Drawing Robust Scratch Tickets: Subnetworks with Inborn Robustness Are Found within Randomly Initialized Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep Neural Networks (DNNs) are known to be vulnerable to adversarial attacks, i.e., an imperceptible perturbation to the input can mislead DNNs trained on clean images into making erroneous predictions. To tackle this, adversarial training is currently the most effective defense method, by augmenting the training set with adversarial samples generated on the fly. Interestingly, we discover for the first time that there exist subnetworks with inborn robustness, matching or surpassing the robust accuracy of the adversarially trained networks with comparable model sizes, within randomly initialized networks without any model training, indicating that adversarial training on model weights is not indispensable towards adversarial robustness. We name such subnetworks Robust Scratch Tickets (RSTs), which are also by nature efficient. Distinct from the popular lottery ticket hypothesis, neither the original dense networks nor the identified RSTs need to be trained. To validate and understand this fascinating finding, we further conduct extensive experiments to study the existence and properties of RSTs under different models, datasets, sparsity patterns, and attacks, drawing insights regarding the relationship between DNNs' robustness and their initialization/overparameterization. Furthermore, we identify the poor adversarial transferability between RSTs of different sparsity ratios drawn from the same randomly initialized dense network, and propose a Random RST Switch (R2S) technique, which randomly switches between different RSTs, as a novel defense method built on top of RSTs. We believe our findings about RSTs have opened up a new perspective to study model robustness and extend the lottery ticket hypothesis. All codes will be released upon acceptance.

1 Introduction

There has been an enormous interest in deploying deep neural networks (DNNs) into numerous real-world applications requiring strict security. Meanwhile, DNNs are vulnerable to adversarial attacks, i.e., an imperceptible perturbation to the input can mislead DNNs trained on clean images into making erroneous predictions. To enhance DNNs' robustness, adversarial training augmenting the training set with adversarial samples is commonly regarded as the most effective defense method. Nevertheless, adversarial training is time-consuming as each stochastic gradient descent (SGD) iteration requires multiple gradient computations to produce adversarial images. In fact, the actual slowdown factor over standard DNN training depends on the number of gradient steps used for adversarial example generation, which can result in a 3~30 times longer training time [1].

In this work, we ask an intriguing question: "*Can we find robust subnetworks within randomly initialized networks without any training*"? This question not only has meaningful practical implication but also potential theoretical ones. If the answer is yes, it will shed new lights on new methods towards

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

robust DNNs, e.g., adversarial training might not be indispensable towards adversarial robustness; furthermore, the existence of such robust subnetworks will extend the recently discovered lottery ticket hypothesis (LTH), which articulates that neural networks contain sparse subnetworks that can be effectively trained from scratch when reset to their initialization, and thus our understanding towards robust DNNs. In particular, we make the following contributions:

- We discover for the first time that there exist subnetworks with inborn robustness, matching or surpassing the robust accuracy of adversarially trained networks with comparable model sizes, within randomly initialized networks without any model training. We name such subnetworks Robust Scratch Tickets (RSTs), which are also by nature efficient. Distinct from the popular LTH, neither the original dense networks nor the identified RSTs need to be trained. For example, an RST with 90% sparsity drawn from a randomly initialized *WideResNet-32* achieves a 1.14%/1.72% higher robust/natural accuracy and a 60% parameter reduction without any training, compared with the adversarially trained dense *ResNet-18*.
- We propose a generic method to search for RSTs within randomly initialized networks, and conduct extensive experiments to identify and study the consistent existence and properties of RSTs under different DNN models, datasets, sparsity patterns, and attack methods, drawing insights regarding the relationship between DNNs' robustness and their initialization/overparameterization. Our RST findings about RSTs opens up a new perspective to study model robustness and can be viewed as a complement to LTH.
- We identify the poor adversarial transferability between RSTs of different sparsity ratios drawn from the same randomly initialized dense network, and propose a Random RST Switch (R2S) technique, which randomly switches between different RSTs, as a novel defense method built on top of RSTs. While an adversarially trained network shared among datasets suffers from degraded performance, our R2S enables the use of one random dense network for effective defense across different datasets.

2 Related Works

Adversarial attack and defense. DNNs are vulnerable to adversarial attacks, i.e., an imperceptible perturbation on the inputs can confuse the network into making a wrong prediction [2]. There has been a continuous war [3, 4] between adversaries and defenders. On the adversary side, stronger attacks continue to be proposed, including both white-box [5, 6, 7, 8, 9] and black-box ones [10, 11, 12, 13, 14], to notably degrade the accuracy of the target DNNs. In response, on the defender side, various defense methods have been proposed to improve DNNs' robustness against adversarial attacks. For example, random smoothing [15, 16] on the inputs can certifiably robustify DNNs against adversarial attacks; [17, 18, 19, 20, 21] purify the adversarial samples back to the distribution of clean samples; and [22, 23, 24] adopt detection models to distinguish adversarial example from clean ones. In particular, adversarial training [25, 5, 26, 27] is currently the most effective defense method. In this work, we discover the existence of RST, i.e., there exist subnetworks with inborn robustness that are hidden in randomly initialized networks without any training.

