DUAL LOTTERY TICKET HYPOTHESIS

Anonymous authors
Paper under double-blind review

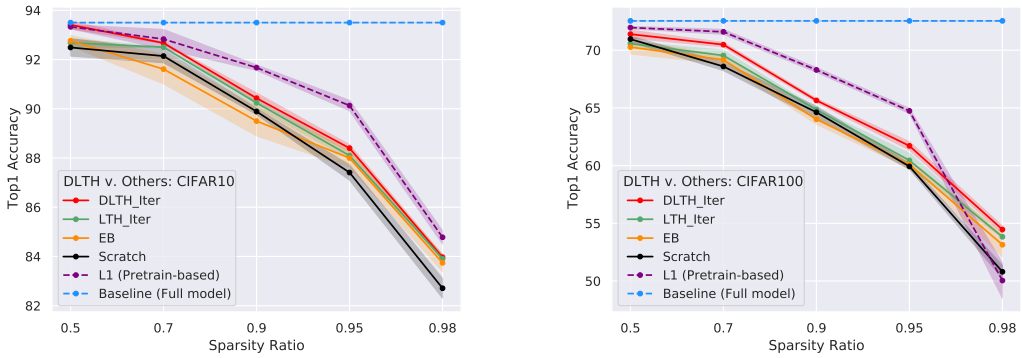
ABSTRACT

Fully exploiting the learning capacity of neural networks requires overparameterized dense networks. On the other side, directly training sparse neural networks typically results in low learning capacity with unsatisfactory performance. Lottery Ticket Hypothesis (LTH) provides a novel view to investigate sparse network training. Concretely, it claims there exist winning tickets from a randomly initialized network found by iterative magnitude pruning. In this work, we go from a complementary direction and articulate the *Dual Lottery Ticket Hypothesis (DLTH): Any randomly selected subnetwork of a randomly initialized dense network can be transformed into an appropriate condition with admirable trainability* — any ticket in a given lottery pool can be transformed into a winning ticket. We propose a simple sparse network training strategy, Random Sparse Network Transformation (RST), to substantiate our DLTH. Specifically, we introduce a regularization term to borrow learning capacity and realize information extrusion from the weights which will be masked. Extensive experiments on several public datasets and comparisons with competitive approaches validate our DLTH as well as the effectiveness of the proposed model RST. Our work is expected to pave a way for inspiring new research directions of sparse network training in the future. Our code will be public online.

1 INTRODUCTION

While over-parameterized networks perform promisingly on challenging machine learning tasks Zagoruyko & Komodakis (2016); Arora et al. (2019); Zhang et al. (2019), they require a high cost of computational and storage resources Wang et al. (2020a); Cheng et al. (2017); Deng et al. (2020). Recent pruning techniques aim to diminish the model size by discarding irrelevant weights of well-trained models based on different criteria Gale et al. (2019); He et al. (2017); Han et al. (2015a;b). Decisive weights are preserved and finetuned for obtaining final compressed model with acceptable performance loss. Following this line, several series of research works have been done to explore effective pruning criterion for better pruning performances. For example, regularization based pruning approaches Liu et al. (2017); Han et al. (2015a;b) leverage a penalty term during training for network pruning. Also, many researches take advantages of Hessian information to build more proper criteria for pruning LeCun et al. (1990); Hassibi & Stork (1993); Wang et al. (2019a); Singh & Alistarh (2020). However, regular pruning methods still requires full pretraining with high computational and storage costs. Pruning at initialization (PI) attempts determining the sparse network before training and maintains the final performances. For instance, Single-shot Network Pruning (SNIP) Lee et al. (2018) uses a novel criterion called connection sensitivity to measure the weights importance and decide which weights should be removed. Gradient Signal Preservation (GraSP) Wang et al. (2020a) regards the gradient flow as an importance factor and correspondingly use it to design PI criterion.

Regular pruning and PI techniques both achieve promising results on sparse network training and model compression. They only focus on designating criteria to find a specific sparse network but ignore exploring the internal relationships between the dense network and its eligible subnetwork candidates, which impedes a full understanding of the sparse network training. Lottery Ticket Hypothesis (LTH) Frankle & Carbin (2018) hits this problem with the first shot. It claims a trainable subnetwork exists in the randomly initialized dense network and can be found by pruning a pretrained network. In other words, as long as the dense network is initialized, the good subnetwork has been implicitly decided but needs to be revealed by the pruning process. The LTH has been



(a) DLTH v. Others on CIFAR10. (b) DLTH v. Others on CIFAR100.

Figure 1: Comparisons between DLTH with other training strategies. 3 times average scores with standard deviation shown as shadow are plotted. Both (a)/(b) (CIFAR10/CIFAR100) demonstrate ours superiority in all non-pretrain based approaches (solid lines). Two pretrain based methods (dash lines) are plotted for reference. (This figure is best viewed in color.)

validated by training the subnetwork from scratch with mask obtained from corresponding iterative magnitude pruning Han et al. (2015b;a). The subnetwork is regarded as the winning ticket among the lottery pool given by dense networks. This valuable discovery naturally describes the relationship between a random initialized dense network and the trainable subnetwork hidden in it.

However, LTH only focuses on finding one sparse structure at the expense of full pretraining, which is not universal to both practical usage and investigating the relationship between dense and its subnetworks for sparse network training. In this work, we go from a complementary direction of LTH and propose a new hypothesis called Dual Lottery Ticket Hypothesis (DLTH). It studies a more challenging and general case and achieves promising performances (see Fig. 1) for sparse network training. DLTH is described as *any random selected subnetwork of a randomly initialized dense network can be transformed into a trainable condition.* In other word, any ticket in a given lottery pool can be turned into a winning ticket. In this way, we construct a closed-loop research perspective from the dual side of LTH as shown in Fig. 2. Compared with LTH, our proposed DLTH considers a general case - studying any subnetwork instead of a specific one – with a simple training strategy proposed to validate it. The comprehensive relationship between dense and sparse networks is inquired for sparse network training. Our contributions can be summarized as follows:

- We investigate a novel view of studying sparse network training. Specifically, we present a *Dual Lottery Ticket Hypothesis (DLTH)*, a dual problem of Lottery Ticket Hypothesis (LTH), articulated as: *Any randomly selected subnetwork of a randomly initialized dense network can be transformed into an appropriate condition with admirable trainability.*

- We propose a simple sparse network training strategy, Random Sparse Network Transformation (RST), to validate our proposed DLTH and achieve promising performance for sparse network optimization. Concretely, we achieve the information extrusion from the weights which will be removed to enhance sparse training by introducing a regularization term. Any subnetwork will be transformed into a trainable condition in this way.

