Lottery Pools

Citation for published version (APA):
Yin, L., Liu, S., Fang, M., Huang, T., Menkovski, V., & Pechenizkiy, M. (2022). Lottery Pools: Winning More by Interpolating Tickets without Increasing Training or Inference Cost.

Document status and date:
Published: 23/08/2022

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Lottery Pools: Winning More by Interpolating Tickets without Increasing Training or Inference Cost

Lu Yin¹, Shiwei Liu¹2†, Meng Fang³, Tianjin Huang¹, Vlado Menkovski¹, Mykola Pechenizkiy¹

¹ Eindhoven University of Technology
² University of Texas at Austin
³ University of Liverpool

Abstract

Lottery tickets (LTs) is able to discover accurate and sparse subnetworks that could be trained in isolation to match the performance of dense networks. Ensemble, in parallel, is one of the oldest time-proven tricks in machine learning to improve performance by combining the output of multiple independent models. However, the benefits of ensemble in the context of LTs will be diluted since ensemble does not directly lead to stronger sparse subnetworks, but leverages their predictions for a better decision. In this work, we first observe that directly averaging the weights of the adjacent learned subnetworks significantly boosts the performance of LTs. Encouraged by this observation, we further propose an alternative way to perform an “ensemble” over the subnetworks identified by iterative magnitude pruning via a simple interpolating strategy. We call our method Lottery Pools. In contrast to the naive ensemble which brings no performance gains to each single subnetwork, Lottery Pools yields much stronger sparse subnetworks than the original LTs without requiring any extra training or inference cost. Across various modern architectures on CIFAR-10/100 and ImageNet, we show that our method achieves significant performance gains in both, in-distribution and out-of-distribution scenarios. Impressively, evaluated with VGG-16 and ResNet-18, the produced sparse subnetworks outperform the original LTs by up to 1.88% on CIFAR-100 and 2.36% on CIFAR-100-C; the resulting dense network surpasses the pre-trained dense-model up to 2.22% on CIFAR-100 and 2.38% on CIFAR-100-C. Our source code can be found at https://github.com/luuyin/Lottery-pools

Introduction

Deep neural networks (DNNs) have revolutionized various machine learning fields with expressive performance (Le-Cun et al. 1989; Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; He et al. 2016; Silver et al. 2016; Dosovitskiy et al. 2020; Brown et al. 2020; Radford et al. 2021; Fedus, Zoph, and Shazeer 2021; Jumper et al. 2021). While achieving increasingly compelling results, a large concern is the massive parameter count that in billions, even trillions resulting heavy burden on environmental and financial systems (García-Martín et al. 2019; Schwartz et al. 2020; Patterson et al. 2021; Zhang et al. 2022). That motivates many techniques toward the efficiency of DNNs. Among them, sparsity is a leading approach that largely preserves the model performance while achieving appealing compression rates (Mozer and Smolensky 1989; Han, Mao, and Dally 2015; Molchanov et al. 2016; Liu et al. 2022; Chen et al. 2022; Sun et al. 2021; Yuan et al. 2021). A recent work on the Lottery Tickets (LTs) (Frankle and Carbin 2018) discovers the existence of sparse subnetworks within a standard network which can be trained in isolation to match the accuracy of the dense counterpart. These lottery tickets are empirically obtained by iterative magnitude pruning (IMP) at the random dense initialization or early training points called “rewinding” (Frankle et al. 2020). Since being proposed, LTs has become the leading approach to reducing model size while preserving accuracy.

However, LTs is a rather costly process involving multiple iterations of pruning-and-retraining, and once the subnetworks at the target sparsity are reached, the previous subnetworks with lower sparsity are commonly discarded, leading to a big waste of computation. One of the traditional ways that can directly benefit from multiple models in machine learning is ensemble. Ensemble (Hansen and Salamon 1990; Levin, Tishby, and Solla 1990; Fort, Hu, and Lakshminarayanan 2019) is well-known for its compelling performance improvements over independently trained, single networks by combining the predictions of the latter. Yet, ensemble does not directly lead to stronger sparse subnetworks but leverages their predictions to make a better decision. Hence, the benefits of ensemble in the context of LTs will be diluted.

In this paper, we build an efficient and accurate alternative to the naive LTs ensemble that yields subnetworks outperforming the original LTs by a large margin in both, in-distribution and out-of-distribution scenarios. We first observe that directly interpolating weights of the adjacent LTs subnetworks improves the performance of LTs. Inspired by this observation, we sequentially interpolate the weights of the natural “byproducts” of LTs (i.e., the previous subnetworks identified by IMP) with the target subnetwork if they improve accuracy on held-out data, following the recent emerging weight averaging techniques (Izmailov et al. 2018; Wortsman et al. 2022; Rame et al. 2022). This simple interpolating step is able to produce much stronger sparse and dense networks, without incurring any additional training and infer-
ence costs. We call our approach “Lottery Pools”\(^1\). Unlike the original LTs, Lottery Pools harnesses the advantage of all the LTs subnetworks, in turn, to further boost the performance of each of them. This property significantly increases the utility of the identified subnetworks compared with the original LTs, where the previous well-learned subnetworks are usually discarded. Overall, our contributions are summarised as follows:

- **Simple weight interpolation between two adjacent subnetworks boosts performance of LTs.** We surprisingly find that adjacent subnetworks of LTs can be linearly interpolated or even averaged into a single subnetwork with higher accuracy while maintaining the same sparsity (shown in Figure 1).

