Good Students Play Big Lottery Better

Haoyu Ma[1], Tianlong Chen[2], Ting-Kuei Hu[3], Chenyu You[4], Xiaohui Xie[1], Zhangyang Wang[2]
[1]University of California, Irvine, [2]University of Texas at Austin,
[3]Texas A&M University, [4]Yale University
{haoyum3,xhx}@uci.edu,{tianlong.chen, atlaswang}@utexas.edu,
tkhu@tamu.edu, chenyu.you@yale.edu

Abstract

Lottery ticket hypothesis [1] suggests that a dense neural network contains a sparse sub-network that can match the test accuracy of the original dense net when trained in isolation from (the same) random initialization. However, the hypothesis failed to generalize to larger dense networks such as ResNet-50. As a remedy, recent studies [2, 3] demonstrate that a sparse sub-network can still be obtained by using a *rewinding* technique, which is to re-train it from early-phase training weights or learning rates of the dense model, rather than from random initialization.

Is rewinding the only or the best way to scale up lottery tickets? This paper proposes a new, simpler and yet powerful technique for re-training the sub-network, called "Knowledge Distillation ticket" (**KD ticket**). Rewinding exploits the value of inheriting knowledge from the *early training phase* to improve lottery tickets in large networks. In comparison, KD ticket addresses a *complementary possibility* - inheriting useful knowledge from the *late training phase* of the dense model. It is achieved by leveraging the soft labels generated by the trained dense model to re-train the sub-network, instead of the hard labels. Extensive experiments are conducted using several large deep networks (e.g ResNet-50 and ResNet-110) on CIFAR-10 and ImageNet datasets. Without bells and whistles, when applied by itself, KD ticket performs on par or better than rewinding, while being nearly free of hyperparameters or ad-hoc selection. KD ticket can be further applied together with rewinding, yielding state-of-the-art results for large-scale lottery tickets.

1 Introduction

Deep neural networks have seen prevailing success in many tasks such as classification [4], object detection [5], and semantic segmentation [6]. However, the increasing computational demand with regard to the growth of the number of parameters makes most existing models less practical in real-world deployments. To address the issue, network pruning identifies sparse sub-networks from a dense neural network by removing unnecessary weights while maintaining similar accuracy [7, 8, 9, 10].

The recent *Lottery Ticket Hypothesis*(LTH) [1] points to a brand-new opportunity to identify an extremely sparse and self-trainable sub-network from a dense model. Such sub-network can be found via the following three steps: (1) training a dense network from scratch; (2) pruning unnecessary weights to form a sparse sub-network; and (3) re-training the sub-network from the same random initial weights used in the dense model (winning ticket). The authors demonstrate on several small-to medium-scale networks, the such re-trained sub-network can be achieve comparable accuracy with the original dense model. However, the efficacy of lottery ticket hypothesis met some challenge while being scaled up to large networks. Later research efforts found that, while it was generally hard to achieve comparable performance of the sub-network from totally (even the same) random initialization, some “auxiliary information" could be obtained from the **early training phase** of

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

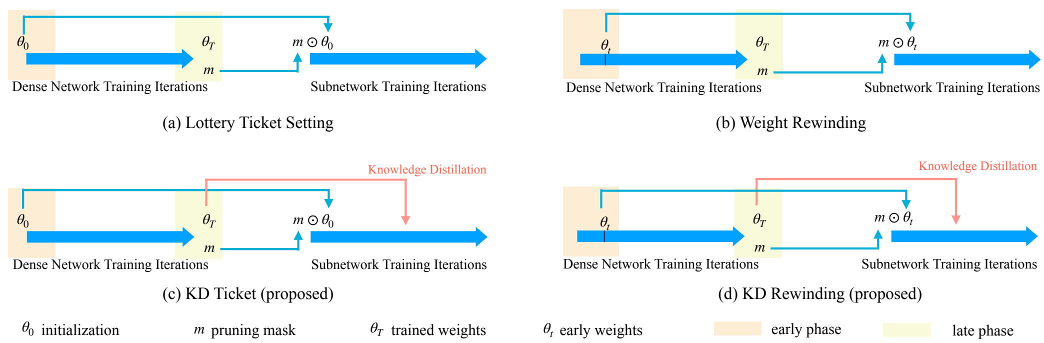


Figure 1: Illustration of our proposed KD ticket and KD rewinding. Most LTH works mainly inherit knowledge from early training phase, while only use the late phase to find pruning masks. Our method presents knowledge distillation as a technique bringing late knowledge into current LTH (KD Ticket), and the combination of both early and late phase knowledge can further boost performance (KD Rewinding).