Model robustness and pruning. Both robustness and efficiency matter for many DNN-powered intelligent applications. There have been pioneering works [28, 29, 30, 31, 32, 33, 34] that explore the potential of pruning on top of robust DNNs. In general, it is observed that moderate sparsity is necessary for maintaining DNNs' adversarial robustness, over-sparsified DNNs are more vulnerable [28], and over-parameterization is important for achieving decent robustness [5]. Pioneering examples include: [29] finds that naively pruning an adversarially trained model will result in notably degraded robust accuracy and concurrently conducts adversarial training and ADMM based pruning [30] to achieve both robustness and efficiency; [31] adopts pruning techniques that are aware of the robust training objective via learning a binary mask on an adversarially pretrained network; and [33] prunes vulnerable features via a learnable mask and proposes a vulnerability suppression loss to minimize the feature-level vulnerability. In contrast, our work studies the existence, properties, and potential applications of RSTs, drawn from randomly initialized networks.

Lottery ticket hypothesis. The LTH [35] shows that there exist small subnetworks (i.e., lottery tickets (LTs)) within dense, randomly initialized networks, that can be trained alone to achieve comparable accuracies to the latter. The following works aim to: (1) study LTs' properties [36, 37, 38, 39], (2) improve LTs' performance [40, 41, 42, 43], and (3) extend LTH to various networks/tasks/training-pipelines [44, 45, 46, 47, 48, 49, 50, 51, 52]. The LTs drawn by existing works need to be (1) drawn

from a pretrained network (except [43]), and (2) trained in isolation after being drawn from the dense network. In contrast, RSTs are drawn from randomly initialized networks and, more importantly, no training is involved. We are inspired by [53], which finds that randomly weighted networks contain subnetworks with impressive accuracy and [54, 55], which provide theoretical support for [53] and extend [53] to identify binary neural networks in randomly weighted networks, respectively. While DNNs' adversarial robustness is important, it is not clear whether robust subnetworks exist within randomly initialized networks. We provide a positive answer and conduct a comprehensive study about the existence, properties, and potential applications for RSTs.

3 Robust Scratch Tickets: How to Find Them

In this section, we describe the method that we adopt to search for RSTs within randomly initialized networks. As modern DNNs contain a staggering number of possible subnetworks, it can be non-trivial to find subnetworks with inborn robustness in randomly weighted networks.

3.1 Inspirations from previous works

Our search method draws inspiration from prior works. In particular, [29] finds that naively pruning an adversarially trained model will notably degrade the robust accuracy, [31] learns a binary mask on an adversarially pretrained network to make pruning techniques aware of model robustness, and [53] shows that randomly weighted networks contain subnetworks with impressive accuracy, inspiring us to adopt learnable masks on top of randomly initialized weights to identify the RSTs.

3.2 The proposed search strategy

Overview. Our search method adopts a sparse and learnable mask m associated with the weights of randomly initialized networks, where the search process for RSTs is equivalent to only update m without changing the weights as inspired by [31, 53]. To search for practically useful RSTs, the update of m has to (1) be aware of the robust training objective, and (2) ensure that the sparsity of m is sufficiently high, e.g., higher than a specified value $(1 - k/N)$ where k is the number of remaining weights and N is the total number of weights. In particular, we adopt an adversarial search process to satisfy (1) and binarize m to activate only a fraction of the weights in the forward pass to satisfy (2).

Objective formulation. We formulate the learning process of m as a minimax problem:

$$\arg\min_{m} \sum_{i} \max_{\|\delta\|_\infty \leq \epsilon} \ell(f(\hat{m} \odot \theta, x_i + \delta), y_i) \quad s.t. \ ||\hat{m}||_0 \leqslant k \tag{1}$$

where ℓ is the loss function, f is a randomly initialized network with random weights $\theta \in \mathbb{R}^N$, x_i and y_i are the i-th input and label pair, $\odot$ denotes element-wise product, δ is a small permutation applied to the inputs and ϵ is a scalar that limits the permutation magnitude in terms of the infinity norm. Following [53], $\hat{m} \in \{0, 1\}^d$ approximates the top k highest elements of $m \in \mathbb{R}^N$ using 1 and 0 otherwise during the forward pass to satisfy the target sparsity constraint, while all the elements in m will be updated during the backward pass via straight-through estimation [56]. Such an adversarial search process guarantees the awareness of robustness, which differs from vanilla adversarial training methods [25, 5, 26, 27] in that here the model weights are never updated. After the adversarial search, a found RST is claimed according to the finally derived binary mask $\hat{m}$. As discussed in Sec. 4.5, $\hat{m}$ can have different sparsity patterns, e.g., element-wise, row-wise, or kernel-wise, where RSTs are consistently observed.

Inner optimization. We solve the inner optimization in Eq. 1 with Projected Gradient Descent (PGD) [5] which iteratively adopts the sign of one-step gradient as an approximation to update δ with a small step size α, where the t-th iteration can be formulated as:

$$\delta_{t+1} = clip_\epsilon\{\delta_t + \alpha \cdot sign(\nabla_{\delta_t} \ell(f(\hat{m} \odot \theta, x_i + \delta_t), y_i))\} \tag{2}$$

where $clip_\epsilon$ denotes the clipping function that enforces its input to the interval $[-\epsilon, \epsilon]$. As shown in Sec. 4.6, other adversarial training methods can also be adopted to draw RSTs.

4 The Existence of Robust Scratch Tickets

In this section, we validate the consistent existence of RSTs across different networks and datasets with various sparse patterns and initialization methods based on the search strategy in Sec. 3.