- **Towards stronger LTs subnetworks.** Encouraged by the above observation, we propose Lottery Pools that selectively interpolate multiple subnetworks into a single subnetwork. Lottery Pools is able to construct stronger subnetworks with much higher accuracy, while maintaining the original sparsity level. Simple as it is, we show that Lottery Pools (Interpolation) surpasses the original LTs by 1.88% and 1.72% on CIFAR-100 with VGG-16 and ResNet-18 respectively.

- **Towards stronger dense networks.** Besides the improved sparse subnetworks, we can also construct a stronger dense network by averaging the LTs tickets back to the pre-trained dense networks. Our reinforced dense network outperforms the original dense ResNet-18 by 2.22% and the original dense VGG-16 by 1.69% on CIFAR-100.

- **Towards Stronger in-distribution and out-of-distribution performance.** Thanks to the “ensemble” property of Lottery Pools, the enhanced (sub)networks enjoy a remarkable performance gain over the original LTs/dense model in both in-distribution (ID) predictive accuracy and out-of-distribution (OoD) robustness.

### Related work

**Lottery ticket hypothesis.** Lottery ticket (LTs) (Frankle and Carbin 2018) conjectures that there exists sparse subnetworks called winning tickets within a dense network, whose performance can match with the dense network when training from the same initialization. Later, weight/learning rate reconditioning techniques (Frankle et al. 2020; Renda, Frankle, and Carbin 2020) was proposed to scale up LTs to larger networks and datasets. Evcı et al. (2022) demonstrates that training LTs solutions with the same initialization converge to the same basin as the original pruning method that they are derived from. LTs has inspired many follow-up works to understand and extend LTs. Zhou et al. (2019); Ramanujan et al. (2020) successfully found winning tickets at the initialization even without training. Morcos et al. (2019) unveiled that the winning tickets discovered using larger datasets consistently transferred better than those generated using smaller datasets. Besides the original imagenet classification (Frankle and Carbin 2018), the existence of winning tickets has been broadly verified under diverse fields, such as natural language processing (Gale, Elsen, and Hooker 1999; Chen et al. 2020), generative adversarial networks (Chen et al. 2021), and reinforcement learning (Yu et al. 2019). Unlike LTs, Lottery Pools takes advantage of all the sparse subnetworks to construct stronger subnetworks without increasing any extra training or inference time.

**Weight averaging.** Model weight averaging has been widely studied in convex optimization and neural networks (Ruppert 1988; Polyak and Juditsky 1992; Zhang et al. 2019). Stochastic Weight Averaging (SWA) (Izmailov et al. 2018) and Exponential Moving Average (EMA) (Polyak and Juditsky 1992) average checkpoints along a single optimization trajectory and can roughly match the prediction ensemble performance. Yin et al. (2022) further generated SWA in the context of sparse training without any pretraining steps. Greedy soup (Wortsman et al. 2022) averages independent dense models across different runs, providing notable improvements.

Weight interpolation, as a more general case of weight averaging, draws explosive interest from the community recently. Nagarajan and Kolter (2019) empirically observed that there exists a linear path between the solutions learned on MNIST dataset with the same initialization. Neyshabur, Sedghi, and Zhang (2020) shown that two models fine-tuned from the same pre-trained model can be linearly connected to match the performance of the single model. Wortsman et al. (2022) proposed learned soup recipe that learns model interpolation by AdamW (Loshchilov and Hutter 2017). Weight interpolation has also been adopted to improve the accuracy of the patching task without compromising accuracy on the supported tasks and transfer learning (Ilharco et al. 2022). Frankle et al. (2020) introduced linear mode connectivity to study the instability of neural networks to the SGD noise introduced during training. They demonstrated that sparse subnetworks discovered by LTs can match the performance of the dense network only when they are stable to SGD noise. Following Frankle et al. (2020), we discover that two linearly connected subnetworks can be interpolated, leading to a more accurate subnetwork without any extra costs.

**Ensemble.** Ensembles (Hansen and Salamon 1990; Levin, Tishby, and Solla 1990) of neural networks have received large success in terms of the in-distribution accuracy (Perrone and Cooper 1992; Breiman 1996; Dietterich 2000), uncertainty estimation (Lakshminarayanan, Pritzel, and Blundell 2017; Wen, Tran, and Ba 2020), and out-of-distribution robustness (Ovadia et al. 2019; Gustafsson, Danelljan, and Schon 2020). Very recently, Liu et al. (2021) proposed an efficient ensemble framework that combines the predictions of multiple individual subnetworks, surpassing the generalization performance of the naive ensemble. Nevertheless, the ensemble requires performing a forward pass for each model, leading to extra costs.