- Extensive experiments are conducted based on benchmark datasets and comparisons with competitive approaches. Our method obtains promising and consistent performance which solidly validates our DLTH and demonstrates the model effectiveness.

- Our DLTH considers a more general and challenging case for sparse network training compared with LTH. In this way, we expect our work inspires a novel perspective to study sparse network training in a more flexible and adjustable way.

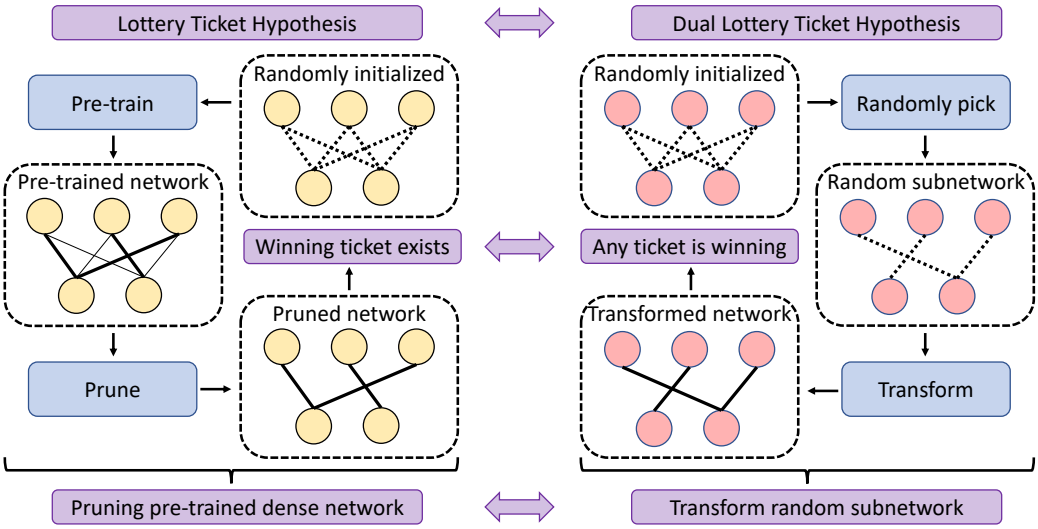


Figure 2: Diagram of Lottery Ticket Hypothesis (LTH) and Dual Lottery Ticket Hypothesis (DLTH).

2 Related Work

2.1 Sparse Network Training

Sparse Evolutionary Training (SET) Mocanu et al. (2018) proposes an evolutionary strategy to randomly add weights on sparse network and achieve better training. Dynamic Sparse Reparameterization (DSR) Mostafa & Wang (2019) presents a novel direction to dynamically modify the parameter budget between different layers. In this way, the sparse structure can be adaptively adjusted for more effective and efficient usage. Sparse Networks from Scratch (SNFS) Dettmers & Zettlemoyer (2019) designates a momentum based approach as criterion to grow weights and empirically proves it benefits the practical learning performance. Deep Rewiring (DeepR) Bellec et al. (2017) introduces the sparsity during the training process and augments the regular SGD optimization by involving a random walk in the parameter space. It can achieve effectively training on very sparse network relying on theoretical foundations. Rigging the Lottery (RigL) Evci et al. (2020) enhances the sparse network training by editing the network connectivity along with the optimizing the parameter by taking advantages of both weight magnitude and gradient information.

2.2 Network Pruning

Network pruning is a relevant research topic to our work. Related approaches aim to remove unnecessary weights based on different well-designed criteria and obtain sparse network for model compression. Diversified pruning methods can be categorized from different angles. To better present the background of our work, we customize the existing pruning methods into *Training-Relevant Pruning* and *Training-Free Pruning* groups as follow.

Training-Relevant Pruning. Most pruning methods require a pre-trained network Louizos et al. (2017); Liu et al. (2017); Ye et al. (2018); Han et al. (2015b;a). Algorithms utilize different ranking strategies to pick redundant weights with low importance scores, and remove them to achieve pruning along with acceptable performance drop. Pioneer magnitude-based method Han et al. (2015b;a) regards weights with low values as unnecessary ones, which is straightforward but may remove important low-value weights. Hessian-based approaches measure the weight importance by computing the their removal effects on the final loss LeCun et al. (1990); Hassibi & Stork (1993). Recently published technique also achieves Hessian-free pruning Wang et al. (2020b) by adding regularization which is more computational friendly. On the other hand, pruning process can be also conducted during the network training instead of relying on completed pre-trained network. To name a few, a special dropout strategy is utilized in Srinivas & Babu (2016) to adjust the dropout during training and obtain a pruned network after training. A L_0 norm regularization based method is proposed for network pruning Louizos et al. (2017). Above algorithms requiring network training are seen as *Training-Relevant Pruning*.

Table 1: Comparisons between DLTH and other settings. `Base model`, `One/All`, `Controllability`, `Transformation`, and `Pretrain` represent where to pick subnetwork, if the setting finds one specific subnetwork or studies all subnetworks, if sparse structure is controllable, if the selected subnetwork needs transformation before finetuning, and if pretraining dense network is needed, respectively.

Settings	Base model	One/All	Controllability	Transformation	Pretrain
Conventional	Pretrained	One	Uncontrollable	No	Yes
LTH	Initialized	One	Uncontrollable	No	Yes
PI	Initialized	One	Uncontrollable	No	No
DLTH (ours)	Initialized	All	Controllable	Yes	No

Training-Free Pruning. This group of research investigates how to prune randomly initialized network without any training. *Pruning at initialization* is one typical direction deriving sparse network by remove part of randomly initialized weights Lee et al. (2018; 2019); Wang et al. (2020a). For instance, Single-Shot Neural Network Pruning (SNIP) algorithm Lee et al. (2018) first proposes a learnable sparse network strategy according to the computed connection sensitivity. An orthogonal initialization method is proposed to investigate pruning problem in signal propagation view Lee et al. (2019). Gradient Signal Preservation (GraSP) considers to preserve the gradient flow as an efficient criterion to achieve pruning at initialization. On the other hand, Lottery Ticket Hypothesis (LTH) claims that the *winning ticket* subnetworks exist in randomly initialized network and can be found by deploying conventional pruning algorithm. Then the selected sparse network can be efficiently trained from scratch and achieve promising performance. Based on LTH, Early-Bird (EB) ticket You et al. (2020) proposes an efficient way to find winning ticket. Above algorithms obtaining sparse network from random network are seen as *Training-Free Pruning*.