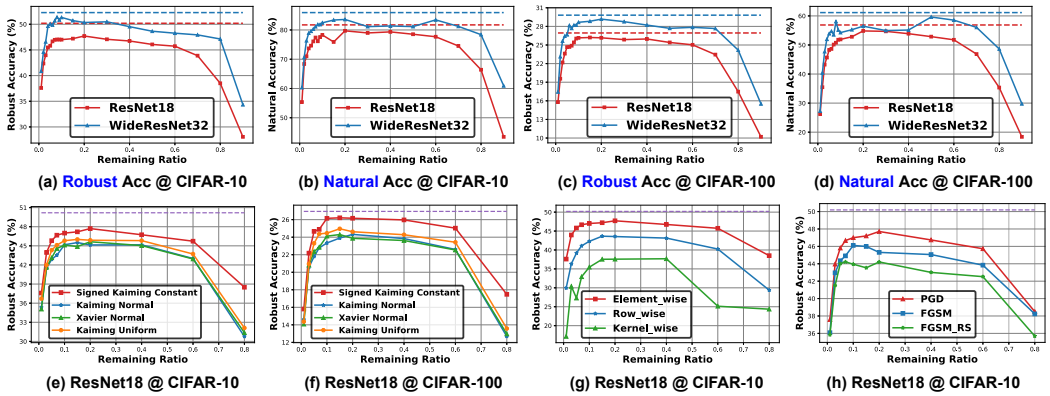


Figure 1: Illustrating RSTs' consistent existence, where (a)~(d): The robust and natural accuracy of RSTs with different remaining ratios in the two networks on CIFAR-10/100, respectively; (e)~(f): The robust accuracy of RSTs with different remaining ratios identified in ResNet-18 with different initializations on CIFAR-10/100, respectively; (g) The robust accuracy of RSTs with different sparsity patterns identified in ResNet-18 on CIFAR-10; (h) The robust accuracy of RSTs identified using different adversarial search methods in ResNet-18 on CIFAR-10. The accuracies of the adversarially trained original dense networks are annotated using dashed lines for reference.

4.1 Experiment Setup

Networks and datasets. Throughout this paper, we consider four networks and three datasets in total, including PreActResNet-18 (noted as ResNet-18 for simplicity)/WideResNet-32 on CIFAR-10/CIFAR-100 [57] and ResNet-50/ResNet-101 on ImageNet [58].

Adversarial search settings. For simplicity, we adopt layer-wise uniform sparsity when searching for RSTs, i.e., the weight remaining ratios k/N for all the layers are the same. On CIFAR-10/CIFAR-100, we adopt PGD-7 (7-step PGD) training for the adversarial search in Eq. 1 following [26, 5]. On Imagenet, we use FGSM with random starts (FGSM-RS) [26], which can be viewed as a 1-step PGD, to update the mask m. The detailed search settings can be found in the Appendix.

Adversarial attack settings. We adopt PGD-20 [5] attacks with $\epsilon = 8$, which is one of the most effective attacks, if not specifically stated. We further examine the robustness of RSTs under more attack methods and larger perturbation strengths in Sec. 5.3 and 5.4, respectively.

Model initialization. As RSTs inherit the weights from randomly initialized networks, initialization can be crucial for their robustness. We consider four initialization methods: the Signed Kaiming Constant [53], Kaiming Normal [59], Xavier Normal [60], and Kaiming Uniform [59]. If not specifically stated, we adopt the Signed Kaiming Constant initialization thanks to its most competitive results (see Sec. 4.4). We report the average results based on three runs with different random seeds.

4.2 RSTs exist on CIFAR-10/100

Fig. 1 (a)~(d) visualizes both the robust and natural accuracies of the identified RSTs with different weight remaining ratios (i.e., k/N) in two networks featuring different degrees of overparameterization on CIFAR-10/100. We also visualize the accuracy of the original dense networks *after adversarial training*, while RSTs are drawn from the same dense networks with only random initialization.

Observations and analysis. RSTs do exist as the drawn RSTs with a wide range of weight remaining ratios achieve both a descent robust and natural accuracy, even without any model training. Specifically, we can see that (1) RSTs can achieve a comparable robust and natural accuracy with the adversarially trained original dense networks under a wide range of remaining ratios (i.e., 5%~40%), e.g., the RST drawn from ResNet-18 with a remaining ratio of 20% (i.e., 80% sparsity) without any training suffers from only 2.08% and 2.43% drop in the robust and natural accuracy, respectively, compared with the adversarially trained dense ResNet-18; (2) RSTs hidden within randomly initialized networks with more overparameterization achieve a better robustness than the adversarially trained *dense* networks with a similar model size, e.g., the RST with a remaining ratio of 10% identified in the randomly initialized WideResNet-32 achieves a 1.14%/1.72% higher robust/natural accuracy with a 60% reduction in the total number of parameters without any training, over the adversarially trained *dense* ResNet-18; and (3) RSTs under very large or small remaining ratios have relatively inferior

robust and natural accuracies, which is expected since RSTs under large remaining ratios (e.g., > 60%) are close to the original randomly initialized networks that mostly make random predictions, while RSTs under small weight remaining ratios (e.g., 5%) are severely underparameterized.

4.3 RSTs exist on ImageNet

We further validate the existence of RSTs on ImageNet. As shown in Tab. 1, we can see that (1) RSTs do exist on ImageNet, e.g., RSTs identified in ResNet-101 with a comparable model size as the dense ResNet-18 achieves a 3.56%/4/31% and 1.22%/4.43% higher robust/natural accuracy under $\epsilon = 2$ and 4, respectively, compared with the adversarially trained ResNet-18; and (2) RSTs identified from more overparameterized networks achieve a better robust/natural accuracy than those from lighter networks, as RSTs from ResNet-101 are notably more robust than those from ResNet-50 under comparable model sizes.