the dense network (first step) to aid the re-training (third step). Specifically, *weight rewinding* [2] proposed to “roll back" the dense model weights to some early training iteration (but not the very beginning), and to use the rewound weight as the re-training initialization. That was found to help difficult lottery ticket finding tasks, such as in large networks and/or at extremely high pruning ratios. [3] then presented a simplified version called *learning rate rewinding*, i.e., rolling back the learning rate schedule rather than the weight status, for the retraining to start with. Those efforts suggest the promise of “recycling" useful knowledge from the original training for finding large-scale lottery tickets.

This paper is inspired by the success of rewinding, and explores an orthogonal direction: *will the **late training phase** of the dense model provide helpful side information for large-scale ticket finding too?* To accomplish the goal, we propose *knowledge distillation ticket* (**KD ticket**). By leveraging the classical tool of knowledge distillation [11], our KD ticket re-trains the sub-network (third step) by employing the soft-labels generated by the trained dense model (first step), instead of the one-hot hard labels only. The KD ticket is almost free of hyperparameter, and avoids the ad-hoc selection of the rewinding epoch. Furthermore, as the KD ticket draws auxiliary knowledge from the different phases from the rewinding (late versus early), they are complementary and can be naturally applied together. We thoroughly examine the effectiveness of KD tickets using several large deep networks (e.g ResNet-50 and ResNet-110) on CIFAR-10 and ImageNet. Without bells and whistles, KD tickets alone lead to comparable or better performance than rewinding. When the two are coupled together, we witness a further performance boost, reaching state-of-the-art performance in large-scale ticking findings.

Our contributions can be summarized as follows:

- We reveal a new opportunity for large-scale lottery ticket finding, by introducing KD ticket to utilize soft-labels as an extra model-level cue from the past training.
- We demonstrate that our KD ticket is compatible to other "rewinding" methods, dubbed KD-rewinding, in which way further performance boost can be achieved.
- We conduct extensive experiments on CIFAR-10 and ImageNet datasets using large models. The tickets identified by our approaches achieve the state-of-the-art results.

2 Methodology

2.1 Revisiting the Weight Rewinding

The lottery ticket hypothesis [1] demonstrates that neural networks contain far smaller subnetworks that emerge early in training, and can be trained in isolation to the same or better accuracy as the full network. More formally, consider a neural network $f_\theta(\cdot)$ with initial parameters $\theta_0 \in \mathbb{R}^d$. Let $m \in \{0,1\}^d$ be the pruning mask, such that the element-wise product $\theta \odot m$ denotes the pruned network. To overcome the scalability/stability issue of lottery tickets in larger networks, Frankle et al. [2] proposed the *weight rewinding* technique to identify the winning tickets for retraining.

1. Train the full network to convergence with T epoch, arriving at weights θ_T
2. Prune $p\%$ of the parameters in θ_T to create a binary mask m. Then, rewind the remaining weights to an intermediate values θ_t, at the training epoch t
3. Starting from θ_t, re-train the pruned subnetwork $f_{\theta_t \odot m}(\cdot)$ for $T - t$ epochs, using the same training parameters and learning rate schedule as original.

Experiments in [1] show that for small networks, we can rewind to $t = 0$, i.e., the original random initialization, to find the winning tickets. The follow-up works [2, 3] prove that for deeper networks, either rewinding the weights to the ones at early training iteration, or rewinding learning rate schedule to the early phase, can achieve better performance than rewinding to iteration 0, when seeking lottery tickets in larger networks. The success of "weight rewinding" implies an opportunity to improve the LTH by inheriting knowledge from the early training phase.

2.2 Knowledge Distillation Ticket: Intuition, Implementation, and Extension

Rationale The main goal of "weight rewinding" is to inherit and utilize knowledge from past full training. Our *key inspiration* is that: the soft labels generated by the trained dense model contain knowledge of the late training phase. That may be helpful for lottery ticket finding too, and could be complementary to the early-phase knowledge leveraged by rewinding.