3 Lottery Ticket Perspective of Sparse Network

Lottery Ticket Hypothesis (LTH) Frankle & Carbin (2018) articulates the hypothesis: *dense randomly initialized networks contain subnetworks, which can be trained in isolation and deliver performances comparable to the original network*. These subnetworks are treated as *winning tickets* in the given lottery pool with good trainability. Subnetworks are discovered by iterative magnitude pruning algorithms, which requires training a full network and removes redundant weights by specific criterion. The mask of the pruned network illustrates the sparse structure of winning tickets.

Problem Formulation. We start from introducing general network pruning problem. The neural network training process can be seen as a sequence of parameter updating status using stochastic gradient descent (SGD) Wang et al. (2021); Bottou (2010):

$$\{w^{(0)}, w^{(1)}, w^{(2)}, \cdots, w^{(k)}, \cdots, w^{(K)}\}, \tag{1}$$

where w is the model parameter with superscript k as training iteration number. For a general case, the sparse network structure can be mathematically described by a binary mask m which has the same tensor shape as w. The process of obtaining m can be formulated as a function $m = F_m(w; D)$, where D is the training data. Further, the weights of the sparse structure are modified based on different pruning strategies Hassibi & Stork (1993); Wang et al. (2020b), which is given by $w^* = F_w(w; D)$. The final pruned network can be integrated as

$$\widetilde{w} = F_m(w^{(k_m)}; D) \cdot F_w(w^{(k_w)}; D), \tag{2}$$

where w^{k_m} and w^{k_w} represent different parameter conditions for F_m and F_w. The conventional pruning requires that $k_m = k_w = K$. The LTH needs $k_m = K, k_w = 0$, and $F_w = I$, where I is the identical mapping representing model directly inherits the randomly initialized weights.

Against with traditional view that directly training sparse network cannot fully exploit network capacity Wang et al. (2020b;a), LTH validates there exists sparse network with better trainability (winning ticket) than other subnetworks (other tickets). However, to uncover this property still needs first using pruning techniques for obtaining mask and the selected sparse structure must match with corresponding randomly initialized dense network - *winning ticket matches with the given lottery pool.* LTH provides a novel angle to understand and reveal the connections between a randomly initialized dense network and its subnetworks with admirable tranability. However, the proposed hypothesis is valid with restrict constraint and finding subnetworks still requires pruning on pre-trained model.

4 DUAL LOTTERY TICKET HYPOTHESIS (DLTH)

In our work, we follow the perspective of LTH and investigate the sparse network training from an complementary direction. We articulate the *Dual Lottery Ticket Hypothesis (DLTH)*. Further, we develop a simple training strategy to valid our DLTH and achieve promising sparse training performance. In this section, we first define our hypothesis and discuss its connections and differences with LTH and other related research topics.

Dual Lottery Ticket Hypothesis (DLTH). *Any randomly picked subnetwork in a dense randomly initialized neural network can be transformed into a trainable condition, where the subnetwork can be trained in isolation and achieves better at least comparable performance to LTH and other strong baselines.*

Formally, we follow the general definition provided in Sec. 3 for problem formulation. Our DLTH requires $k_m = 0, k_w = 0$ and allows F_m being random mapping. F_w represents our proposed random sparse network transformation (RST) process which will be detailed introduced in Sec. 5

Comparisons with LTH. LTH focuses on proving there *exist* appropriate subnetworks in the randomly initialized dense network whose weights can be easily trained and overcome the difficulties of training sparse network. The sparse structure is picked from the *initial network* but requires *pruning pre-trained* dense network to obtain mask. On the contrary, our DLTH claims *any* subnetworks can be *transformed* into a proper condition *without pruning* where we can freely define the mask and further conduct efficient sparse network training. Mathematically, these two statements form a complementary dual statements of sparse network training. Thus, we name our work as DLTH. We clarify the advantages of our work in detail from three angles below: 1) Although LTH and DLTH are *twins* in mathematics, our hypothesis considers a more general scenario (*exist subnetwork versus any subnetwork*) which studies the common case academically and derives more potential of practical usage; 2) Our goal is more valuable: transferring random subnetworks into trainable condition (*making random ticket to winning ticket*) versus pretraining dense network to identify the good subnetwork (*finding one specific winning ticket from all random tickets*). Our transformation process is also more efficient than pretraining a dense network. Our training efficiency will be discussed in Sec. 6; 3) Our work allows flexible sparse network selection but LTH must inherit the mask from pruned pre-trained network and the sparse structures are not controllable.

Comparisons with Pruning at Initialization (PI). Pruning at initialization (PI) is another relevant research topic. Instead of pruning pre-trained network, PI algorithms select sparse network from scratch according to well-designed criterion or dynamic selection process. In this way, the subnetworks with better trainability are picked for following finetuning. Unlike LTH finding sparse candidate through conventional pruning, PI approaches choose candidates without training dense network. Following our definition in Sec. 3, PI requires $k_m = 0, k_w = 0$. F_m and F_w is determined by different algorithms. Similarly, their goal is still obtaining a specific sparse network according to a selection process and not controllable. Our work is not for choosing one specific subnetwork but transferring any subnetwork into a good condition. In this way, the sparse structure will be flexible and under our control. We include the related pruning settings and summarize the comparisons between them and our work in Tab. 1.

5 RANDOM SPARSE NETWORK TRANSFORMATION (RST)

Being analogous to the real-world lottery game where the winning ticket exists in the lottery pool but hard to find, randomly picking subnetworks from random initialization is not wise and need to be manipulated leveraging on the existing information from other parts of network (other tickets). We follow this analogy to propose a simple training strategy, Random Sparse Network Transformation (RST), to obtain trainable sparse network. Specifically, we utilize a widely-used regularization term Wang et al. (2020b; 2019b) to borrow information from other weights. It benefits the sparse network training by extruding information from other weights to target sparse structure.

Regularization Based Information Extrusion. We use a regularization term for information extrusion. Given a dense random network $f(x; \theta)$ with a random picked subnetwork represented by

Table 2: Test accuracy of ResNet56 on CIFAR10 and CIFAR100 datasets.