Table 1: Comparing RSTs identified in ResNet-50/101 with the adversarially trained ResNet-18 on ImageNet.

Network	Remaining Ratio	ϵ	# Parameter (M)	Natural Acc (%)	Robust Acc (%)
ResNet-18 (Dense)	-	2	11.17	56.25	37.09
RST @ ResNet-50	0.4	2	9.41	53.84	35.24
RST @ ResNet-110	0.25	2	10.62	**60.56**	**40.65**
ResNet-18 (Dense)	-	4	11.17	50.41	25.46
RST @ ResNet-50	0.4	4	9.41	46.36	21.99
RST @ ResNet-110	0.25	4	10.62	**54.84**	**26.68**

4.4 RSTs exist under different initialization methods

We mainly consider Signed Kaiming Constant initialization [53] for the randomly initialized networks, as mentioned in Sec. 4.1, and here we study the influence of different initialization methods on RSTs. Fig. 1 (e) and (f) compare the robust accuracy of the RSTs drawn from ResNet-18 with different initializations on CIFAR-10/100, respectively. Here we only show the robust accuracy for better visual clarity, as the corresponding natural accuracy ranking is the same and provided in the Appendix.

Observations and analysis. We can see that (1) RSTs consistently exist, when using all the initialization methods; and (2) RSTs drawn under the Signed Kaiming Constant [53] initialization consistently achieve a better robustness than that of the other initialization methods under the same weight remaining ratios, indicating that RSTs can potentially achieve stronger robustness with more dedicated initialization. We adopt Signed Kaiming Constant initialization in all the following experiments.

4.5 RSTs exist under different sparsity patterns

We mainly search for RSTs considering the commonly used element-wise sparsity, as mentioned in Sec. 4.1. A natural question is whether RSTs exist when considering other sparse patterns, and thus here we provide an ablation study. Specifically, for each convolutional filter, we consider another three sparse patterns: row-wise sparsity, kernel-wise sparsity, and channel-wise sparsity.

Observations and analysis. Fig. 1 (g) shows that RSTs with more structured row-wise and kernel-wise sparsity still exist, although suffering from a larger robustness accuracy drop over the adversarially trained original dense networks, and we fail to find RSTs with a decent robustness when considering channel-wise sparsity. Indeed, RSTs with more structured sparsity show a more inferior accuracy, since it is less likely that a consecutive structure within a randomly initialized network will be lucky enough to extract meaningful and robust features. Potentially, customized initialization can benefit RSTs with more structured sparse patterns and we will leave this as a future work.

4.6 RSTs exist under different adversarial search methods

To study RSTs' dependency on the adopted adversarial search methods, here we adopt another two adversarial search schemes, FGSM [2] and FGSM-RS [26], for the inner optimization in Eq. 1.

Observations and analysis. Fig. 1 (h) shows that (1) using an FGSM or FGSM-RS search scheme can also successfully identify RSTs from randomly initialized networks; and (2) RSTs identified uing PGD-7 training consistently show a better robustness under the same weight remaining ratios compared with the other two variants. This indicates that better adversarial search schemes can lead to high quality RSTs and there potentially exist more robust RSTs within randomly initialized networks than the reported ones, when adopting more advanced adversarial search schemes.

5 The Properties of Robust Scratch Ticket

In this section, we systematically study the properties of the identified RSTs for better understanding.

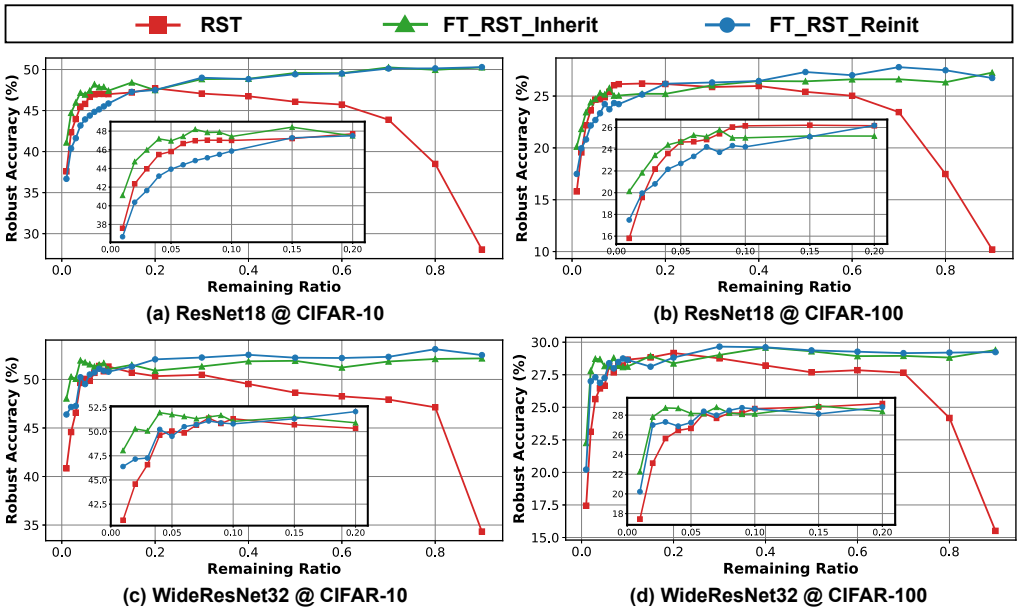


Figure 2: Comparing the robust accuracy of RSTs, fine-tuned RSTs with inherited weights, and fine-tuned RSTs with reinitialization, with zoom-ins for the low remaining ratios (1%~20%).