\(^1\)Lottery pools refers to a group of people who purchase lottery tickets together to get better odds of winning a lottery. We borrow this concept to highlight that we combine (interpolate) multiple LTs subnetworks into a stronger one with higher accuracy.
Table 1: The primary methods contrasted in this work. $\tilde{\theta}$ is a subnetwork learned by IMP from different iteration. Cost refers to the memory and computes requirements during inference relative to a single model. All methods require the same training.

| Method          | Formulation                                                                 | Cost |
|-----------------|------------------------------------------------------------------------------|------|
| Lottery Tickets | $f(x; \theta)$                                                              | $O(1)$ |
| Naive Ensemble  | $\frac{1}{k} \sum_{i=1}^{k} f(x; \theta_i)$                               | $O(k)$ |
| Lottery Pools   | Algorithm 2                                                                  | $O(1)$ |

Algorithm 1: Pseudocode of LTs

Require: Randomly initialization network $\theta \in \mathbb{R}^d$, binary mask $m$, IMP iterations $T$, training steps used for rewinding $j$, pruning rate $p$.

1. Train $\theta$ to completion; save the weights at $j$ steps $\theta_j$.
2. for pruning iteration $t \in \{1, \ldots, T\}$ do
3. Prune the lowest magnitude $p$ of weights and update mask $m^t$.
4. Train $m^t \odot \theta_j$ to completion.
5. end for

Methodology

In this section, we introduce “Lottery Pools”, a simple weight interpolation approach that constructs more accurate subnetworks than LTs, without requiring any extra training or inference cost. Different from previous works on interpolation, our goal is to boost the performance of LTs subnetworks (including the pre-trained dense network) while maintaining their desirable sparsity.

We first recap the concept of the lottery ticket hypothesis. Then, we show that directly averaging LTs subnetworks identified at two adjacent IMP iterations leads to stronger subnetworks. Finally, we introduce our Lottery Pools, which is able to improve the accuracy of the original LTs by interpolating subnetworks obtained across different IMP iterations.

Recapping the lottery ticket hypothesis. The lottery ticket hypothesis (Frankle and Carbin 2018) indicates that there exist subnetworks (winning tickets) within dense network initialization such that those lottery tickets could be trained in isolation to the matching performance of their dense counterparts.

To be specific, we denote neural network with parameters $\theta \in \mathbb{R}^d$ as $f(x; \theta)$, after $j$ steps of training, there exists a sparse subnetwork characterized by the binary mask $m$ such that $f(x; m \odot \theta_j)$ will perform as well as $f(x; \theta)$ after training. The initial results show LTs hold when the subnetwork is trained with the original initialization, i.e. $j = 0$. Later results (Frankle et al. 2020) state a rewinding step to a pretrain point ($j > 0$) is required on larger datasets. The overall training procedure of LTs is given in Algorithm 1.

Simple weight averaging boosts the performance of LTs. Recently, the work on model soup (Wortsman et al. 2022) shows that averaging models fine-tuned from the same pre-trained model provides substantial performance improvements. Since the LTs subnetworks at different sparsities are also fine-tuned from the same pre-trained dense model, we hypothesize that these subnetworks can also be averaged for better performance.

Let’s simplify the sparse subnetwork parameters under a mask $m^t \odot \theta$ as $\theta$ for simplicity. To verify our hypothesis, we average two LTs subnetworks with different sparsities (assuming sparsity of $\tilde{\theta}_1$ is higher than sparsity of $\tilde{\theta}_2$) across all the learned subnetworks, i.e., $\frac{\tilde{\theta}_1 + \tilde{\theta}_2}{2}$. To address the sparsity decrease caused by average, we further use magnitude pruning $\frac{\tilde{\theta}_1 + \tilde{\theta}_2}{2}$ to the same sparsity as $\tilde{\theta}_1$ following (Yin et al. 2022). We consider two widely used settings for LTs: with rewinding and without rewinding. We report the results of ResNet-18 on CIFAR-100 within a heat map in Figure 1 and put the results of VGG-16 in Appendix.

As we can see, rewinding matters for the improved performance of weight averaging. With rewinding, simple averaging subnetworks from nearby IMP iterations could achieve better performance than the original LTs subnetworks (the diagonal cells), and the closer two subnetworks are located, the larger performance gains the averaged subnetworks tend to achieve. In stark contrast, we observe a significant accuracy drop without rewinding, across all sparsities. This observation is in line with the findings from Frankle et al. (2020) that large-scale settings are unstable to SGD noise at initialization according to linear interpolation. Thereby, we confirmed that our hypothesis holds in the context of rewinding.

Lottery Pools

Inspired by the above observation, we introduce Lottery Pools, a simple weight interpolation approach on LTs that leverages the subnetworks obtained across different IMP iterations. Lottery Pools has two key ideas: (1) Interpolating weights instead of simply averaging; (2) Sequentially searching over all the candidate LTs subnetworks and coefficients for interpolation.

Firstly, Lottery Pools goes beyond weight averaging and probes a more general variant of weight connection—weight interpolation. Linear weight interpolation has been previously used to study dense networks (Nagarajan and Kolter...
We argue that directly averaging two subnetworks (Algorithm 2: Lottery Pools Interpolation Recipe) might not be the optimal option, since the local linearly-connected minimum may lie in the sides of the linear path rather than in the right middle. To prove this, we linearly interpolate two adjacent LTs subnetworks, with various coefficient $\alpha \in [0, 1]$. We set the increments of $\alpha$ as 0.1 to create 9 interpolated subnetworks between two LTs subnetworks. The test loss/error are reported in Figure 2. The best accuracy is achieved at the middle points of the linear path in most cases (the middle part of the green lines) with exceptions at the highest and lowest sparsities (the endpoints of the green lines) where weight average achieves no better accuracy than interpolation. Again, interpolation without weight rewinding (yellow lines) fails to find such linear paths between two subnetworks.

