Partially Frozen Random Networks Contain Compact Strong Lottery Tickets

Hikari Otsuka[1,*,†] **Daiki Chijiwa**[2,*] **Ángel López García-Arias**[2,*]

Yasuyuki Okoshi[1] **Kazushi Kawamura**[1] **Thiem Van Chu**[1] **Daichi Fujiki**[1]

Susumu Takeuchi[2] **Masato Motomura**[1]

[1]Institute of Science Tokyo, Japan, [2]NTT Corporation, Japan

Abstract

Randomly initialized dense networks contain subnetworks that achieve high accuracy without weight learning—strong lottery tickets (SLTs). Recently, Gadhikar et al. [11] demonstrated that SLTs could also be found within a randomly pruned source network. This phenomenon can be exploited to further compress the small memory size required by SLTs. However, their method is limited to SLTs that are even sparser than the source, leading to worse accuracy due to unintentionally high sparsity. This paper proposes a method for reducing the SLT memory size without restricting the sparsity of the SLTs that can be found. A random subset of the initial weights is frozen by either permanently pruning them or locking them as a fixed part of the SLT, resulting in a smaller model size. Experimental results show that `Edge-Popup` [31, 34] finds SLTs with better accuracy-to-model size trade-off within frozen networks than within dense or randomly pruned source networks. In particular, freezing 70% of a ResNet on ImageNet provides $3.3\times$ compression compared to the SLT found within a dense counterpart, raises accuracy by up to 14.12 points compared to the SLT found within a randomly pruned counterpart, and offers a better accuracy-model size trade-off than both.

1 Introduction

The Strong Lottery Ticket Hypothesis (SLTH) conjectured the existence of subnetworks within a randomly weighted network—*strong lottery tickets* (SLTs)—that achieve comparable accuracy to trained dense networks [42, 31, 26]. The existence of such subnetworks that do not require weight training, illustrated in Figure 1 (left), has been demonstrated experimentally [42, 31, 18, 22, 40] and proven theoretically for dense source networks [26, 28, 29, 7, 6, 1, 3, 9, 11]. They appear in networks with excessive amounts of parameters, and their existence indicates the possibility of new optimization strategies for deep neural networks based on a network connectivity perspective.

SLTs offer a particularly advantageous opportunity for specialized inference hardware [15, 5, 4] since the random weights can be reconstructed from the seed (i.e., they do not need to be memorized), and the binary mask—*supermask*—can be greatly compressed with entropy coding, vastly reducing off-chip memory access and its associated overheads. Furthermore, the binary nature of both the supermask and the random weights can be exploited for multiplier-less execution, as demonstrated practically by Hirose et al. [15].

*Equal contribution. † Correspondence to: Hikari Otsuka <otsuka.hikari@artic.iir.titech.ac.jp>.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

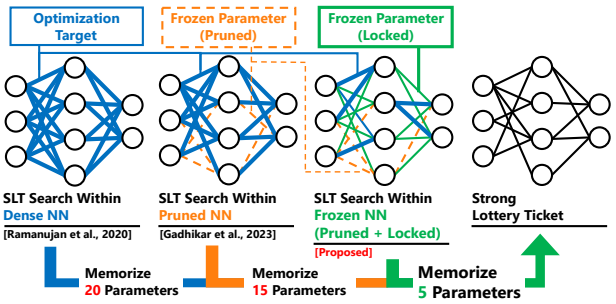


Figure 1: Freezing the source network by randomly pruning some parameters and locking others reduces the memorized supermask for finding an SLT.

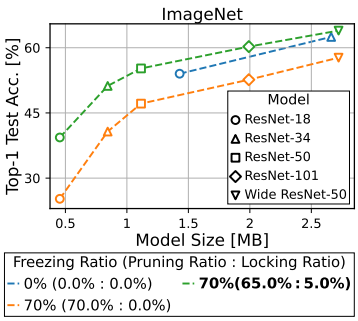


Figure 2: Freezing (●) improves the accuracy-to-model size trade-off over pre-pruning only (●) or non-freezing (●).

Recently, Gadhikar et al. [11] showed that accurate SLTs could be found even if the number of edges to be optimized was reduced by randomly pruning the source network at initialization (see Figure 1, center). Since the random pre-pruning mask can be reconstructed from the seed in the same way as the random weights, and thus there is no need to store the part of the supermask corresponding to pre-pruned weights, their approach can be exploited to further reduce the memory cost required by the SLT in specialized hardware. However, when aiming for high compression, their method can only search for SLTs in the relatively high sparsity region, and may even lead to layer collapse [12, 35]. Furthermore, since the SLT sparsity regions where highly accurate SLTs exist depend on the dataset and network architecture, a search limited to sparse regions may fail to find accurate SLTs. For instance, work on SLTs within graph neural networks has shown that dense SLTs are more accurate than sparse ones in some settings [18, 39]. Therefore, it would be desirable to use a method that allows for increased randomness in SLTs for further model compression, but has the freedom to allocate it in the SLT sparsity regions that lead to more accurate tickets.

