[Re] Sanity-Checking Pruning Methods: Random Tickets can Win the Jackpot

Andrei Atanov
École polytechnique fédérale de Lausanne
andrei.atanov@epfl.ch

Valentina Shumovskaia
École polytechnique fédérale de Lausanne
valentina.shumovskaia@epfl.ch

Miloš Vujasinović
École polytechnique fédérale de Lausanne
milos.vujasinovic@epfl.ch

Reproducibility Summary

Scope of Reproducibility

In this work, we reproduce the following claims of the original paper:

Sanity-checking: show that recently proposed GraSP [1] and SNIP [2] methods for pruning neural networks do not need information about data (and labels) to find a sparse sub-network performing on par with the original network.

Random tickets: random pruning with simple per-layer keep-ratios has similar performance without using additional data/magnitude/gradient information.

Hybrid Ticket: the idea of *smart-ratios* can be applied to the recently proposed learning rate rewinding method [3] with further performance improvement.

Methodology

For baseline GraSP and SNIP methods, we use the open-sourced code [1]. Based on these methods' standard training pipeline, we implement two methods proposed in the original paper [4]: Random Tickets and Hybrid Tickets. As hardware, we use the V100 GPU, and each run takes approximately 1 GPU hour. Our code is available at `https://github.com/CS-433/cs-433-project-2-answer42`.

Results

We find that simple data corruption and random weight shuffling interventions do not hurt the performance as claimed in the original paper for the baseline methods. For Random Tickets, we reproduce the accuracy to within 1% for most of the settings. Hybrid Tickets and its baseline method are slightly different from the original paper for some settings on CIFAR-100. However, we did not find evidence that Hybrid Tickets improve the baseline's method performance.

What was easy

Random Tickets and Hybrid Tickets methods and experimental setting were well-described by the original paper authors, and we didn't face many difficulties implementing them. For baselines methods, we implement the modifications (sanity-checking, Seq. 2.2) based on the original code. The code is well-organized, and we do not face problems modifying it.

What was difficult

The authors do not provide information about the data preprocessing they perform, nor the number of pretraining epochs and rewind epoch they use for partially-trained tickets. Therefore, we use simple data preprocessing.

[1] GraSP code: `https://github.com/alecwangcq/GraSP`, and SNIP code:`https://github.com/mil-ad/snip`

1 Introduction

It has been shown that neural networks are overparametrized and can be significantly pruned with almost no performance drop during or after training [5, 6, 7, 8]. However, at high enough sparsity levels, called *non-trivial*, the found sub-networks cannot be trained to the same performance from scratch after re-initialization. The Lottery Ticket Hypothesis [9] suggests that it is possible to find a sub-network at a non-trivial sparsity level and train it from scratch with small or no accuracy drop compared to the original network. Several methods have been proposed to find such *winning tickets* before training [1, 2, 9] using gradient or magnitude information and several training examples.

The original paper [4] performs sanity-checking of several existing pruning methods and finds that these methods work even for random data and if the retained connections are shuffled inside each layer. Based on these observations, the authors hypothesize that it is only essential to use an appropriate pruning rate per layer and propose a new simple pruning method that doesn't use any training examples and prune weights randomly at initialization.

The recently proposed learning rate rewinding method [3] passes the sanity-checking, meaning the data information is important to keep the performance the same. To further improve this method's performance, the authors proposed to prune the network according to the smart-ratios.

2 Unstructured Pruning at Initialization

We consider a classification problem: given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ we train the weights $\{w_l\}_{l=1}^L$ of an L-layers neural network $f(x; \{w_l\}_{l=1}^L)$ minimizing the cross-entropy loss. An unstructured pruning method seeks to find binary masks $c_l \in \{0, 1\}^{m_l}$, where m_l is the number of the weights in a layer l, such that the pruned network $f(x; \{w_l \odot c_l\})$ has the same or similar performance, where $\odot$ denotes Hadamard product. The total pruning ratio for a network is defined as follows:

$$p = 1 - \frac{\sum_{i=1}^L \|c_l\|_0}{\sum_{i=1}^L m_l}, \tag{1}$$

The authors focus on pruning methods that find $\{c_l\}$ before training (except LT [9]) using a batch of training data. In this work we focus on two baseline methods: GraSP [1] ans SNIP [2].