5.1 Robustness of vanilla RSTs vs. fine-tuned RSTs

An interesting question is "*how do the vanilla RSTs perform as compared to RSTs with fine-tuned model weights*"? Inspired by [35], we consider two settings: (1) fine-tuned RSTs starting with model weights inherited from the vanilla RSTs, and (2) fine-tuned RSTs with reinitialized weights. Fig. 2 compares their accuracies on three networks and two datasets with zoom-ins under low remaining ratios (1%~20%). Here we only show the robust accuracy for better visual clarity, where the natural accuracy with a consistent trend is provided in the Appendix.

Observations and analysis. We can see that (1) the vanilla RSTs achieve a comparable robust accuracy with that of the fine-tuned RSTs under a wide range of weight remaining ratios according to Fig. 2 (a), excepting under extremely large and small remaining ratios for the same reason as analyzed in Sec. 4.2; (2) under low remaining ratios (1%~20%), fine-tuned RSTs with inherited weights can mostly achieve the best robustness among the three RST variants, but the robust accuracy gap with the vanilla untrained RSTs is within -0.22%~3.52%; and (3) under commonly used weight remaining ratios (5%~20%) [61], fine-tuned RSTs with re-initialization can only achieve a comparable or even inferior robust accuracy compared with the vanilla RSTs without any model training. We also find that under extremely low remaining ratios like 1% (i.e., 99% sparsity), fine-tuned RSTs with re-initialization achieves a better robustness than the vanilla RSTs on extremely overparameterized networks (e.g, WideResNet-32) since they still retain enough capacity for effective feature learning.

Key insights. The above observations indicate that the existence of RSTs under low weight remaining ratios reveals **another lottery ticket phenomenon** in that (1) an RST without any model training can achieve a better or comparable robust and natural accuracy with an adversarially trained subnetwork of the same structure, and (2) the fine-tuned RSTs with inherited weights can achieve a notably better robust accuracy than the re-initialized ones, indicating that RSTs win a better initialization.

5.2 Vanilla RSTs vs. RTTs drawn from trained networks

As the adversarial search in Eq. 1 can also be applied to trained networks, here we study whether subnetworks searched from trained models, called robust trained tickets (RTTs), can achieve a comparable accuracies over the vanilla RSTs. Fig. 3 compares RST with two kinds of RTTs, i.e., natural RTTs and adversarial RTTs which are searched from naturally trained and adversarially trained networks, respectively. For better visual clarity, we only show the robust accuracy in Fig. 3, and the corresponding natural accuracy with a consistent trend is provided in the Appendix.

Observations and analysis. We can observe that (1) adversarial RTTs consistently achieve the best robustness, indicating that under the same weight remaining ratio, subnetworks drawn from

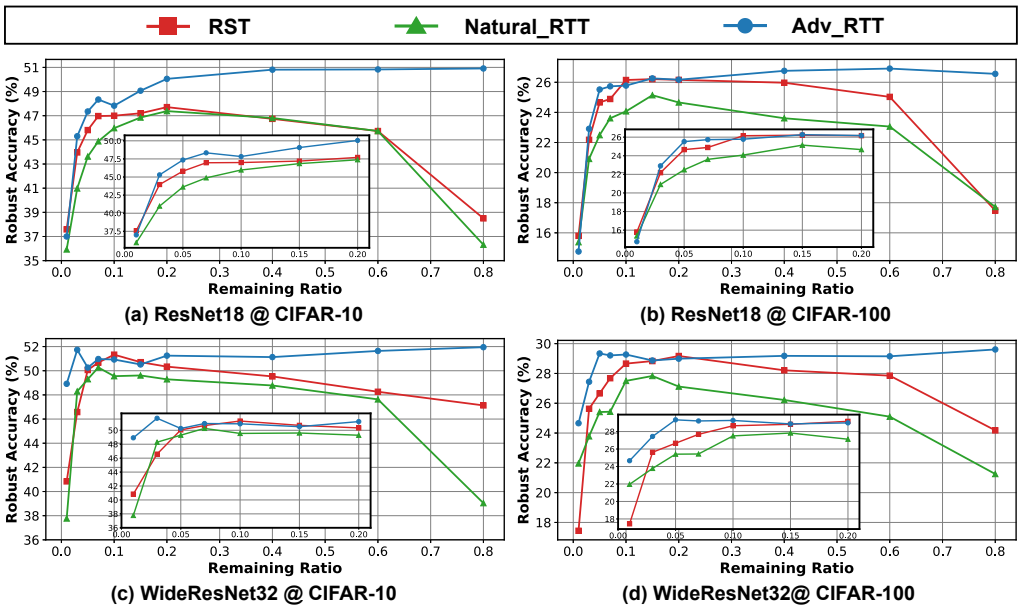


Figure 3: The robust accuracy achieved by RSTs, natural RTTs, and adversarial RTTs drawn from ResNet-18/WideResNet-32 on CIFAR-10/100, with zoom-ins under low remaining ratios (1%~20%).

adversarially trained networks are more robust than RSTs drawn from randomly initialized or naturally trained ones; (2) there exist natural RTTs with decent robustness even if the robust accuracy of the corresponding trained dense networks is almost zero; and (3) interestingly, RSTs consistently achieve a better robust/natural accuracy over natural RTTs, indicating the subnetworks from naturally trained networks are less robust than the ones drawn from randomly initialized networks.

