We point out that these three cases require **the same number of total epochs** under the same pruning algorithm, as mask m is generated later than θ_k or θ_T. We conduct a comprehensive comparison on the relative performance, with detailed results and discussions in Appendix G. The major conclusion is that pruning & fine-tuning consistently outperforms the method Frankle et al. (2019) over different sparsity ratios, DNN models, and datasets. As they exhibit the same number of training epochs, we suggest directly applying the mask m to θ_T and perform fine-tuning, instead of applying to θ_k.

Remarks: If one wants to optimize the accuracy of sparse subnetwork and restore the accuracy of the pretrained dense DNN, we suggest adopting the pruning & fine-tuning method instead of lottery ticket setting.

6. Conclusion

In this work, we investigate the underlying condition and rationale behind lottery ticket property. We introduce a correlation indicator for quantitative analysis. Extensive experiments over multiple deep models on different datasets have been conducted to justify that the existence of winning property is correlated with an insufficient DNN pretraining, and is unlikely to occur for a well-trained DNN. Meanwhile, the sparse training of lottery ticket setting is difficult to restore the accuracy of the pre-trained dense DNN. To overcome this limitation, we propose the "pruning & fine-tuning" method that consistently outperforms lottery ticket sparse training under the same pruning algorithm and total training epochs over various DNNs on different datasets.

Acknowledgements

This research was supported by the National Science Foundation under awards CNS-1704662 and CCF-1937500. Any opinions, findings, and conclusions or recommendations in this material are those of the authors and do not necessarily reflect the views of NSF.

References

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*, 2020.

Dai, X., Yin, H., and Jha, N. Nest: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 2019.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*, 2018.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Pruning neural networks at initialization: Why are we missing the mark? *arXiv preprint arXiv:2009.08576*, 2020a.

Frankle, J., Schwab, D. J., and Morcos, A. S. The early phase of neural network training. *arXiv preprint arXiv:2002.10365*, 2020b.

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 (NIPS)*, pp. 1135–1143, 2015a.

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 (NeurIPS)*, pp. 1135–1143, 2015b.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 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 (ICCV)*, pp. 1389–1397, 2017.

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 (CVPR)*, pp. 4340–4349, 2019.

He, Y., Ding, Y., Liu, P., Zhu, L., Zhang, H., and Yang, Y. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., and Shao, L. Hrank: Filter pruning using high-rank feature map. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Liu, N., Ma, X., Xu, Z., Wang, Y., Tang, J., and Ye, J. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In *AAAI*, 2020.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Min, C., Wang, A., Chen, Y., Xu, W., and Chen, X. 2pfpce: Two-phase filter pruning based on conditional entropy. *arXiv preprint arXiv:1809.02220*, 2018.

Morcos, A. S., Yu, H., Paganini, M., and Tian, Y. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv preprint arXiv:1906.02773*, 2019.

Ouyang, H., He, N., Tran, L., and Gray, A. Stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, pp. 80–88. PMLR, 2013.

Ren, A., Zhang, T., Ye, S., Xu, W., Qian, X., Lin, X., and Wang, Y. Admm-nn: an algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In *ASPLOS*, 2019.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Tan, C. M. J. and Motani, M. Dropnet: Reducing neural network complexity via iterative pruning. In *International Conference on Machine Learning*, pp. 9356–9366. PMLR, 2020.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems (NeurIPS)*, pp. 2074–2082, 2016.

Ye, M., Gong, C., Nie, L., Zhou, D., Klivans, A., and Liu, Q. Good subnetworks provably exist: Pruning via greedy forward selection. In *International Conference on Machine Learning*, pp. 10820–10830. PMLR, 2020.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

Zhang, T., Ye, S., Zhang, Y., Wang, Y., and Fardad, M. Systematic weight pruning of dnns using alternating direction method of multipliers. *arXiv preprint arXiv:1802.05747*, 2018.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.