2.1 Baselines

SNIP (Single-shot Network Pruning) [2]

The idea of the SNIP method is to treat each connection importance based on its effect on the loss function, independently of its weight. Such an importance measure is called *connection sensitivity* and defined in terms of how removing a single weight w_{lj} in isolation affects the loss:

$$S(w_{lj}) = \lim_{\epsilon \to 0} \left| \frac{L(w) - L(w + \epsilon \delta_j)}{\epsilon} \right| = \left| w_{lj} \frac{\partial L}{\partial w_{lj}} \right|, \tag{2}$$

where δ_j is one hot vector whose j-th element of equals w_{lj}. Once the sensitivity is computed, only the top-K connections are retained.

GraSP (Gradient Signal Preservation) [1]

The idea of GraSP method is close: each weight importance is based on how removing one weight will affect the gradient flow after pruning. Let δ be a perturbation of the initial weights, then its effect is computed as follows:

$$S(\delta) = \Delta L(w + \delta) - \Delta L(w). \tag{3}$$

2.2 Sanity-Checking

Dependency on Data

Random labels: in the pruning step, the initial dataset $\mathcal{D} = \{x_i, y_i\}$ is replaced with $\mathcal{D}' = \{x_i, y_i'\}$, where y_i' is randomly generated from a uniform distribution over each class.

Random pixels: in the pruning step, the initial dataset $\mathcal{D} = \{x_i, y_i\}$ is replaced with $\mathcal{D}' = \{x_i', y_i\}$, where x_i' is the randomly independently shuffled x_i.

Importance of the Pruned Network's Architecture

Layerwise rearrange: the mask of each pruned layer c_l is randomly independently rearranged into c'_l.

3 Random Tickets

The sanity-checking results suggest that it is crucial to have specific per-layer pruning ratios, and the weights in each layer could be pruned randomly. The authors introduce so-called *smart-ratios* p_l defined in the following way:

$$p_l = 1 - \frac{(L-l+1)^2 + (L-l+1)}{C} \tag{4}$$

where the normalizing constant C is defined s.t. the total number of retained weights stays the same, i.e. $\sum_{i=1}^{L} p_l m_l = p \sum_{i=1}^{L} m_l$. For the wider networks, such as VGG, the authors multiply the denominator in equation 4 by l^2 for faster decay. They additionally fix the pruning ratio of linear layers to be 0.7. Giving the smart-ratios equation 4, the weights are then pruned randomly in each layer without using any additional data or gradient/magnitude information. We will refer to this method as RT further in the text.

3.1 Hybrid Tickets

Hybrid tickets are a type of partially-trained tickets. The idea behind partially-trained tickets is to train a model (called pretraining) and then apply magnitude pruning that prunes a percent of weights with the lowest magnitude. Afterward, the model is rewinded to a middle epoch early in training and retrained again from there. The authors of the original paper consider two methods for rewinding a model:

- **Weights rewinding**: during pretraining the values of the weights at the epoch model should be rewinded to are saved. The values of the unpruned weights before retraining are set to be the same as on the given epoch. The learning rate schedule is set to start from the given epoch using the original schedule.
- **Learning rate rewinding**: the weights' values stay the same, but the learning rate schedule is set to start from the epoch the model should be rewinded to.

Hybrid tickets use learning rate rewinding, as well as smart ratios, first used with random tickets, in combination with magnitude pruning to prune weights.

4 Results

Based on our results we found that: 1) the baseline methods gives almost the same performance given random data or after perturbing the found architecture (see Seq.4.3), 2) the proposed RT method works on par with baseline (see Seq.4.4 and 3) Hybrid Tickets does not improve the results of the baseline method.

4.1 Experimental Setup

We test pruning methods using ResNet32 [10] and VGG19 [11] neural network architecture. These architectures are used also in the original paper.

We measure performance of pruned networks on datasets of different difficulties, CIFAR-10 and CIFAR-100. Both datasets consist of 50000 training and 10000 test colour images of size 32×32 pixels. However, CIFAR-10 contains 10 classes which these images should be classified into, while CIFAR-100 contains 100 classes.

We try to replicate results as close as possible. Therefore, we train all models using similar training parameters as the authors of the original paper. Models are trained using stochastic gradient descent and following parameters:

- initial learning rate: 0.1
- momentum: 0.9
- weight decay: 10^{-4}
- batch size: 64 for SNIP and Partially-trained Tickets / 128 for Random Tickets and GraSP
- number of epochs: 160