ResNet56 + CIFAR10: Baseline accuracy: 93.50%					
Pruning ratio	50%	70%	90%	95%	98%
L1 Li et al. (2016) (Pretrain-based)	93.33±0.08	92.83±0.39	91.67±0.11	90.13±0.21	84.78±0.27
LTH Frankle & Carbin (2018)	92.67±0.25	91.88±0.35	89.78±0.35	88.05±0.50	83.85±0.55
LTH Iter-5 Frankle & Carbin (2018)	92.68±0.39	92.50±0.15	90.24±0.27	88.10±0.36	83.91±0.15
EB You et al. (2020)	92.76±0.21	91.61±0.60	89.50±0.60	88.00±0.38	83.74±0.35
Scratch	92.49±0.35	92.14±0.27	89.89±0.12	87.41±0.31	82.71±0.40
RST (**ours**)	92.34±0.12	92.27±0.24	90.41±0.05	88.24±0.08	83.77±0.47
RST Iter-5 (**ours**)	93.41±0.16	92.67±0.02	90.43±0.21	88.40±0.14	83.97±0.09
ResNet56 + CIFAR100: Baseline accuracy: 72.54%					
Pruning ratio	50%	70%	90%	95%	98%
L1 Li et al. (2016) (Pretrain-based)	71.96±0.10	71.59±0.21	68.29±0.20	64.74±0.22	50.04±1.52
LTH Frankle & Carbin (2018)	69.95±0.47	68.24±0.60	65.66±0.47	60.97±0.30	52.77±0.44
LTH Iter-5 Frankle & Carbin (2018)	70.57±0.15	69.54±0.46	64.84±0.11	60.45±0.61	53.83±0.09
EB You et al. (2020)	70.27±0.59	69.16±0.36	64.01±0.42	60.09±0.33	53.14±1.04
Scratch	70.96±0.25	68.59±0.35	64.62±0.52	59.93±0.24	50.80±0.55
RST (**ours**)	71.13±0.48	69.85±0.23	66.17±0.18	61.66±0.37	54.11±0.37
RST Iter-5 (**ours**)	71.39±0.34	70.48±0.19	65.65±0.15	61.71±0.36	54.46±0.32

mask $m \in \{0, 1\}$, the information extrusion can be formulated as optimizing following loss:

$$\mathcal{L}_R = \mathcal{L}(f(x; \theta), \mathcal{D}) + \frac{1}{2}\lambda \|\theta^*\|_2^2, \tag{3}$$

where $\mathcal{L}_R$ is the regularization based loss containing a regular training loss $\mathcal{L}$ on given data $\mathcal{D}$ and a L_2 regularization term added on θ^*. θ^* is the part of parameter θ which being masked by $m = 0$. In this way, the unmasked weights $\overline{\theta^*}$ are regularly trained by SGD, while the masked weights θ^* are still involved for network optimization but their magnitudes are limited via the regularization. The trade-off parameter λ is set to gradually increase from a small value during optimizing $\mathcal{L}_R$:

$$\lambda_{p+1} = \begin{cases} \lambda_p + \eta, & \lambda_p < \lambda_b, \\ \lambda_p, & \lambda_p = \lambda_b, \end{cases} \tag{4}$$

where λ_p is the regularization term at p-th updating. η is the given mini-step for gradually increasing λ value. λ_b is the bound to limit the λ increasing. In this way, the magnitude of θ^* is diminished from a regular to a small scale along with training simultaneously. The above dynamic process adjusts the importance of weight from balanced (equal for all weights) to imbalanced (θ^* becoming less important with diminishing magnitude). The information of θ^* is gradually extruded into $\overline{\theta^*}$. Analogically, *the random ticket is transformed to winning ticket by leveraging information from other tickets in the whole lottery pool.* After sufficient extrusion, the θ^* will have very limited effect on the network and we remove them to obtain the sparse network.

6 Experimental Validation

To validate our proposed Dual Lottery Ticket Hypothesis (DLTH), we conduct extensive experiments on several datasets compared with Lottery Ticket Hypothesis (LTH) Frankle & Carbin (2018) and other strong baselines with other experimental analysis.

Datasets and Backbone Models. We first conduct experiments compared with LTH and other baselines on CIFAR10, CIFAR100 Krizhevsky et al. (2009), and a subset of ImageNet Deng et al. (2009) datasets using ResNet56 and ResNet18 as backbones He et al. (2016).

Comparison Approaches We compare our RST with several approaches: `L1` Li et al. (2016) is the regular L1 pruning based on pretrained network. `LTH` Frankle & Carbin (2018) is the lottery ticket performance based on one-shot pruning strategy. `LTH-Iter` follows the iterative magnitude pruning strategy used in LTH to evaluate sparse network performances. `EB` You et al. (2020) is the Early-Bird ticket for LTH with one-shot pruning strategy. `Scratch` represents training a random subnetwork from scratch without pretraining. `RST` and `RST-Iter` are our training strategy in one-shot way and iterative way, respectively.

Evaluation Settings. We follow consistent experimental settings for different comparison methods to make fair comparisons. We set five different sparsity ratios: 50%, 70%, 90%, 95%, and 98%. Specifically, we leave the first convolutional layer and the last dense layer as complete, and prune the weights in other middle layers with the consistent sparsity ratio. We report their 3 time runs average top1 classification accuracy and standard deviation. Due to the different training strategies for different methods (see Tab. 1), their experimental configurations have slightly differences which will be detailedly discussed and compared in Sec. 6.1

Implementations. All experiments follow the same fine-tuning schedule for fairness. Specifically, experiments on CIFAR10 and CIFAR100 are optimized by SGD with 0.9 momentum and 5e-4 weight decay using 128 batch size. Total number of epochs is 200 with 0.1, 0.01, 0.001 learning rates starting at 0, 100, and 150 epochs, respectively. Those for ImageNet are optimized by SGD with 0.9 momentum and 1e-4 weight decay using 256 batch size. Total number of epochs is 90 with 0.1, 0.01, 0.001, 0.0001 learning rates starting at 0, 30, 60, and 75 epochs, respectively.

For approaches using pretrained model, we use the same training schedule as finetuning to pretrain the network for L1 and LTH. We set the total number of epochs as 50 with 0.1, 0.01, 0.001 learning rates starting at 0, 25, 37 epochs, respectively, for mini-pretraining in LTH-Iter setting. Other configurations are kept as the same. Following the original paper of EB You et al. (2020), we set the early stop point at 25 epochs (1/8 of total epochs) and keep other settings as the same for EB.