Moreover, Figure 1 and 2 present that the accuracy of the interpolated subnetworks varies across different subnetworks pairs and different interpolation coefficients. While it is possible to adopt gradient-based optimization to learn the optimal subnetworks and coefficients, the cost is rather expensive. We instead choose a more practical way: greedily searching for interpolated subnetworks and the corresponding coefficients. Given a target LTs subnetwork $\tilde{\theta}_t$ learned at the $t$ iteration of IMP, we sequentially interpolate it with the rest of the subnetworks (Candidate Lottery Pools). For each candidate subnetwork, we search the best coefficients $\alpha$ over 11 candidates from 0.05 to 0.95 (Candidate Coefficient Pools). Please refer to the Appendix for more details. We only keep the incoming subnetwork for interpolation if its accuracy on the held-out set does not decrease.

Instead of loading all the candidate subnetworks in the memory (Wortsman et al. 2022), we apply a more memory-friendly search approach. Specifically, we iteratively interpolate one of the subnetworks in the Candidate Lottery Pools with our target subnetwork $\tilde{\theta}_t$ and let the resulting subnetwork be our new target subnetwork, i.e. $\tilde{\theta}_t \leftarrow \tilde{\theta}_{\text{inter}}$, until we have searched all the subnetworks. This operation allows us to accomplish the interpolation operation across all the candidate subnetworks by maintaining only one extra copy of the interpolated weights.

In summary, Lottery Pools is a two-step procedure for constructing stronger LTs subnetworks. **Step 1:** Perform the standard Lottery Tickets method; **Step 2:** Linearly interpolate the original LTs subnetworks to produce stronger sparse subnetworks. Please note that we adopt magnitude weight pruning to remove the weights with the smallest magnitude after each interpolation to maintain the same sparsity level as the original LTs.

### Experiments

To verify the effectiveness of Lottery Pools, we evaluate it with three popular model structures VGG-16, ResNet-18
and ResNet-34 on various datasets, including CIFAR-10 and CIFAR-100 and ImageNet.

Table 2: Implementation details, including: IMP iteration count, rewinding epochs, learning rate (LR), batch size (BS), learning rate drop (LR Drop), training epochs (Epoch), etc.

| Network      | Dataset      | Epochs | BS  | LR  | LR Drop, Epochs | Warmup | Rewinding | IMP Iterations |
|--------------|--------------|--------|-----|-----|-----------------|--------|-----------|----------------|
| ResNet-18    | CIFAR-10/100 | 182    | 128 | 0.1 | 16[30, 150]     | -      | -         | 9 epochs       |
| ResNet-16    | CIFAR-10/100 | 182    | 128 | 0.1 | 16[30, 150]     | -      | -         | 9 epochs       |
| ResNet-18/34 | ImageNet     | 19     | 128 | 0.1 | 16[30, 150]     | 5 Ep     | 5 Epochs   | 9              |

Experiments setup. Since our method directly performs weight interpolation over the subnetworks produced by LTs rewinding, our most direct baseline is the standard LTs rewinding. Following the common rewinding setting used in Frankle et al. (2020); Chen et al. (2020), we rewind the LTs roughly to the 5% training time. We summarize the implementation details for LTs in Table 2. To highlight the performance difference between weight interpolation and weight averaging, we implement two variants of Lottery Pools: Lottery Pools (Interpolation) and Lottery Pools (Average), the latter applies direct averaging instead of interpolation in the second step of Lottery Pools. The results are illustrated in Figure 3(a). All the reported results are averaged over 3 independent runs.

Comparison with the original LTs. Overall, we see a clear performance gain from Lottery Pools over the original LTs under different sparsity levels (including the dense network), and Lottery Pools (Interpolation) achieves better performance than Lottery Pools (Average) due to the searched optimal interpolation value. Impressively, Lottery Pools (Interpolation) achieves up to 1.88% and 1.72% accuracy increase over the original LTs with VGG-16 and ResNet-18 on CIFAR-100, respectively. Even on the relatively more saturated CIFAR-10, we still observe up to 0.93% and 1.05% performance gains with VGG-16 and ResNet-18. We highlight that Lottery Pools also outperforms LTs in even extremely sparse situations. For instance, Lottery Pools brings 0.81% higher accuracy over the LTs with VGG-16 on CIFAR-100.
and 0.6% higher accuracy on CIFAR-10, with only 1.8% weights. It is quite encouraging to see that Lottery Pools can still improve performance when the interpolating space is extremely small. Besides, by averaging the learned LTs subnetworks back to the pre-trained dense model, Lottery Pools (interpolation) could also construct stronger dense networks, which outperform the original dense ResNet-18 by 2.22% and the original dense VGG-16 by 1.69% on CIFAR-100. We report the results with ResNet-18/34 on ImageNet in the Appendix due to the limited space, where Lottery Pools also consistently outperforms LTs.