	CIFAR-10			CIFAR-100		
Sparsity	90%	95%	98%	90%	95%	98%
VGG19						
SNIP	93.29 (−0.36)	93.13 (−0.26)	89.56 (+8.19)	72.08 (−0.75)	*69.75 (−2.06)*	14.03 (+3.20)
GraSP	93.14 (+0.13)	92.62 (−0.20)	92.00 (+0.10)	71.63 (+0.56)	70.89 (−0.75)	68.30 (−0.04)
RT	**93.65** (−0.12)	**93.22** (−0.20)	**92.41** (−0.04)	**72.75** (+0.20)	**72.00** (+0.63)	**68.77** (−0.21)
ResNet32						
SNIP	92.55 (−0.26)	90.98 (−0.22)	87.49 (−0.45)	*63.45 (−6.52)*	*55.74 (−9.07)*	*49.35 (+1.38)*
GraSP	92.29 (−0.50)	91.36 (−0.44)	*88.09 (−1.12)*	*68.83 (−1.49)*	66.22 (−0.83)	59.25 (−0.00)
RT	**92.84** (−0.13)	**91.64** (+0.04)	**88.68** (−0.42)	*68.50 (−1.20)*	65.99 (−0.83)	**59.67** (−0.44)

Table 1: Test accuracy results for Random Tickets (RT), SNIP, and GraSP for CIFAR-10 and CIFAR-100 datasets for VGG19 and ResNet32 models. We provide results of our implementation of RT and results reproduced from GraSP and SNIP repositories. In the brackets, we show the difference to the reference numbers from the original paper. We found that our results are close to the reference in most cases (we put in italic cases with the difference more than 1% in accuracy) and support the hypothesis.

Weights are initialized using Kaiming initialization [12]. Learning rate is divided by 10 at $1/2$ and $3/4$ of the total number of training epochs.

Partially-trained ticket pruning methods require two step training, one before pruning and the other after. In each of the steps we train models for 160 epochs. Before the second step weights or learning rate, depending on the used rewinding method, are rewinded to the 40th epoch of the first training step.

We use standard data preprocessing and augmentation pipeline: random cropping, random horizontal flipping and normalization.

4.2 Baselines

The results for baseline methods can be seen in Tab 1. They are close to the results presented in the original paper and follow the similar pattern. However, we observe several outliers for CIFAR-100, especially in case of SNIP.

In some cases, the baseline methods GraSP and SNIP work worse than stated in the papers, even using original code. We note that in the official GraSP implementation the best accuracy over the training is computed, and it is not clear what numbers are reported. In our work, we always report the accuracy from the last epoch for all the methods. Using the best accuracy further improves the results and makes them closer to the original paper's results.

4.3 Sanity-Checking

The results of applying sanity-checks can be seen in Figure 1, which reproduces Figure 3 from the original paper. We compare GraSP and SNIP initial tickets for ResNet32 and VGG19 on CIFAR-10 dataset under different sanity checks: corrupted data (a combination of random labels and random pixels) and layerwise rearrange, described in Section 2.2. The results contradict both beliefs of data dependency of initial tickets and the importance of pruned structure.

4.4 Random Tickets

The results for Random Tickets can be seen in Tab.1. We found that results are close to the original paper in most cases, and the proposed method works on par or better than baselines that use additional information.

4.4.1 Hybrid Tickets

The results for Hybrid tickets can be seen in Tab 2. we found that the results on CIFAR-10 are close to the results of the original paper, while the results on CIFAR-100 are slightly different. The results indicate that hybrid tickets perform on par or worse than learning rate rewinding which was used as a baseline. Hybrid Tickets outperforms the baseline method only for high sparsity ratio (98%).