For our RST strategy, we set the start point λ_0 and bound λ_b as 0 and 1, respectively. The mini-step η is set as 1e-4 and the λ is increased every 5 iterations termed as v_η. Each iteration optimizes the parameter with 64 batch size and 1e-3 learning rate. After λ reaches the bound, we continue the extrusion with 40000 iterations for model stability termed as v_s. For RST-Iter, in order to keep fairness with LTH-Iter, we accordingly diminish the transformation process by setting λ being increased every 1 iterations and 10000 iterations for model stability. More detailed comparisons are provided in Sec. 6.1. Since our transformation uses 1e-3 learning rate, we add a small warm-up slot in finetuning for our methods: 10 epochs and 4 epochs with 1e-2 learning rate, for CIFAR10/CIFAR100 and ImageNet, respectively, at the beginning then back to 0.1 as mentioned above.

6.1 Baseline Comparisons

Tab. 2 shows the comparisons on CIFAR10 and CIFAR100. The full accuracy (first line) is trained by regular SGD optimization on random dense network. For consistency, L1, LTH, LTH-Iter, and EB inherit the same random dense network as the full accuracy model. Scratch, RST, and RST-Iter use the same random dense network as above and they also share the same randomly selected sparse network from the dense one. L1 is the only one requires pretrained model and others start with random dense network. Note that, once the sparse subnetwork is randomly decided, it will be kept intact without any structural modification. Since L1 method prunes the pretrained network and finetunes it, it should have the highest performance. Thus, we exclude the L1 row and emphasize the rest rows using red/blue fonts as the highest/second highest performances.

Iterative Cycle Number Choice. We use CIFAR10 with pruning ratio 0.7 as an examplar to set the iterative cycles. In Fig. 3, we run 3 times and plot the average with shadow area as standard deviation. LTH performs better using more iterative cycles; our DLTH takes less advantages of more cycles but still outperforms LTH. To fully experimentally demonstrate our superiority, we set the cycle number as 5 (the best choice for LTH). The DLTH cycle number ablation study is provided in Fig. 5.

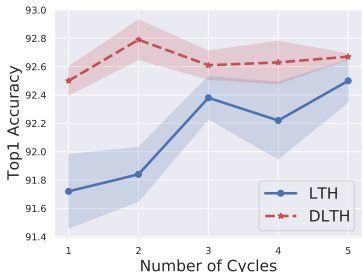


Figure 3: LTH v. DLTH on CIFAR10 with 0.7 sparsity to choose cycle number.

Performance Analysis. From the results in Tab. 2, we draw the following conclusions: 1) Our method generally outperforms other approaches. 2) Compared with LTH which uses wisely picked trainable subnetwork, our method achieves promising results on randomly picked subnetworks, which solidly validates our proposed DLTH. Our method also pushes the performance close to the L1 with comparable results and even better in certain cases; 3) Compared with training from scratch, our method consistently achieves better

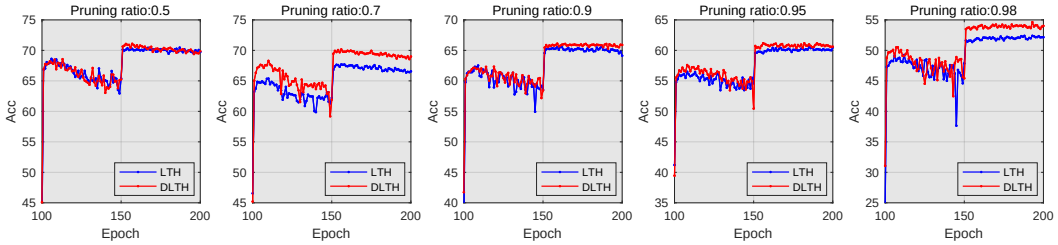


Figure 4: Test accuracy comparison of LTH and DLTH on CIFAR100 using ResNet56.

Table 3: Comparison with GraSP on CIFAR10 and CIFAR100 datasets.

ResNet56 + CIFAR10: Baseline accuracy: 93.50%					
Sparsity	50%	70%	90%	95%	98%
GraSP Wang et al. (2020a)	91.72	90.75	89.93	88.62	84.46
RST (**ours**)	**92.69±0.13**	**92.03±0.28**	**90.88±0.11**	**89.14±0.27**	**84.95±0.05**
ResNet56 + CIFAR100: Baseline accuracy: 72.54%					
Sparsity	50%	70%	90%	95%	98%
GraSP Wang et al. (2020a)	67.88	67.38	64.21	59.39	45.01
RST (**ours**)	**70.18±0.29**	**69.54±0.07**	**64.86±0.33**	**60.20±0.14**	**45.98±0.36**

performance; 4) The patterns on CIFAR10 and CIFAR100 are consistent. For example, the LTH becomes competitive with larger sparsity ratio; 5) Iterative strategy benefits both LTH and DLTH. The accuracy comparisons between LTH and DLTH are visualized in Fig. 4 based on CIFAR100 and ResNet56 with all pruning ratios. We plot the 100 to 200 finetuning epochs for a clear visualization.

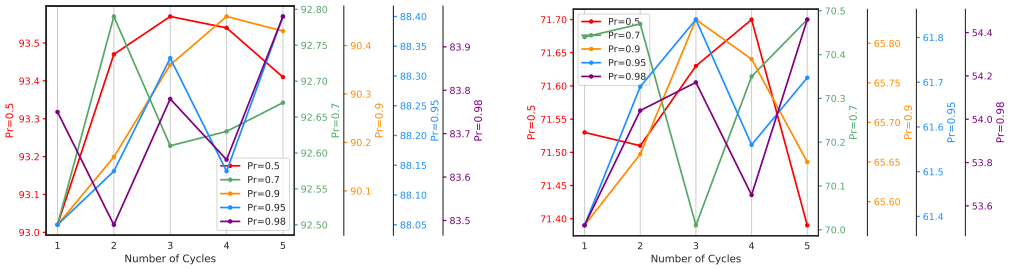
ImageNet Results. We compare RST with LTH in a ImageNet subset in Tab. 4. ImageNet subset contains 200 out of 1000 classes from the original dataset. We use the one-shot strategy for LTH/RST and our RST performs better.

Table 4: Test Accuracy of ResNet18 on ImageNet Subset.