**Comparison with weight averaging baselines.** We further compare our method with two strong weight averaging baselines: SWA and EMA. SWA (Izmailov et al. 2018) averages the weights of multiple networks along a single optimization trajectory. By setting the averaging coefficient to \( \frac{1}{n+1} \), where \( n \) is the current model number, it achieves the same results of averaging across all the models while maintaining the memory consumption as just two DNNs. The exponential moving average (EMA) (Polyak and Juditsky 1992; Kingma and Ba 2014; Karras et al. 2017) average the weights of a series of models exponentially using a fixed decay factor, which was set to 0.95 in this experiment. As these baselines are all designed for dense model averaging, the same as Lottery Pools, we alter EMA and SWA by pruning the interpolated model to the sparsity of the original LHS during interpolating. In fact, both SWA and EMA could be regarded as special cases of linear interpolation that use specific averaging coefficients and decay factors for weight interpolating.

As shown in Figure 3(b), our method clearly outperforms the other baselines by a large margin, especially in highly sparse situations. That apparently comes from two possible reasons. Firstly, EMA and SWA use pre-chosen coefficient values, which might not generate all the subnetworks, whereas Lottery Pools searches for the optimal values for each subnetwork. Secondly, Lottery Pools greedily searches for potential subnetworks to interpolate, which can eliminate the negative effects of subnetworks that are adding networks within the different loss basins for interpolation.

**Out-of-distribution robustness.** Beyond the in-distribution accuracy, we highlight that Lottery Pools also improves the performance of LTs in the OoD scenario. We train Lottery Pools with standard CIFAR-10 and CIFAR-100 and test it on CIFAR-10-C and CIFAR-100-C, respectively. As shown in Figure 3(c), Lottery Pools improves the OoD robustness than the original LTs by a large margin with both dense and sparse subnetworks. Remarkably, it improves the dense ResNet-18 by 2.38% on CIFAR-100, and by 2.12% with VGG-16 on CIFRA-10. For sparse subnetworks, Lottery Pools achieves up to 2.27% performance gain with VGG-16, and 2.36% with ResNet-18 on CIFAR-100. This result indicates that our interpolated subnetworks is able to inherit the appealing properties of model ensemble, e.g., good OoD robustness.

**Extensive Analysis**

**Candidate lottery pools count.** In this section, we study how the number of candidate tickets count, i.e., \( \|S_t\|_0 \) affects the achieved model’s performance. In the main experiment section we take advantage of all the learned networks across different IMP iterations for interpolation. Here, we alter the count of networks in \( S_t \) as 4, 9, 19, which represents 25%, 50%, 100% total number of possible networks. All the net-
Lottery Pools (Interpolation)

| Accuacy (%) | Accuacy (%) |
|-------------|-------------|
| 71          | 73          |
| 50          | 75          |
| 75          | 76          |
| 100         | 100         |

Prune after

Prune during (ours) Prune after

Output Ensemble Lottery Pools (Interpolation)

When to Prune. We study when to prune the interpolated subnetwork to target sparsity. In the default setting of Lottery Pools, we perform the pruning operation every once when we interpolate a subnetwork during the greedy search (line 5, 6 in Algorithm 2), which we called prune during. Another option would be first greedy interpolating all the searched subnetworks and then pruning the achieved subnetwork to the target sparsity, namely prune after. We compare these two methods in Figure 6. As we can see, the performance of prune after drops dramatically at the high sparsity while prune during keeps a steady high accuracy across all the sparsities. The reason might be in prune after, we greedy search for the best un-pruned interpolated networks regardless of their performance at the target sparsity. Whereas the prune during operation in greedy search keeps the interpolated networks always having good performances at desirable sparsity level.

Conclusion

In this paper, we explore a new perspective to leverage the existing learned LTs subnetworks by interpolation. We call this approach Lottery Pools. Without increasing training or inference costs, a network can be identified with significant performance improvements for in- and out-distribution scenarios. Impressively, Lottery Pools is capable of creating not only stronger subnetworks that maintain the original LTs sparsity level but also stronger dense networks. Extensive experiments verify the effectiveness of Lottery Pools across various network architectures with VGG-16 and ResNet-18/34 on CIFAR-10/100 and ImageNet.

Acknowledgments

This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. NWO-2021.060, EINF-2694 and EINF-2943/L1.
References

Breiman, L. 1996. Bagging predictors. *Machine learning*, 24(2): 123–140.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.;Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models Are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Chen, T.; Chen, X.; Ma, X.; Wang, Y.; and Wang, Z. 2022. Coarsening the granularity: Towards structurally sparse lottery tickets. *arXiv preprint arXiv:2202.04736*.

Chen, T.; Frankle, J.; Chang, S.; Liu, S.; Zhang, Y.; Wang, Z.; and Carbin, M. 2020. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33: 15834–15846.

Chen, X.; Zhang, Z.; Sui, Y.; and Chen, T. 2021. Gans can play lottery tickets too. *arXiv preprint arXiv:2106.00134*.

Dietterich, T. G. 2000. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Evcı, U.; Ioannou, Y.; Keskin, C.; and Dauphin, Y. 2022. Gradient flow in sparse neural networks and how lottery tickets win. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6577–6586.