To understand the above observations, we visualize the normalized distance between the feature maps of the last convolution layer generated by (1) clean images and (2) noisy images, i.e., clean images plus a random noise with a magnitude of ϵ on top of ResNet-18 with random initialization (Signed Kaiming Constant [53]), weights trained on clean images, and weights trained on adversarial images on CIFAR-10, inspired by [62]. Fig. 4 shows that the naturally trained ResNet-18 suffers from large distance gaps between the feature maps with clean and noisy inputs, indicating that it is more sensitive to the perturbations applied to its inputs and thus the natural RTTs identified within it are potentially less robust, while the adversarially trained ResNet-18 is the opposite.

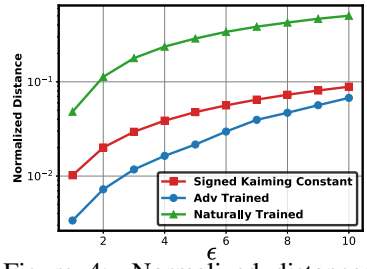


Figure 4: Normalized distances between the feature maps generated by clean and noisy images on ResNet-18 / CIFAR-10.

Is overparameterization necessary? We further compare the vanilla RSTs, fine-tuned RSTs with inherited weights, and adversarial RTTs, drawn from ResNet-18 on CIFAR-10 in Fig. 5. We can see that adversarial RTTs consistently achieve the best robust under the same weight remaining ratio. The comparison between adversarial RTTs and fine-tuned RSTs with inherited weights indicates that robust subnetworks within an adversarially trained and overparameterized network can achieve a better robustness than those drawn from a randomly initialized network or adversarially trained from scratch, i.e., the overparameterization during adversarial training is necessary towards decent robustness of RSTs.

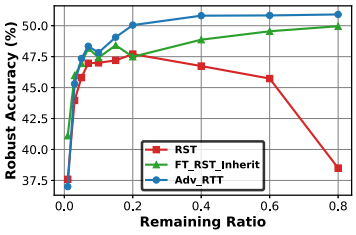
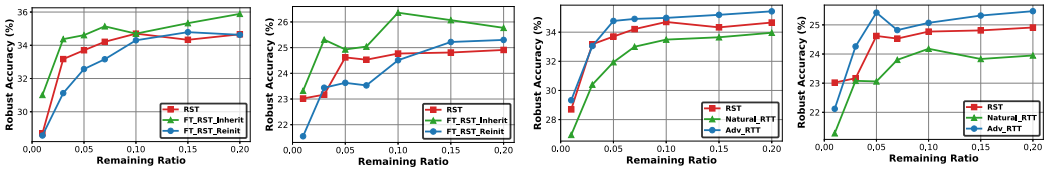


Figure 5: Comparing RSTs, fine-tuned RSTs with inherited weights, and adversarial RTTs.

5.3 RSTs' scalability to stronger perturbations

To explore whether the findings in Sec. 5.1 and Sec. 5.2 can be scaled to more aggressive attacks with stronger perturbations, we visualize the comparison between RSTs and fine-tuned RSTs/RTTs in Fig. 6, when considering the perturbation strength $\epsilon = 12, 16$ in ResNet-18 on CIFAR-10.



(a) $\epsilon = 12$ (b) $\epsilon = 16$ (c) $\epsilon = 12$ (d) $\epsilon = 16$

Figure 6: Validating RSTs' properties under stronger perturbations on ResNet-18 / CIFAR-10. (a)∼(b): Comparing vanilla RSTs with fine-tuned RSTs under ϵ =12 and 16, respectively; and (c)∼(d): Comparing vanilla RSTs with RTTs under ϵ =12 and 16, respectively.

Observations and analysis. We can observe that (1) RSTs still can be successfully identified under stronger perturbations; and (2) the insights in both Sec. 5.1 and Sec. 5.2 still hold and the accuracy gaps between the vanilla RSTs and natural RTTs are even larger under stronger perturbations.

5.4 RSTs' robustness against more adversarial attacks

We also evaluate the robustness of RSTs against more adversarial attacks, including the CW-L2/CW-Inf attacks [7], an adaptive attack called Auto-Attack [6], and the gradient-free Bandits [11] (detailed results are the Appendix), and find that RSTs are generally robust against different attacks, e.g., under Auto-Attack/CW-Inf attacks on CIFAR-10, the RST with a weight remaining ratio of 5% identified in a randomly initialized WideResNet-32 achieves a 2.51%/2.80% higher robust accuracy, respectively, with a 80% parameter reduction compared with the adversarially trained *dense* ResNet-18.

6 Applications of the Robust Scratch Tickets

In this section, we show that RSTs can be utilized to build a defense method, serving as the first heuristic to leverage the existence of RSTs for practical applications. Specifically, we first identify the poor adversarial transferability between RSTs of different weight remaining ratios drawn from the same randomly initialized networks in Sec. 6.1, and then propose a technique called (R2S in Sec. 6.2. Finally, we evaluate R2S's robustness over adversarially trained networks in Sec. 6.3.