The rewinding studies [2, 3] did not find directly inheriting *late stage model weights* to yield any competitive lottery ticket. However, we suspect it more likely to be an artifact of the optimization algorithm, rather than indicating the actual value of the late-stage model. Recent efforts in investigating deep learning optimization [12, 13, 14] discovered that the training of deep networks using gradient-based algorithms display distinct behaviors at the early and late stages (usually empirically divided by the validation error plateau, or by the learning-rate switch point(s)). The early stage is found to be pivotal to the final models' generalization ability, but the objective loss does not notably decrease meanwhile. In contrast, the late stage exhibits more "convex-like" optimization behavior in decreasing the loss, but is not a primary contributor to model generalization.

Those findings echo practical observations: if we take a late-stage trained weight as the initialization for training another new model, then its generalization gain is often limited even after new (re-)training. Further, note that weight rewinding to a late epoch would exploit a very sparse subset of trained weight, whose own generalization performance would be sharply weakened after pruning.

In plain words, the late-stage training weight might be too "stuck" and not grant sufficient exploration flexibility to start off the new ticket training. Then, what are our alternatives to gain this part of useful knowledge, without sacrificing the flexibility? We notice that, besides model-based information like weight and learning rate schedule, the knowledge from past training can also be acquired from data.

The *knowledge distillation* technique [11] reveals that the predicted probability distribution over classes for each sample implies how the model generalizes. Inspired by that, we leverage this classical tool to pass on the late-stage knowledge. Specifically, during re-training the sub-network (third step), we employ the soft-labels generated by the trained dense model (first step), instead of the one-hot hard labels, as the output targets. We call this new technique *knowledge distillation ticket* (KD ticket). Figure 1 present an illustration of our proposed method and its difference with previous rewinding techniques.

KD Ticket For a C-class classfication task, given a trained dense network $f_{\theta_T}(\cdot)$ and input x^i, we leverage the logit $z^i \in \mathbb{R}^C$ (the final output before softmax layer) from $f_{\theta_T}(x^i)$ to supervise the desired sparse sub-network $f_{\theta \odot m}(\cdot)$. Following the setting of standard knowledge distillation, the logit z^i is distilled to the knowledge $q^i \in \mathbb{R}^C$ by the temperature τ according to the following:

$$q_j^i = \sigma_\tau(z_j^i) = \frac{\exp\left(z_j^i/\tau\right)}{\sum_{j=1}^{C} \exp\left(z_j^i/\tau\right)}, \tag{1}$$

where q_j^i denotes the jth element of q^i and $\sigma_\tau(\cdot)$ denotes the standard softmax function with the distilling temparature τ. Usually, τ is a positive value > 1, and a higher value for τ can produce a softer probability distribution over classes.

Given an un-pruned dense network $f_{\theta_0}(\cdot)$ with random initialization θ_0, we firstly train it from scratch with T epochs and achieve $f_{\theta_T}(\cdot)$, and prune the network by the specified pruning ratio $p\%$, resulting a mask m and a sparse sub-network $f_{\theta \odot m}(\cdot)$. Then, we rewind the weights of the sub-network to the values of original initialization θ_0, and achieve $f_{\theta_0 \odot m}(\cdot)$. During the retraining process, we adopt the same hyper-parameters, such as learning rate and learning scheduler, as those we used during the training of $f_{\theta_T}(\cdot)$. Then, provided $\{q^i\}_{i=1}^N$, a specified temperature τ, and hard labels $\{y^i\}_{i=1}^N$ over whole training set $\{x^i\}_{i=1}^N$, we supervise the sparse sub-network $f_{\theta \odot m}(\cdot)$ by minimizing the following mini-batch loss ℓ over K samples,

$$\ell = \sum_{i=1}^{K} (\alpha \tau^2 \mathcal{KL}(q^i, p^i) + (1-\alpha)\mathcal{XE}(f_{\theta \odot m}(x^i), y^i)) \tag{2}$$

where $\mathcal{KL}$ and $\mathcal{XE}$ represent KL divergence and cross entropy, respectively. Here, $q_j^i = \sigma_\tau(z_j^i)$ and $p_j^i = \sigma_\tau(s_j^i)$. In detail, z^i is the logit for x^i from fully-trained dense network $f_{\theta_T}(x^i)$, and s^i is the prediction logit for x^i from sub-network $f_{\theta \odot m}(x^i)$. The hyper-parameter $\alpha \in [0, 1]$ balances the KL divergence loss and cross entropy loss.