ResNet18 + ImageNet Subset: Baseline: 78.96%		
Sparsity	50%	70%
LTH Frankle & Carbin (2018)	77.87	76.48
RST (**ours**)	**78.24**	**76.80**

Discussion of DLTH v. LTH. As a dual format of LTH, our DLTH allows randomly selecting sparse networks instead of using pretraining and pruning processes. We involve our RST procedure to transferring random tickets for winning, instead of directly employing obtained mask to represent the winning ticket by LTH. This pair of hypothesis use the same finetuning schedule and we compare their differences as "pretrain + prune" versus "randomly pick + transform": 1) Pretraining requires exactly the same computational resource as finetuning in our experiments. L1 pruning can be done efficiently without any parameter updating. 2) DLTH randomly picks subnetworks with no extra cost (even no sorting cost). The transformation can be seen as an optimization process. As mentioned in implementation section, the extra training cost can be calculated as $c_e = ((\lambda_b/\eta) \cdot v_\eta + v_s) * N_b/N_D$, where N_b and N_D are batch size and number of training samples. For RST, this number is (1/1e-4)*5+40000)*64/50000 $\approx$ 115 epochs (CIFAR as example). We conclude that 1) Since RST still maintains the full network during transformation, the required memory space is the same as pretraining in LTH. 2) Complete LTH and DLTH require 400 epochs and 315 epochs training cost with finetuning (200 epochs). The extra computational cost of DLTH is lower than LTH. 3) In 5-cycle iterative setting, for RST-Iter5: there are c_e = ((1/1e-4)*1+10000)*64/50000 = 25.6 epochs each cycle and 5 * 25.6 = 128 epochs for total before finetuning; for LTH-Iter5: this number is 5*50 = 250 epochs. DLTH is still more efficient than LTH. As shown is Fig. 3, using small cycle number may increase DLTH performance which even requires less costs.

Exactly as saying goes: *There ain't no such thing as a free lunch*, finding trainable sparse structure in a random dense network always needs extra cost. As the calculations above, these cost of both LTH and DLTH are in reasonable scope (ours is more efficient). However, our hypothesis considers a more general and valuable side of the pair-wise hypothesis. We reemphasize our advantages as follows. Firstly, we demonstrate any random subnetwork can be transferred into a good format



(a) Iterative cycle number ablation on CIFAR10. (b) Iterative cycle number ablation on CIFAR100.

Figure 5: Ablation study for iterative cycle number of DLTH. (a)/(b) show the 3 times average performance on CIFAR10/CIFAR100 with sparsity ratio from 0.5 to 0.98 and cycle number from 1 to 5. Different colors represent different sparsity ratios. (This figure is best viewed in color.)

Table 5: Top-1 and top-5 accuracy comparison with GraSP on ImageNet Subset.

ResNet18 + Imagenet Subset						
Sparsity	70%		90%		95%	
Metric	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
GraSP Wang et al. (2020a)	75.20	92.17	73.64	91.15	69.70	89.62
RST (**ours**)	**76.95**	**92.87**	**74.49**	**91.52**	**70.33**	**90.01**

which is more challenging compared with finding a specific trainable subnetwork in dense network presented by LTH. Secondly, LTH requires the mask determined by pruning matches with its initial dense network. The sparse structure is unpredictable. DLTH allows the sparse structure being controlled by ourselves as any subnetwork can be flexibly transformed into good condition.

Cycle Number Ablation Study. Fig. 5 shows the DLTH ablation study using different cycle number on CIFAR10/CIFAR100 datasets. Generally, larger cycle number improves the performance especially for CIFAR10. But the best cycle number for DLTH varies for different sparsity ratios. Overall, our DLTH performs consistently for different cycle numbers with promising robustness. We choose the cycle number as 5 to compare with other methods in Tab. 2, which may not be the best for DLTH but it still outperforms other approaches.

Comparison with GraSP. We compare our RST with Gradient Signal Preservation (GraSP) Wang et al. (2020a), a representative Pruning at Initialization (PI) approach, to further validate our DLTH. GraSP finds a good subnetwork in a randomly initialized dense network. The selected sparse structure is expected to perform promising trainability when optimized from scratch. Experiments are conducted on CIFAR10/CIFAR100 datasets (see Tab. 3) using ResNet56 and ImageNet subset (see Tab. 5) using ResNet18 on different sparsity ratios. GraSP picks a subnetwork from the random dense network and conducts finetuning. To be fair, our RST inherits the weights of the same random dense network and follows the same layer-wise sparsity ratio derived by GraSP. Our RST achieves better performances using the layer-wise sparsity from GraSP, which means RST has generalibility to handle different sparse structures and further validates the proposed DLTH.

7 Conclusion and Discussion

We propose a Dual Lottery Ticket Hypothesis (DLTH) as a dual problem of Lottery Ticket Hypothesis (LTH). We find that any random selected subnetwork in a random dense network can be transformed into a appropriate condition with admirable trainability. Compared with LTH, our DLTH considers a more challenging and general case about studying sparse network training, being summarized as follows: 1) It is executable to transferring any randomly selected subnetwork of a randomly initialized dense network into a format with admirable trainability; 2) The flexibility of selecting sparse structure ensures us having the controllability of the subnetwork instead of determined by pruning method; 3) Our DLTH studies the sparse network training in a complementary direction of LTH. It investigates general relationships between dense network and its sparse subnetworks which is expected to inspire following research for sparse network training. To validate our DLTH, we propose a simple strategy, Random Sparse Network Transformation (RST), for sparse network training. Specifically, we naturally involve a regularization term and leverage other weights to enhance sparse network learning capacity. Extensive experiments on several datasets with competitive comparison methods substantially validate the DLTH and the effectiveness of proposed RST.