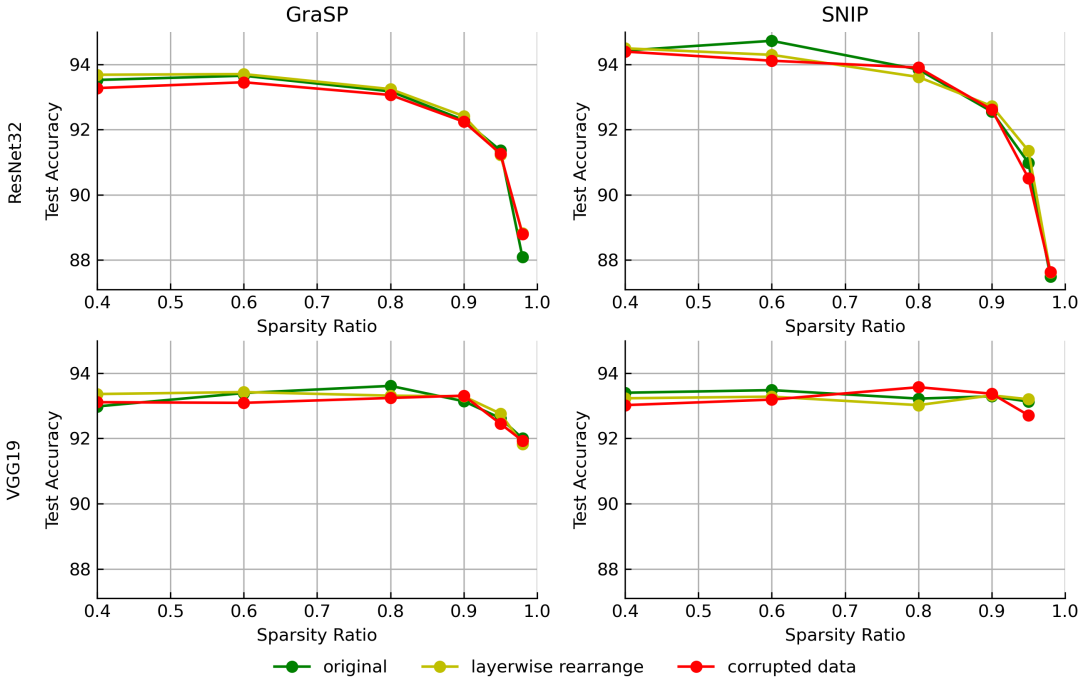

Figure 1: Comparison of test accuracies of the initial tickets of ResNet32 and VGG19 on CIFAR-10 dataset under different sanity checks. Corrupted data is obtained by applying random pixels and random labels sanity check on training dataset.

We also observe high difference in test accuracy for the baseline method used on VGG19 model, trained on CIFAR-100 dataset with 98% of weights pruned. They obtain 1% test accuracy which is the accuracy of a model that returns a constant value which indicates that the output is independent of input as a result of pruning a part of each path that connects the two in the original model. In our case, on the other hand, it doesn't happen and there exists at least one path between the input and the output layer. However, this could be just a matter of coincidence and would require additional runs to confirm the difference.

	CIFAR-10			CIFAR-100		
Sparsity	90%	95%	98%	90%	95%	98%
VGG19						
LR Rewind	**93.61** (−0.53)	**93.47** (−0.52)	10.00 (0.00)	***71.60*** *(-2.13)*	***71.13*** *(-1.26)*	*57.01* (+*56.01*)
Hybrid tickets	93.52 (−0.48)	93.30 (−0.53)	**92.93** (−0.59)	*71.43* (*-2.10*)	*70.96* (*-2.14*)	***70.01*** *(-1.60)*
ResNet32						
LR Rewind	**94.07** (−0.10)	**92.92** (−0.10)	**90.78** (−0.05)	**72.08** (−0.33)	***69.82*** (+*2.60*)	*61.25* (+*2.03*)
Hybrid tickets	93.60 (−0.38)	92.83 (−0.13)	90.47 (−0.38)	*69.95* (*-1.52*)	*68.02* (*-1.26*)	**62.56** (−0.88)

Table 2: Test accuracy results for Learning Rate Rewinding (LR Rewind) and Hybrid tickets on CIFAR-10 and CIFAR-100 datasets for VGG19 and ResNet32 models. In the brackets we show difference between our and the original results (we put into italic values where absolute value of difference is $> 1\%$). The found that proposed Hybrid Tickets method works on par or worse than the baseline and outperforms if only for the sparsity ratio of 98%.

5 Discussion

The reproduced experimental results are close to the original paper and strongly support the paper's main hypotheses: (1) Pruning methods don't seem to exploit information from training data to find good subnetworks; (2) The architecture of the pruned network is not crucial for good performance; (3) it is most important to use the specific pruning-ratio for each layer.

We should note that a part of the obtained numbers (e.g., GraSP results in Table 1) is slightly less than claimed in the paper. However, it does not contradict the aim of the original paper. We suppose that the reason for it could be that we use the last epoch test result, while authors might have used the best epoch test result (see Seq.4.2). We also observe high difference in results for SNIP on both models trained with CIFAR-100 dataset. Our implementation of SNIP is based on the implementation that closely follows the algorithm proposed in the original paper for SNIP [2, Algorithm 1].

Similarly, the results for Hybrid tickets are close to the original paper when trained on dataset CIFAR-10, but the difference increases on CIFAR-100.

References

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

[2] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019.

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

[4] Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. Sanity-checking pruning methods: Random tickets can win the jackpot. *arXiv preprint arXiv:2009.11094*, 2020.

[5] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017.

[6] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4857–4867. Curran Associates, Inc., 2017.

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

[8] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 1135–1143. Curran Associates, Inc., 2015.

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

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

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.