KD Rewinding The above KD ticket by default rewinds the weights to the original initialization θ_0. It will be demonstrated to be already competitive in out experiments. Furthermore, it is blithe to combine "weight rewinding" (early knowledge) with our "KD ticket" (late knowledge) to further boost the performance of sparse network $f_{\theta \odot m}(\cdot)$ together. We name the idea of combining "KD ticket" and "weight rewinding" as "KD rewinding", where the sparse network $f_{\theta \odot m}(\cdot)$ rewind the weights to the values of early training of $f_{\theta_T}(\cdot)$, and then starts to be re-trained with the supervision of distilled knowledge from the fully-trained $f_{\theta_T}(\cdot)$. The performance of both "KD ticket" and "KD rewinding" could be referred in section 3.

Table 1: Hyperparameters for networks and datasets

Dataset	Network	#Params	Optimizer	Learning rate
CIFAR-10	ResNet[c]-56 ResNet[c]-110 ResNet-18	0.85M 1.7M 11.2M	SGD, $\beta = 0.9$ Weight decay: 2e-4 Batch size: 128 Epochs :182	$\alpha = \begin{cases} 0.1 & \text{t} \in [0, 91) \\ 0.01 & \text{t} \in [91, 136) \\ 0.001 & \text{t} \in [136, 182] \end{cases}$
Tiny-ImageNet	ResNet[c]-32++	1.8M	SGD, $\beta = 0.9$ Weight decay:1e-4 Batch size:128 Epochs :300	$\alpha = \begin{cases} 0.1 & \text{t} \in [0, 150) \\ 0.01 & \text{t} \in [151, 225) \\ 0.001 & \text{t} \in [225, 300] \end{cases}$
ImageNet	ResNet-50	23.5M	SGD, $\beta = 0.9$ Weight decay:1e-4 Batch size:512 Epochs :90	$\alpha = \begin{cases} 0.4 \cdot \frac{t}{5} & t \in [0, 5) \\ 0.4 & t \in [5, 30) \\ 0.04 & t \in [30, 60) \\ 0.004 & t \in [60, 80) \\ 0.0004 & t \in [80, 90] \end{cases}$

3 Experiments

3.1 Experiment settings

To prove the efficacy of our proposed KD tickets, we study neural network pruning on various standard architectures for image classification tasks. Here, we discuss the datasets, networks, and training hyperparameters used in experiments.

Datasets. Similar to previous studies [1, 3], we perform experiments on CIFAR-10, Tiny-ImageNet and ImageNet. In detail, for CIFAR-10, we randomly divide the training set into 45,000 training examples and 5,000 validation examples. We select the model with the highest validation accuracy to evaluate the performance on test set. For ImageNet, we use the LSVRC 2012 [15]. Tiny-ImageNet is a subset of ImageNet with 200 classes, where each image is down-sized to 64 $\times$ 64 pixels.

Network. In keeping with the rewinding studies, we mainly focus on deeper neural networks. For experiments on CIFAR-10, we intentionally apply the simplified ResNet designed for this dataset (in short, ResNet[c])[16]. By design, this network requires its depth minus 2 to be divisible by 6. In detail, we use ResNet[c]-56 and ResNet[c]-110. Besides, we apply the standard ResNet-18 on CIFAR-10 additionally. For ImageNet, we directly apply the standard ResNet-50. For for Tiny-ImageNet, we adopt the specific ResNet32 with double the number of filters in each convolutional layer, same as the setting in a recent improved variants of LTH [17], we named ResNet[c]-32++ for short.

Training. We use Pytorch[18] to implement all experiments and follow hyperparameters identically in [3]. Standard data augmentation was applied to all datasets during the training time. We use Stochastic Gradient Descent (SGD) to train all networks. All hyperparameters and number of parameters for each network are described in Table 1.

Pruning. In our experiments, we only consider magnitude-based unstructured pruning, which prunes the lowest global magnitude weights throughout the network. Specifically, we use the one-shot magnitude pruning (OMP), as prunes all at once after training. For the pruning ratio, we prune to the same sparsity level as the setting in [19].