6.1 Poor adversarial transferability between RSTs with different weight remaining ratios

To study the adversarial transferability between RSTs of different remaining ratios drawn from the same randomly initialized networks, we transfer the adversarial attacks generated under one remaining ratio to attack RSTs with other remaining ratios and visualize the resulting robust accuracy on ResNet-18 with CIFAR-10 in Fig. 7. We can see that the robust accuracies in the diagonal (i.e., the same RST for generating attacks and inference) are notably lower than those of the non-diagonal ones, i.e., the transferred ones. Similar observations, provided in the Appendix, can consistently be found on different networks/datasets, indicating that the adversarial transferability between RSTs is poor and thus providing potential opportunities for developing new defense methods.

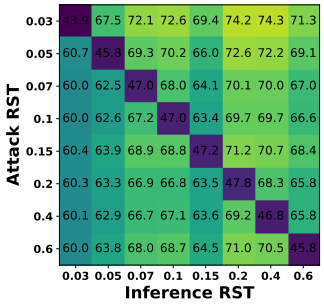


Figure 7: Adversarial transferability between RSTs from the same randomly initialized ResNet-18 on CIFAR-10, where the robust accurracies are annotated.

6.2 The proposed R2S technique

RSTs have two attractive properties: (1) RSTs with different remaining ratios from the same networks inherently have a weight-sharing nature; and (2) the adversarial transferability between RSTs of different remaining ratios is poor (see Fig. 7). To this end, we propose a simple yet effective technique called R2S to boost model robustness. Specifically, R2S randomly switches between different RSTs from a candidate RST set during inference, leading to a mismatch between the RSTs adopted by the adversaries to generate attacks and the RSTs used for inference. In this way, robustness is guaranteed by the poor adversarial transferability between RSTs while maintaining parameter-efficiency thanks to the weight-sharing nature of RSTs with different weight remaining ratios drawn from the same networks. The only

Table 2: Evaluating R2S under PGD-20 on CIFAR-10 (column 2∼3) and CIFAR-100 (column 4∼5).

Network	Natural Acc (%)	PGD-20 Acc (%)	Natural Acc (%)	PGD-20 Acc (%)
ResNet-18 (Dense)	81.73	50.19	56.88	26.94
R2S @ ResNet-18	77.49	63.93	52.12	41.01
WideResNet-32 (Dense)	85.93	52.27	61.14	29.81
R2S @ WideResNet-32	81.87	67.55	54.72	43.27

overhead is the required binary masks for different RSTs, which is negligible (e.g., 3%) as compared with the total model parameters. In addition, as evaluated in Sec. 6.3, the same randomly initialized network can be shared among *different tasks* with task-specific RSTs, i.e., only extra binary masks are required for more tasks, further improving parameter efficiency.

6.3 Evaluating the R2S technique

Evaluation setup. Since both the adversaries and defenders could adjust the probability for their RST choices, we assume that both adopt uniform sampling from the same candidate RST set for simplicity. We adopt an RST candidate set with remaining ratios [5%, 7%, 1%, 15%, 20%, 30%] and results under different RST choices can be found in the Appendix.

Table 3: Evaluating R2S under Auto-Attack [6] and CW-Inf attack [7] on CIFAR-10.

Network	Auto-Attack		CW-Inf	
	$\epsilon = 8$	$\epsilon = 12$	$\epsilon = 8$	$\epsilon = 12$
ResNet-18 (Dense)	46.39	41.13	48.99	45.91
R2S @ ResNet-18	51.92	49.77	54.02	52.63
WideResNet-32 (Dense)	49.66	44.97	54.16	49.22
R2S @ WideResNet-32	56.06	53.91	58.55	57.48

Results and analysis. As shown in Tab. 2 and 3, we can see that R2S consistently boosts the robust accuracy under all the networks/datasets/attacks, e.g., a 13.74% $\sim$ 15.28% and 5.53% $\sim$ 8.94% higher robust accuracy under PGD-20 and Auto-Attack , respectively, on CIFAR-10.

Applying random switch on adversarial RTTs. Indeed, the R2S technique can also be extended to adversarial RTTs as the poor adversarial transferability still holds between adversarial RTTs with different remaining ratios (see the Appendix). For example, when adopting the same candidate RST sets as in Tab. 2, R2S on adversarial RTTs achieves a 0.84%/1.58% higher robust/natural accuracy over R2S using RSTs on CIFAR-10 under PGD-20 attacks, indicating R2S is a general technique to utilize robust tickets. More results of R2S using adversarial RTTs are in the Appendix.

Advantages of RSTs over adversarial RTTs. A unique property of RSTs is that the same randomly initialized network can be shared among different tasks and task-specific RSTs can be drawn from this same network and stored compactly via instantiated binary masks, leading to advantageous parameter efficiency. In contrast, the task-specific adversarial RTTs drawn from adversarially trained networks on a different task will lead to a reduced robustness.

To evaluate this, we benchmark the CIFAR-100-specific adversarial RTTs drawn from ResNet-18 adversarially trained on CIFAR-10 (with the last fully-connected layer reinitialized), denoted as transferred adversarial RTTs, with (1) CIFAR-100-specific RSTs and (2) CIFAR-100-specific adversarial RTTs, in Fig. 8. We can see that the adversarial RTTs drawn from the adversarially trained network on another task consistently achieve the worst robust accuracy, e.g., a 1.68% robust accuracy drop over the corresponding RST under a remaining ratio of 5%. Therefore, it is a unique advantage of RSTs to improve parameter efficiency, especially when more tasks are considered under resource constrained applications, while simultaneously maintaining decent robustness.