Fedus, W.; Zoph, B.; and Shazeer, N. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Fort, S.; Hu, H.; and Lakshminarayanan, B. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

Frankle, J.; and Carbin, M. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

Frankle, J.; Dziugaite, G. K.; Roy, D.; and Carbin, M. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 3259–3269. PMLR.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

García-Martín, E.; Rodrigues, C. F.; Riley, G.; and Grahn, H. 2019. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134: 75–88.

Garipov, T.; Izmailov, P.; Podoprikhin, D.; Vetrov, D.; and Wilson, A. G. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 8803–8812.
Liu, S.; Chen, T.; Chen, X.; Shen, L.; Mocanu, D. C.; Wang, Z.; and Pechenizkiy, M. 2022. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. arXiv preprint arXiv:2202.02643.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2016. Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440.

Morcos, A.; Yu, H.; Puganini, M.; and Tian, Y. 2019. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. Advances in neural information processing systems, 32.

Mozер, M. C.; and Smolensky, P. 1989. Using relevance to reduce network size automatically. Connection Science, 1(1): 3–16.

Nagarajan, V.; and Kolter, J. Z. 2019. Uniform convergence may be unable to explain generalization in deep learning. Advances in Neural Information Processing Systems, 32.

Neyshabur, B.; Sedghi, H.; and Zhang, C. 2020. What is being transferred in transfer learning? Advances in neural information processing systems, 33: 512–523.

Ovadia, Y.; Fertig, E.; Ren, J.; Nado, Z.; Sculley, D.; Nowozin, S.; Dillon, J.; Lakshminarayanan, B.; and Snoek, J. 2019. Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; and Garnett, R., eds., Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

Patterson, D.; Gonzalez, J.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.; Texier, M.; and Dean, J. 2021. Carbon emissions and large neural network training. arXiv preprint arXiv:2104.10350.

Perrone, M. P.; and Cooper, L. N. 1992. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, Brown Univ Providence RI Inst for Brain and Neural Systems.

Polyak, B. T.; and Juditsky, A. B. 1992. Acceleration of stochastic approximation by averaging. SIAM journal on control and optimization, 30(4): 838–855.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020.

Ramanujan, V.; Wortsman, M.; Kembhavi, A.; Farhadi, A.; and Rastegari, M. 2020. What’s hidden in a randomly weighted neural network? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11893–11902.

Rame, A.; Kirchmeyer, M.; Rahier, T.; Rakotomamony, A.; Gallinari, P.; and Cord, M. 2022. Diverse Weight Averaging for Out-of-Distribution Generalization. arXiv preprint arXiv:2205.09759.

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

In this appendix, we compare the performance of Lottery Pools against the original Lottery Tickets on ImageNet and report the results in Figure 8. Overall, we notice a clear performance gain from Lottery Pools over the original LTs at a set of sparsities.

To be specific, by adopting Lottery Pools (Interpolation), we observe up to 0.55% and 0.63% improvements to the original sparse LTs on ResNet-18 and ResNet-34, respectively. Besides, the constructed stronger dense networks outperform the original dense ResNet-18 and ResNet-34 by 0.71% and 0.76%, respectively. All these results demonstrated Lottery Pools’ effectiveness on the large-scale dataset.

Figure 8: Test accuracy % of original Lottery Tickets and Lottery Pools on ImageNet.

Accuracy Heat Map of VGG-16

The accuracy heat map of VGG-16 is reported in Figure 9. Here, we observe a similar pattern to the results of Resnet-18 (Figure 1 in the main paper). First, we notice that rewinding is necessary for performance gain using weight averaging. Secondly, the averaged subnetwork’s performance highly depends on its parents’ IMP iteration adjacency. The closer IMP iterations two LTs are from, the better their averaged subnetwork performs. Under rewinding, weight averaging could achieve higher accuracy than the original LTs subnetwork at the same sparsity level if the parent subnetworks are close enough in IMP iteration.

Figure 9: Accuracy heat map of the averaged LTs on CIFAR-100. Each cell refers to the test accuracy of the averaged subnetworks using the LTs under the sparsity of X-axis and Y-axis. If the averaged subnetwork decreases in sparsity, we prune it to the higher sparsity of its parents.
Candidate Interpolation Coefficients

In Table 3, we report the Candidate Interpolation Coefficients used in the main experiments of this work. We apply 12 candidate coefficient values ranging from 0.05 to 0.95 to enlarge the optimal value searching space. In Table 4, we show the values adopted in the chapter “Extensive Analysis” of the main paper, where the effect of interpolation coefficient count on Lottery Pools’ performance is studied.

Table 3: Candidate interpolation coefficient in main experiments.

| Network       | Dataset          | Coefficient Count | Coefficient Values          |
|---------------|------------------|-------------------|----------------------------|
| ResNet-18/VGG-16 | CIFAR-10/100     | 11                | 0.05, 0.1, 0.2, 0.3, 0.4, 0.5 |
| ResNet-18/34   | ImageNet         |                   | 0.6, 0.7, 0.8, 0.9, 0.95    |

Table 4: Candidate interpolation coefficient in extensive analysis.

| Network       | Dataset          | Coefficient Count | Coefficient Values          |
|---------------|------------------|-------------------|----------------------------|
| ResNet-18/VGG-16 | CIFAR-10/100     | 1                 | 0.5                        |
| ResNet-18/VGG-16 | CIFAR-10/100     | 3                 | 0.05, 0.5, 0.95            |
| ResNet-18/VGG-16 | CIFAR-10/100     | 7                 | 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 0.95 |

Dataset Details

We reported the datasets details, including the number of classes, the size of training, validation and testing sets on CIFAR-10, CIFAR-100, and ImageNet in Table 5. As there are no publicly available labeled sets for testing in these datasets, we use original validation sets for testing, split 10% of the training sets as hold-out validation sets, and use the rest of the training sets for training.