Knowledge distillation. The logits produced by the dense network are considered as the teacher's knowledge. For all experiments on CIFAR-10, the weight α and temperature τ in Equation 2 are set to 0.9 and 5, respectively. For Tiny-ImageNet, we follow the recent KD experiments on it [20] and set α and τ to 0.9 and 20, respectively. For ImageNet, to the best of our knowledge, there are only a few studies report results of KD on it [21, 22, 23]. [21] found that the standard setting of KD fails on ImageNet and even performs worse than training from scratch. As a reason for this, [23] hypothesized that student models may underfit on this challenging dataset. Additionally, [23] demonstrates that stopping the knowledge distillation early in the training process, while only applying the cross-entropy loss for the rest training phase, serves as a remedy for the worse performance of KD. Thus, for ImageNet, we set $\alpha = 0.9$ and $\tau = 4$ as the same in [21, 23]. The Early-Stopped Knowledge Distillation ("ESKD") technique in [23] are also applied in our experiments.

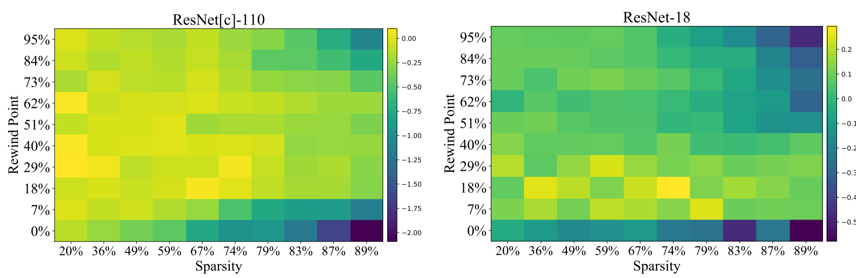
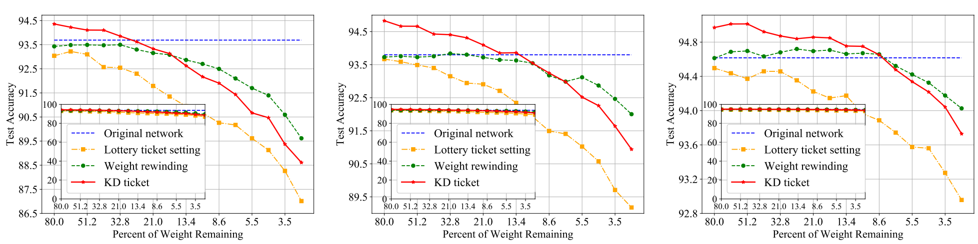


Figure 2: Rewind point and sparsity. The value indicates the difference between the performance of original network and pruned network. The yellower or brighter a cell is, the better performance it has. A yellower cell means the pruned network at its specific sparsity and rewind point can achieve or even outperform the unpruned dense model.



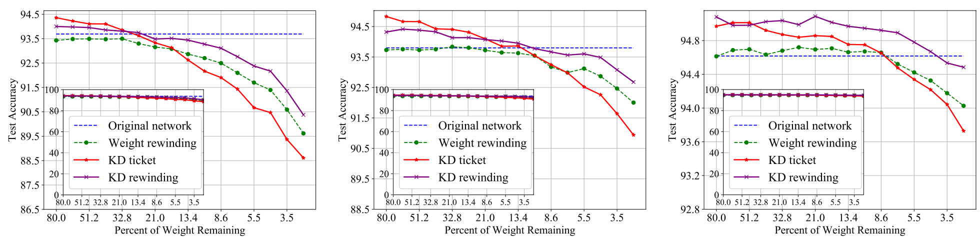
(a) ResNet[c]-56 (b) ResNet[c]-110 (c) ResNet-18

Figure 3: Pruning sparsity and accuracy of KD tickets and weight rewinding on CIFAR-10. All experiments are using OMP.

3.2 Compare KD Tickets with Weight Rewinding

In this section, we study the performance of knowledge distillation tickets and early weight rewinding on CIFAR-10. We find that our KD tickets can achieve or even outperform weight rewinding in almost all scenarios. We also conduct experiments that rewinding to the initial weights (the lottery ticket setting), and treat them as our baseline.