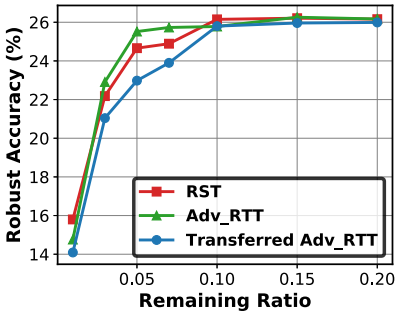


Figure 8: Comparing transferred adversarial RTTs with RSTs and adversarial RTTs in ResNet-18 on CIFAR-100.

7 Conclusion

In this work, we show for the first time that there exist subnetworks with inborn robustness, matching or surpassing the robust accuracy of the adversarially trained networks with comparable model sizes, within randomly initialized networks without any model training. Distinct from the popular LTH, neither the original dense networks nor the identified RSTs need to be trained. To validate and understand our RST finding, we further conduct extensive experiments to study the existence and properties of RSTs under different models, datasets, sparsity patterns, and attacks. Finally, we make a heuristic step towards the practical uses of RSTs. In particular, we identify the poor adversarial transferability between RSTs of different sparsity ratios drawn from the same randomly initialized dense network, and propose a novel defense method called R2S, which randomly switches between different RSTs. While an adversarially trained network shared among datasets suffers from degraded performance, R2S enables one random dense network for effective defense across different datasets.

References

[1] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *CoRR*, abs/1904.12843, 2019.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[4] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[8] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[9] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[10] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[11] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[12] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[13] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[14] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

[15] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[16] Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified adversarial robustness with additive noise. *arXiv preprint arXiv:1809.03113*, 2018.

[17] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

[18] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017.

[19] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[20] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

[21] Changhao Shi, Chester Holtz, and Gal Mishne. Online adversarial purification based on self-supervision. *arXiv preprint arXiv:2101.09387*, 2021.

[22] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[23] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[24] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *arXiv preprint arXiv:1807.03888*, 2018.

[25] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

[26] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2019.

[27] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[28] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. *arXiv preprint arXiv:1810.09619*, 2018.

[29] Shaokai Ye, Kaidi Xu, Sijia Liu, Hao Cheng, Jan-Henrik Lambrechts, Huan Zhang, Aojun Zhou, Kaisheng Ma, Yanzhi Wang, and Xue Lin. Adversarial robustness vs. model compression, or both? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 111–120, 2019.

[30] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[31] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 7, 2020.

[32] Adnan Siraj Rakin, Zhezhi He, Li Yang, Yanzhi Wang, Liqiang Wang, and Deliang Fan. Robust sparse regularization: Simultaneously optimizing neural network robustness and compactness. *arXiv preprint arXiv:1905.13074*, 2019.

[33] Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Adversarial neural pruning with latent vulnerability suppression. In *International Conference on Machine Learning*, pages 6575–6585. PMLR, 2020.

[34] Shupeng Gui, Haotao Wang, Chen Yu, Haichuan Yang, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. *arXiv preprint arXiv:1902.03538*, 2019.

[35] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[36] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[37] Jonathan Frankle, David J Schwab, and Ari S Morcos. The early phase of neural network training. *arXiv preprint arXiv:2002.10365*, 2020.

[38] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv preprint arXiv:1906.10732*, 2019.

[39] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

[40] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[41] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR, 2020.

[42] Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*, 2019.

[43] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

[44] Sai Prasanna, Anna Rogers, and Anna Rumshisky. When bert plays the lottery, all tickets are winning. *arXiv preprint arXiv:2005.00561*, 2020.

[45] 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.

[46] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *arXiv preprint arXiv:2007.12223*, 2020.

[47] Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*, 2020.

[48] Sharath Girish, Shishira R Maiya, Kamal Gupta, Hao Chen, Larry Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. *arXiv preprint arXiv:2012.04643*, 2020.

[49] Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Ultra-data-efficient gan training: Drawing a lottery ticket first, then training it toughly. *arXiv preprint arXiv:2103.00397*, 2021.

[50] Neha Mukund Kalibhat, Yogesh Balaji, and Soheil Feizi. Winning lottery tickets in deep generative models. *arXiv preprint arXiv:2010.02350*, 2020.

[51] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. *arXiv preprint arXiv:2102.06790*, 2021.

[52] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations, 2021a. URL https://openreview. net/forum*, 2021.

[53] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902, 2020.

[54] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

[55] James Diffenderfer and Bhavya Kailkhura. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. *arXiv preprint arXiv:2103.09377*, 2021.

[56] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[57] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[58] 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.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[60] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[61] 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.

[62] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Sec. 4.5, we discuss the potential of customizing new initializations to support more structured RSTs with decent robustness as our future work, which is a new problem created by our work.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work proposes a new finding related to model robustness and lottery ticket hypothesis which sheds lights on how to build more robust DNN models. So the negative societal impacts are not clear.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A]
(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will release our code, data, and instructions upon acceptance so that we can ensure ease of use.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Sec. 4.1 and the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We report the average results for clarity of the figures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] because the experiments are not computationally intensive.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] The used dataset and the license information can be found via the cited reference.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use ImageNet/CIFAR-10/CIFAR-100 dataset, which is known to have issues in this regard yet is indispensable for publishing in this research area.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We use ImageNet/CIFAR-10/CIFAR-100 dataset, which is known to have issues in this regard yet is indispensable for publishing in this research area.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]