REFERENCES

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pp. 322–332. PMLR, 2019. 1

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*, 2017. 3

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010. 4

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017. 1

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 6, 14

Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4): 485–532, 2020. 1

Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019. 3

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020. 3

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 1, 4, 6, 8, 12, 13, 14

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 1

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*, 2015a. 1, 2, 3

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015b. 1, 2, 3

Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993. 1, 3, 4

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 6

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017. 1

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 14

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990. 1, 3

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018. 1, 4

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019. 4

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 6, 12, 13

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017. 1, 3

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017. 3

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018. 3

Hesham Mostafa and Xin Wang. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*, pp. 4646–4655. PMLR, 2019. 3

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximations for model compression. *arXiv preprint arXiv:2004.14340*, 2020. 1

Suraj Srinivas and R Venkatesh Babu. Generalized dropout. *arXiv preprint arXiv:1611.06791*, 2016. 3

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International Conference on Machine Learning*, pp. 6566–6575. PMLR, 2019a. 1

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020a. 1, 4, 8, 9, 14, 15

Huan Wang, Qiming Zhang, Yuehai Wang, Lu Yu, and Haoji Hu. Structured pruning for efficient convnets via incremental regularization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019b. 5

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. *arXiv preprint arXiv:2012.09243*, 2020b. 3, 4, 5

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Emerging paradigms of neural network pruning. *arXiv preprint arXiv:2103.06460*, 2021. 4

Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018. 3

Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Yingyan Lin, Zhangyang Wang, and Richard G. Baraniuk. Drawing early-bird tickets: Toward more efficient training of deep networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxsrgStvr. 4, 6, 7

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

Guodong Zhang, James Martens, and Roger Grosse. Fast convergence of natural gradient descent for overparameterized neural networks. *arXiv preprint arXiv:1905.10961*, 2019. 1

A Detailed Experimental Comparison

As an extension of experimental comparisons in the tables of **Baseline Comparisons** section, we provide the performances of each run for `L1`, `LTH`, `Scratch`, and `RST` with their average and standard deviation shown in Tab. 6 and Tab. 7, which demonstrate the consistent effectiveness of our proposed RST. The column of `Mean±Std` is the same values in the tables of our main text.

Table 6: Different runs details of test accuracy comparisons on CIFAR10 dataset using ResNet56. Our method generally achieves better performance and validates the proposed DLTH.

ResNet56 + CIFAR10: Baseline accuracy: 93.50%				
Pruning ratio 50 %	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	93.40	93.22	93.36	93.33±0.08
LTH Frankle & Carbin (2018)	92.99	92.65	92.37	92.67±0.25
Scratch	92.47	92.07	92.93	92.49±0.35
RST (**ours**)	92.43	92.17	92.42	92.34±0.12
Pruning ratio 70%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	93.26	92.31	92.93	92.83±0.39
LTH Frankle & Carbin (2018)	92.34	91.49	91.82	91.88±0.35
Scratch	92.50	92.07	91.84	92.14±0.27
RST (**ours**)	92.26	91.98	92.57	92.27±0.24
Pruning ratio 90%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	91.62	91.57	91.82	91.67±0.11
LTH Frankle & Carbin (2018)	89.30	89.89	90.14	89.78±0.35
Scratch	89.97	89.73	89.98	89.89±0.12
RST (**ours**)	90.39	90.35	90.48	90.41±0.05
Pruning ratio 95%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	90.19	90.35	89.85	90.13±0.21
LTH Frankle & Carbin (2018)	88.41	87.35	88.40	88.05±0.50
Scratch	87.66	86.97	87.59	87.41±0.31
RST (**ours**)	88.36	88.21	88.16	88.24±0.08
Pruning ratio 98%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	85.15	84.67	84.52	84.78±0.27
LTH Frankle & Carbin (2018)	84.11	84.35	83.08	83.85±0.55
Scratch	82.82	82.17	83.14	82.71±0.40
RST (**ours**)	83.94	83.13	84.24	83.77±0.47

Table 7: Different runs details of test accuracy comparisons on CIFAR100 dataset using ResNet56. Our method generally achieves better performance and validates the proposed DLTH.

ResNet56 + CIFAR100: Baseline accuracy: 72.54%				
Pruning ratio 50 %	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	71.95	71.85	72.09	71.96±0.10
LTH Frankle & Carbin (2018)	70.08	69.32	70.46	69.95±0.47
Scratch	71.18	71.09	70.61	70.96±0.25
RST (**ours**)	70.56	71.74	71.08	71.13±0.48
Pruning ratio 70%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	71.35	71.54	71.87	71.59±0.21
LTH Frankle & Carbin (2018)	69.08	67.95	67.70	68.24±0.60
Scratch	68.61	69.01	68.16	68.59±0.35
RST (**ours**)	69.81	69.59	70.16	69.85±0.23
Pruning ratio 90%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	68.55	68.25	68.07	68.29±0.20
LTH Frankle & Carbin (2018)	65.08	66.23	65.67	65.66±0.47
Scratch	64.33	64.18	65.34	64.62±0.52
RST (**ours**)	66.42	65.98	66.11	66.17±0.18
Pruning ratio 95%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	64.57	64.61	65.05	64.74±0.22
LTH Frankle & Carbin (2018)	60.55	61.22	61.15	60.97±0.30
Scratch	59.89	59.66	60.24	59.93±0.24
RST (**ours**)	61.77	62.05	61.17	61.66±0.37
Pruning ratio 98%	Run # 1	Run # 2	Run # 3	Mean±Std
L1 Li et al. (2016)	52.19	48.88	49.04	50.04±1.52
LTH Frankle & Carbin (2018)	53.38	52.53	52.39	52.77±0.44
Scratch	51.36	51.00	50.05	50.80±0.55
RST (**ours**)	53.84	53.85	54.63	54.11±0.37

B VISUALIZATIONS OF PERFORMANCE COMPARISON

We provide more visualizations of performance comparisons. The comparison between Lottery Ticket Hypothesis (LTH) Frankle & Carbin (2018) and our Dual Lottery Ticket Hypothesis (DLTH) using the proposed Random Sparse Network Transformation (RST) on CIFAR10 Krizhevsky et al. (2009) is shown in Fig. 6. The comparisons between our RST and training from scratch on CIFAR10 and CIFAR100 Krizhevsky et al. (2009) datasets are shown in Fig. 7 and Fig. 8.

C IMPLEMENTATION DETAILS

We use 4 NVIDIA Titan XP GPUs to perform our experimental evaluations. Each experiment on CIFAR10/CIFAR100 dataset requires around 6 hours on one GPU and each experiment on ImageNet Deng et al. (2009) subset requires around 12 hours on two GPUs. Further, in the section of comparison between our RST and GraSP Wang et al. (2020a) algorithm, we use the layer-wise sparsity ratio obtained from GraSP. We provide the detailed layer-wise ratio for CIFAR10 and CIFAR100 shown in Fig. 9 and Fig. 10, respectively.

D CURRENT LIMITATION AND FUTURE WORK

Our Dual Lottery Ticket Hypothesis (DLTH) paves a complementary direction to explore Lottery Ticket Hypothesis (LTH) problem. Current experimental results and analysis substantially validate our proposed DLTH. However, there are still some remaining problems. For example, we validate the random sparse network can be manipulated into a good trainable condition in this work. However, which type of sparse structure can be transformed in a high efficiency and which fits the practical deployment appropriately to benefit computational acceleration? These questions are still open and we will continually explore them in our future work.