Rewind Point. Apply weight rewinding requires to first pick the specific rewinding training point. From the studies in [2, 3], a rewind point at the very beginning of training usually results in lower



(a) ResNet[c]-56 (b) ResNet[c]-110 (c) ResNet-18

Figure 4: Pruning sparsity and accuracy of the KD rewinding on CIFAR-10. All experiments are using OMP. The curve of weight rewinding and KD tickets are same as Figure 3.

accuracy than a rewind point at a small amount into training. From experiments in [3], an early training point at 18% of the original training time is a good choice for ResNet[c]-56 on CIFAR-10. For the ResNet[c]-110 and standard ResNet-18, to quantitatively determine where to rewind to, we perform abundant experiments to explore the relationship of rewind points and sparsity levels. As the specific rewind points, we follow the setting in [3]. We use a heatmap to represent the result of our experiments succinctly. In Figure 2, the color of each cell indicates the difference between the accuracy of our original network and the accuracy of a pruned network at a certain sparsity level and rewind point. The yellower the cell is, the higher accuracy the pruned network has. We aim at picking up a rewind point whose cells at all sparsity levels are yellower than others.

The heatmaps in Figure 2 are consistent with the conclusion that the best rewind point usually lies in the early training phase(e.g 18% or 29%), as cells at this rewind point are yellower and brighter than others. Specifically, the overall best rewind points for ResNet[c]-110 and the standard ResNet-18 are also around 18%. Thus, we consider 18% training point as the best weight rewinding we compare.

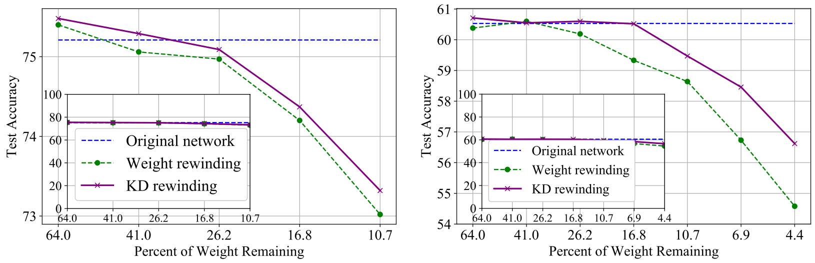
Results. Figure 3 presents the results of experiments on both weight rewinding and KD tickets. The solid red lines of our KD tickets are above the solid green lines of weight rewinding or coincide with green lines in several networks. Both of them outperform the lottery ticket setting, which is consistent with the conclusion that the LTH is challenged in deeper network. This indicates that our proposed KD tickets can serve as a competitive substitute for the weight rewinding, without any trail-and-error effort in finding the specific rewind point.

Furthermore, to compare between KD tickets and rewinding, we name the sparsity level where the line of KD tickets(red) and weight rewinding(green) cross with each other as the *critical sparsity level*. This critical sparsity level reflects the critical point that the performance of KD tickets can match weight rewinding exactly. At sparsity levels lower than the critical level, KD tickets outperforms the weight rewinding. Figure 3 shows that with lower sparsity, there is more differentiation between KD tickets and weight rewinding. While at higher sparsity levels, our KD tickets performs marginally worse than the weight rewinding. Moreover, the results of several networks show that models with different parameters generally have different critical sparsity levels. We notice that the larger depth or more parameters the original network has, the higher critical level it exhibits in general. This indicates that when the parameter capacity of network is relatively large, the soft labels would already provide sufficient information for the pruned network to inherit, while the early weights may not.

3.3 Combine KD Tickets with Weight Rewinding: New State-of-the-Art

In this section, we study the combination of KD tickets and weight rewinding, the KD rewinding. The logits only maintain the probability distribution of each training data, which is just a high level knowledge of the unpruned network. While the weights at early training points preserve more detailed information of the original training process and made it possible to find trainable sub-networks. To fully derive the original training procedure, we perform experiments to combine both our KD tickets and the weight rewinding techniques together. In detail, we rewind the remaining weights of pruned network to the specific early training point (18% in our experiments), and retrain the pruned subnetwork with the soft labels of the original network.