Table 5: Datasets Details.

| Dataset     | Size of the set used for Training | Validation | Testing | Number of classes |
|-------------|-----------------------------------|------------|---------|-------------------|
| CIFAR-10    | 45,000                            | 5,000      | 10,000  | 10                |
| CIFAR-100   | 45,000                            | 5,000      | 10,000  | 100               |
| ImageNet    | 1,255,167                         | 26,000     | 50,000  | 1,000             |

Lottery Tickets Implementation Details

In Table 6 and Table 7, we report the implementation details of creating Lottery Tickets subnetworks that are used for interpolating in Lottery Pools. The reported hyperparameters include total training epochs (Epoch), learning rate (LR), batch size (BS), learning rate drop (LR Drop), weight decay (WD), SGD momentum (Momentum), IMP iteration count, IMP weight pruning fraction and the rewinding epochs, etc.

Table 6: Implementation hyperparameters of Lottery Tickets on CIFAR-10/100.

| Model | Epoch | BS  | LR  | LR Drop, Epochs | Optimizer | WD | Momentum | Warmup (epochs) | Rewinding (epochs) | IMP iterations (epochs) | Pruning Fraction (%) |
|-------|-------|-----|-----|------------------|-----------|----|----------|-----------------|---------------------|-----------------------|----------------------|
| VGG-16| 182   | 128 | 0.1 | 10x, [91, 136]   | SGD       | 0.9| 1e-4     | -               | 9                   | 19                    | 20                   |
| ResNet-18| 182 | 128 | 0.1 | 10x, [91, 136]   | SGD       | 0.9| 1e-4     | -               | 9                   | 19                    | 20                   |

Implementation details on ImageNet.

Table 7: Implementation hyperparameters of Lottery Tickets on ImageNet.

| Model | Epoch | BS  | LR  | LR Drop, Epochs | Optimizer | WD | Momentum | Warmup (epochs) | Rewinding (epochs) | IMP iterations (epochs) | Pruning Fraction (%) |
|-------|-------|-----|-----|------------------|-----------|----|----------|-----------------|---------------------|-----------------------|----------------------|
| ResNet-18| 90  | 1024| 0.4 | 10x, [30, 60, 80]| SGD       | 0.9| 1e-4     | 5               | 5                   | 5                     | 9                    |
| ResNet-34| 90  | 1024| 0.4 | 10x, [30, 60, 80]| SGD       | 0.9| 1e-4     | 5               | 5                   | 5                     | 9                    |
Algorithm of Lottery Pools (Average)

When applying a fixed interpolation value of 0.5 instead of the searched optimal one within the Candidate Coefficient Pools, the Lottery Pools (Interpolation) turns into its variant, the Lottery Pools (Average) that is described in Algorithm 3. Lottery Pools (Interpolation) tends to achieve better performance than Lottery Pools (Average) by using the searched optimal value for interpolating, while Lottery Pools (Average) is benefited from efficiency due to simple averaging.

Algorithm 3: Lottery Pools Average Recipe

**Input**: The original learned sparse subnetwork $\tilde{\theta}_t$ from LTs, Candidate Lottery Pools $S_t = \{\tilde{\theta}_{t-1}, \tilde{\theta}_{t+1}, \tilde{\theta}_{t+2}, \ldots\}$, The averaged subnetwork $\tilde{\theta}_{\text{Inter}}$ during greedy search.

**Output**: Final averaged subnetwork $\tilde{\theta}_{\text{best}}$ that has the same sparsity with $\tilde{\theta}_t$.

1: $S_t \leftarrow \{\tilde{\theta}_{t-1}, \tilde{\theta}_{t+1}, \tilde{\theta}_{t+2}, \ldots\}$  \hspace{1cm} \triangleright Create Candidate Lottery Pools, and sort all candidates by their adjacency to $\tilde{\theta}_t$
2: $\tilde{\theta}_{\text{best}} \leftarrow \tilde{\theta}_t$  \hspace{1cm} \triangleright Greedily search the candidate LTs subnetworks for averaging
3: for $\tilde{\theta}_i \in S_t$ do
4: \hspace{1cm} $\tilde{\theta}_{\text{Inter}} \leftarrow \text{MagnitudePruning}\left(\frac{\tilde{\theta}_{\text{best}} + \tilde{\theta}_i}{2}\right)$  \hspace{1cm} \triangleright Average
5: \hspace{1cm} if $\text{ValAcc}\left(\tilde{\theta}_{\text{Inter}}\right) \geq \text{ValAcc}\left(\tilde{\theta}_{\text{best}}\right)$ \hspace{1cm} \triangleright Update the best averaged subnetwork
6: \hspace{2cm} $\tilde{\theta}_{\text{best}} \leftarrow \tilde{\theta}_{\text{Inter}}$
7: end for

**Diversity analysis.**

We plot the prediction disagreement matrix across the learned subnetwork from IMP across different iterations in Figure 10. We can see that the LTs subnetworks behave similarly to their neighbor subnetworks with small prediction disagreement, and such disagreement gradually increases as their distance becomes larger and larger in the context of IMP. The results could explain the phenomena of Figure 1 as a larger diversity indicates it is more likely that the subnetworks are located in the different basins, where weight interpolation does not provide satisfactory performance (Neyshabur, Sedghi, and Zhang 2020; Yin et al. 2022).