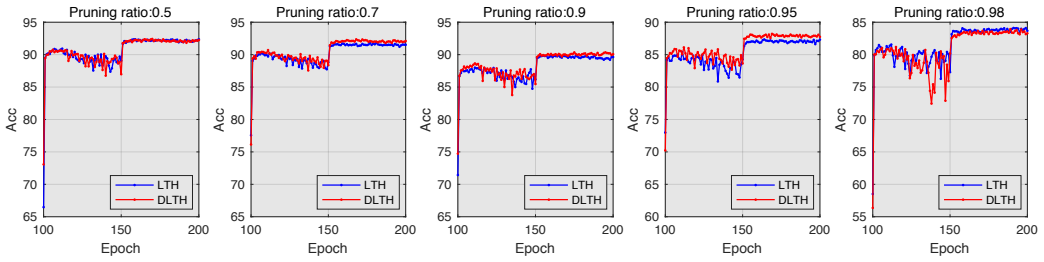


Figure 6: Visualization of test accuracy comparison of LTH and DLTH on CIFAR10 using ResNet56. Our DLTH general outperforms LTH and achieves comparable performance for 98% pruning ratio.

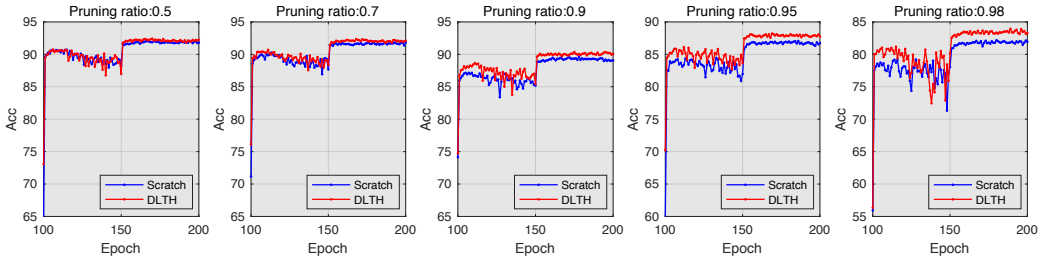


Figure 7: Visualization of test accuracy comparison of training from scratch and DLTH on CIFAR10 using ResNet56. Our training strategy consistently surpasses training from scratch on different pruning ratios. The performance gain increases when the pruning ratio becomes larger.

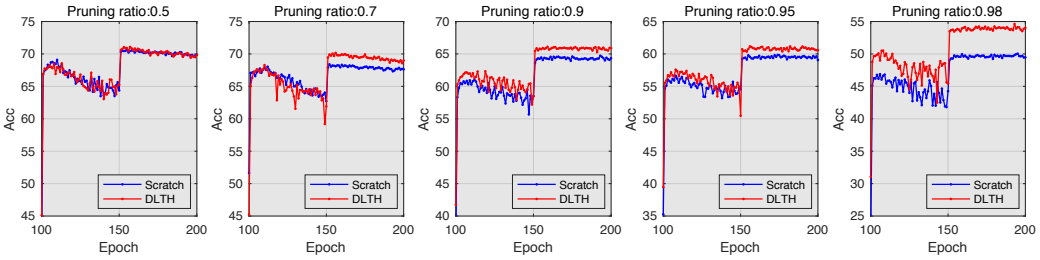


Figure 8: Visualization of test accuracy comparison of training from scratch and DLTH on CIFAR100 using ResNet56. Our training strategy consistently surpasses training from scratch on different pruning ratios. The performance gain increases when the pruning ratio becomes larger.

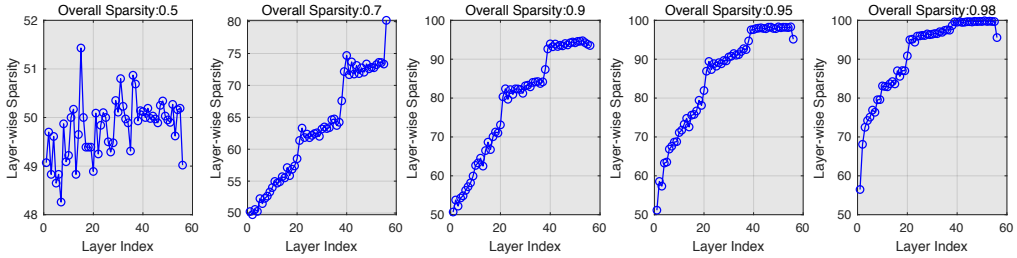


Figure 9: ResNet56 layer-wise sparsity ratio of different overall sparsity on CIFAR10 dataset. These layer-wise ratios are obtained by GraSP Wang et al. (2020a) algorithm. The last layer is the final fully-connected layer whose value is relatively special compared with others. The layer-wise sparsity generally starts from around 50% and increases when layer index increases for different overall sparsity (except for 50% whose layer-wise sparsity are distributed around 50% for all layers).

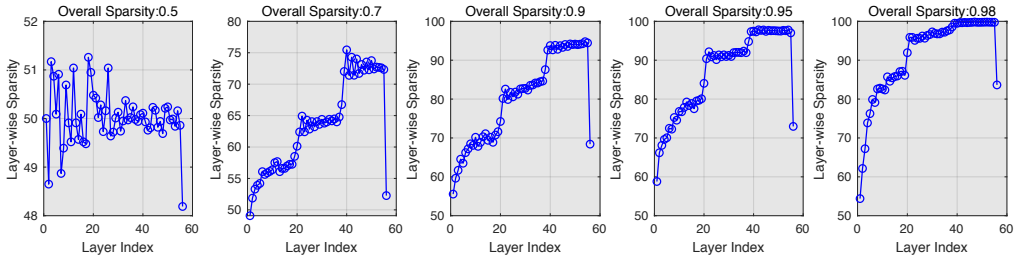


Figure 10: ResNet56 layer-wise sparsity ratio of different overall sparsity on CIFAR100 dataset. These layer-wise ratios are obtained by GraSP Wang et al. (2020a) algorithm. The last layer is the final fully-connected layer whose value is relatively special compared with others. The layer-wise sparsity generally starts from around 50% and increases when layer index increases for different overall sparsity (except for 50% whose layer-wise sparsity are distributed around 50% for all layers).