Results. As the purple line in Figure 4, the overall performance of the KD rewinding is better than either of them. Furthermore, we notice that differences between the combination and single technique vary at different sparsity. At the lower sparsity level, the KD rewinding perform almost the same as the KD ticket, while at a higher sparsity level, the KD rewinding achieve better performance than either of them. Moreover, the combination of KD ticket and weight rewinding has the ability to maintain the original accuracy at an extremely sparsity level.



(a) ImageNet (b) Tiny-ImageNet

Figure 5: KD rewinding on ImageNet and Tiny-ImageNet

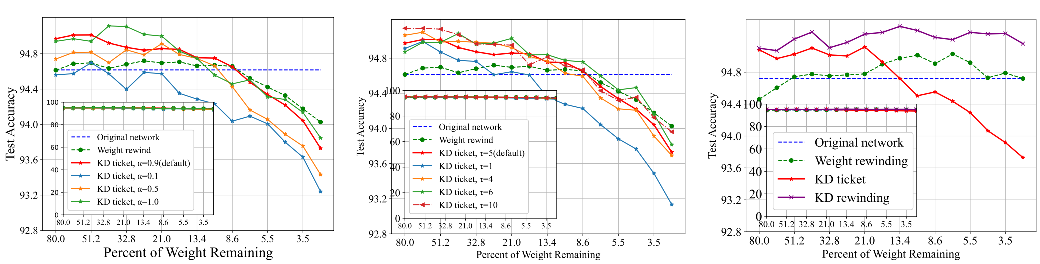
Discussion. The success of KD rewinding proves that the knowledge inherited from the soft labels and from the early weights are complementary rather than conflicting. With the two different cues combined, we can recover the accuracy of unpruned network at an extremely high sparsity and achieve state-of-the-art performance in that pruning ratio. Moreover, the low/high sparsity difference also echos our hypothesis. At low sparsity levels, the soft labels already carry enough information to regain the performance, making the early weights unnecessary. However, at higher compression ratios, the KD tickets turns relatively weaker, and the early weights supply extra information to maintain the accuracy.

3.4 KD rewinding for ImageNet and Tiny-ImageNet

After successfully demonstrating our methods's abilities on the small CIFAR-10 dataset, we move on to examine its effectiveness on the large-scale ImageNet dataset and its subset Tiny-ImageNet. For the specific early rewind position, we stay consistent with previous findings and set it to epoch 6 for ImageNet[2] and epoch 5 for Tiny-ImageNet[17].

Results. From Figure 5, we notice that KD rewinding still outperforms weight rewinding at all sparsity levels on both ImageNet and Tiny-ImageNet, especially on higher compression ratios. This results indicate the generalization ability of KD rewinding technique. We acknowledge that the improvement on ImageNet is relatively marginal. However, similar as previous explanation for the failure of KD on ImageNet[23] , we hypothesis that the subnetwork may be in the underfitting regime as ImageNet is a challenging dataset. Thus it may not have enough capacity to minimize both the training loss and the KD loss. Nevertheless, the success on the relatively easier Tiny-ImageNet can support our claim to some degree.

3.5 Ablation study



(a) ablation study on weight α (b) ablation study on temperature τ (c) KD tickets with IMP

Figure 6: Results of ablation studies

Hyperparameters We carry out ablation studies on CIFAR-10 with the standard ResNet-18 to explore the effects of hyperparameters in KD tickets, specifically, the coefficient α and the temperature τ. By default, we set $\tau = 5$ and $\alpha = 0.9$, as the popular setting in [11, 23]. For temperature τ, we consider values around default setting ($\tau = 4$ and $\tau = 6$) and extreme values ($\tau = 1$ and $\tau = 10$). For α, we consider $\alpha = 0.1$, $\alpha = 0.5$ and $\alpha = 1.0$. From Figure 6, we notices that for both the coefficient α and the temperature τ, as long as we use values around the default values, our proposed KD tickets can match or outperform the performance of weight rewinding, unless choosing extreme value that we will never use in practice (e.g. $\alpha = 0.1$ and $\tau = 1$). Therefore, KD ticket is quite robust to the hyperparameter choices. Practically, we find no need to tune them across experiments, and simply stick to our default ones.