Figure 10: Prediction disagreement between lottery tickets at the different sparsity levels. Each block in the heatmap shows the fraction of labels on which the predictions from different lottery tickets disagree.
Results Tables
Here we report the performance details of Lottery Pools, EMA, SWA and original Lottery tickets on CIFAR-10/100 in Table 8 and Table 9. The performance details on ImageNet are reported in Table 10. We apply the decay factor of 0.95 in EMA. For a fair comparison, EMA and SWA are modified by pruning the interpolated models to the sparsity of the original LHS during interpolating. The reported accuracy on CIFAR-10/100 is averaged over 3 independent runs, while we only run the experiments on ImageNet once due to the limited resources.

Compared with the original LTs, Lottery Pools achieves universal performance improvements at all sparsity levels on both ImageNet and CIFAR-10/100 datasets. Compared with the other two baselines (SW A and EMA), our Lottery Pools outperforms them across all databases.

Table 8: Accuracy (%) of Lottery Pools, the original Lottery Tickets, EMA and SWA on CIFAR-10/100 (1).

| Dataset  | Network | Method                  | Weights Remaining% |
|----------|---------|-------------------------|--------------------|
| CIFAR-10 | VGG-16  | Lottery Pools (interpolation) | 73.52 ± 0.22       |
|          |         | Lottery Pools (Average)    | 73.14 ± 0.30       |
|          |         | SWA                     | 73.14 ± 0.33       |
|          |         | EMA (0.95)               | 73.14 ± 0.33       |
|          |         | Original LTs            | 73.14 ± 0.33       |
| CIFAR-10 | ResNet-18 | Lottery Pools (interpolation) | 75.60 ± 0.06       |
|          |         | Lottery Pools (Average)    | 75.60 ± 0.06       |
|          |         | SWA                     | 75.60 ± 0.06       |
|          |         | EMA (0.95)               | 75.60 ± 0.06       |
|          |         | Original LTs            | 75.60 ± 0.06       |
| CIFAR-10 | ResNet-18 | Lottery Pools (interpolation) | 94.54 ± 0.04       |
|          |         | Lottery Pools (Average)    | 94.54 ± 0.04       |
|          |         | SWA                     | 94.54 ± 0.04       |
|          |         | EMA (0.95)               | 94.54 ± 0.04       |
|          |         | Original LTs            | 94.54 ± 0.04       |

Table 9: Accuracy (%) of Lottery Pools, the original Lottery Tickets, EMA and SWA on CIFAR-10/100 (2).

| Dataset  | Network | Method                  | Weights Remaining% |
|----------|---------|-------------------------|--------------------|
| CIFAR-10 | VGG-16  | Lottery Pools (interpolation) | 73.52 ± 0.22       |
|          |         | Lottery Pools (Average)    | 73.14 ± 0.30       |
|          |         | SWA                     | 73.14 ± 0.33       |
|          |         | EMA (0.95)               | 73.14 ± 0.33       |
|          |         | Original LTs            | 73.14 ± 0.33       |
| CIFAR-10 | ResNet-18 | Lottery Pools (interpolation) | 75.60 ± 0.06       |
|          |         | Lottery Pools (Average)    | 75.60 ± 0.06       |
|          |         | SWA                     | 75.60 ± 0.06       |
|          |         | EMA (0.95)               | 75.60 ± 0.06       |
|          |         | Original LTs            | 75.60 ± 0.06       |
| CIFAR-10 | ResNet-18 | Lottery Pools (interpolation) | 94.54 ± 0.04       |
|          |         | Lottery Pools (Average)    | 94.54 ± 0.04       |
|          |         | SWA                     | 94.54 ± 0.04       |
|          |         | EMA (0.95)               | 94.54 ± 0.04       |
|          |         | Original LTs            | 94.54 ± 0.04       |

Table 10: Accuracy (%) of Lottery Pools, the original Lottery Tickets, EMA and SWA on ImageNet.

| Dataset  | Network | Method                  | Weights Remaining% |
|----------|---------|-------------------------|--------------------|
| ImageNet | ResNet-18 | Lottery Pools (interpolation) | 70.68 ± 0.04       |
|          |         | Lottery Pools (Average)    | 70.68 ± 0.04       |
|          |         | SWA                     | 69.60 ± 0.09       |
|          |         | EMA (0.95)               | 68.36 ± 0.35       |
|          |         | Original LTs            | 69.60 ± 0.09       |
| ImageNet | ResNet-34 | Lottery Pools (interpolation) | 74.36 ± 0.24       |
|          |         | Lottery Pools (Average)    | 74.36 ± 0.24       |
|          |         | SWA                     | 72.50 ± 0.21       |
|          |         | EMA (0.95)               | 72.50 ± 0.21       |
|          |         | Original LTs            | 72.50 ± 0.21       |