Pruning method Our experiments mainly focus on OMP as it is more efficient. To prove its effectiveness, we also conduct experiments on ResNet18 with Iterative Magnitude Pruning (IMP). From Figure 6(c), we can observe that KD ticket is still equivalent to weight rewinding with IMP, and KD rewinding always outperforms weight rewinding at all sparsity levels, as our conclusion on experiments with OMP. Thus, our method is also effective on IMP.

4 Related Work

4.1 Model compression

Recent success of deep neural network [24, 15, 16] has been coupled with considerable computational resources. In particular, the storage size, run-time memory, and number of computing operations mainly constrain the CNNs towards the real-word deployment, such as mobile devices, or Internet of Things (IoT) devices. In general, there are two major types of network pruning techniques. One major class of network pruning methods is *weight* pruning [7, 25, 8, 10, 26, 27]. Han et al. [8] proposes to prune redundant weights in trained neural networks, and then retrain the sparse network in the manner that attain the network performance. Louizos et al. [27] introduces a practical method to learn a sparse network with the L_0 norm regularization through non-negative stochastic gates. Another main class is *structured* pruning in the way of pruning at the level of channels or even layers [9, 28, 29, 30, 31, 32, 33]. Li et al. [9] proposes to prune filters with relatively small incoming weights in the trained network, and then fine-tune the network without hurting original accuracy. Wen et al. [29] imposes sparsity constraints on channel-wise scaling factors to regularize different structures (e.g. filter, depth, channel) in CNNs. Recent works [34, 11] on knowledge distillation demonstrate a compressed or small model can mimic the behavior of a more complex or deeper model by learning from the soft output of the complicated large one. It follows a student-teacher paradigm to train smaller, cheaper models by forcing student models to imbibe some knowledge from the teacher ones and generalize beyond the information expressed in the training labels.

4.2 The Lottery Ticket Hypothesis

The lottery ticket hypothesis (LTH) [1] argues that over-parameterized neural networks contain sparse, trainable sub-networks which can achieve a better performance when trained with original random initialization. In addition to supervised image classification, LTH has been explored widely in numerous contexts, such as natural language processing, reinforcement learning[35, 36] and transfer learning [37, 38]. Scaling up LTH to larger networks was achieved with the “late rewinding" technique [19, 3]. This kind of rewinding claim that rewinding the weights to their values at some early iteration in the unpruned model training, instead of to their original initialization, is capable of achieving better performance on large networks, which demonstrates the changes in the early stage of network training plays a vital role in the success of LTH. However, choosing an appropriate rewinding position is ad hoc, for which the expensive trial-and-error search is often needed. Our proposal resolves this dilemma through utilizing knowledge distillation to inherit important training information of unpruned models, while being almost free of hyperparameters.

5 Conclusion

On the basis of the recent lottery ticket hypothesis, we propose "Knowledge Distillation Ticket" (KD ticket) for improving lottery ticket finding on large-scale networks. Soft-labels of the dense teacher model is considered as an extra data-level hint, and the sparse lottery sub-network is served as the student model to inherit the late knowledge from the past training. In the experiments, without bells and whistles, we demonstrate that our KD ticket outperforms the original winning tickets, and achieves similar or better performance compared to "weight winding". Furthermore, we show that our KD tickets are compatible with "weight rewinding" and the performance of resultant "KD rewinding" achieves state-of-the-art performance. The extensive experiments have justified the efficacy of our approaches. In future work, we will study other pruning approaches, including structure pruning, iterative pruning, etc. Other more complicated tasks beyond computer vision areas will also be considered.

References

[1] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[2] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

[3] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[5] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[6] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *PAMI*, 2017.

[7] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[8] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

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

[10] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11669–11680, 2019.

[13] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020.

[14] Guillaume Leclerc and Aleksander Madry. The two regimes of deep network training. *arXiv preprint arXiv:2002.10376*, 2020.

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

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. *arXiv preprint arXiv:1912.05671*, 2019.

[20] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.

[21] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[22] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[23] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4794–4802, 2019.

[24] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[25] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.

[26] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[27] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018.

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

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

[30] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.

[31] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[32] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016.

[33] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pages 2181–2191, 2017.

[34] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[35] Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. *arXiv preprint arXiv:1906.02768*, 2019.

[36] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[37] Rahul Mehta. Sparse transfer learning via winning lottery tickets, 2019.

[38] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems*, pages 4933–4943, 2019.