This paper introduces such a novel method to reduce the memory cost of the optimized supermask without restricting the desired sparsity of SLTs to be searched for: in addition to random pruning at initialization, it also locks randomly chosen parameters at initialization to be a permanent part of the SLT (i.e., never pruned), as exemplified in Figure 1 (right). Both the randomly pruned and the locked parameters—the *frozen* parameters—are left completely random and can be regenerated from seed. The weights corresponding to the optimized supermask region (less than 50% of the total) are also reconstructed from seed, and the supermask is binary and sparse, producing a highly compressible model. Far from negatively impacting performance, this cost reduction is efficient: as shown in Figure 2, the frozen SLTs achieve a higher accuracy than SLTs with a similar size resulting from conventional methods. The contributions of this paper are summarized as follows:

- We propose a novel method that vastly reduces the number of parameters to be memorized for finding an accurate SLT by freezing (pruning and locking) the source random network.
- We experimentally validate our method in three scenarios corresponding to low, medium, and high optimal SLT sparsity regions, which reveal that parameter freezing consistently produces smaller yet accurate supermasks. Even with randomly frozen parameters, we find highly accurate SLTs that cannot be found within dense networks for some desired sparsities.
- Furthermore, the experimental results show that SLTs found in frozen networks achieve comparable or better accuracy-to-model size trade-off than SLTs found within dense (non-freezing) or sparse (non-locking) random networks.

As mentioned above, SLTs are quite attractive for neural engine design, as they can vastly reduce the memory size for model storage, meaning that off-chip memory access—by far the major bottleneck of energy and time consumption [16]—can be drastically reduced for energy-efficient inference acceleration [15]. Our contributions have the potential to reduce off-chip memory access further and to make inference more energy-efficient than previous designs.

2 Strong Lottery Tickets in Frozen Networks

Recently, Gadhikar et al. [11] revealed that SLTs exist within not only dense but also sparse source networks, i.e., random networks that have been *randomly pruned at initializaton* (see Figure 1, center). Their approach offers a practical advantage: as the randomly pre-pruned parts can be reconstructed

from the seed, this pre-processing of the source network before finding SLTs can be exploited to reduce the cost of storing the supermask. (For more detailed preliminaries, see Appendix A)

However, this approach also imposes the limitation that it can only search for SLTs with a sparsity higher than the pre-pruning ratio. For example, a random pre-pruning ratio of 90% vastly reduces the memory size of the supermask, but it limits the search to only SLTs sparser than 90%. Therefore, their method is incompatible with exploring the whole SLT sparsity range for optimal accuracy.

This paper proposes a novel method that allows us to find SLTs within the optimal sparsity range while still capitalizing on the compression gains offered by random connectivity initialization. We explore the performance of SLTs within a *frozen source network*, i.e., a random network that (in addition to randomly pruned parameters) has randomly chosen parameters forced to be permanent part (i.e., never pruned) of the SLT—*locked parameters* (see Figure 1, right). Since the random locking pattern can be reconstructed from seed in the same way as the random pre-pruning pattern, our freezing method allows us to compress SLTs further. Additionally, locking allows us to extend the benefits of pre-pruning to the scenarios where the SLT sparsity should not be too high, e.g., as found in graph neural networks [18, 39].

This section first describes the construction of frozen networks and the freezing pattern encoding for model compression. Then, we perform a preliminary experiment of the optimal settings for the proposed method in preparation for the evaluation experiments in Section 3.

2.1 Partial Freezing for Enhanced SLT Compression

In our method, since frozen regions in the source network are completely random and thus can be reconstructed from seed, we need to memorize only the optimized supermask region. The total amount of frozen parameters is determined by a global *freezing ratio* F_r, which is the sum of the respective *pre-pruning ratio* P_r and *locking ratio* L_r. Therefore, we optimize the $1{-}F_r$ of the parameters (i.e., non-frozen parameters) of the original dense network. As explained previously, it is not possible to search for an SLT with a lower sparsity (denser) than the frozen source network. Consequently, as shown in Figure 3, the pre-pruning ratio P_r sets the lower bound of the sparsity of the SLTs that can be found. On the other hand, it is not possible to search for an SLT that prunes more parameters than those available, so the locking ratio L_r sets the upper bound of the sparsity of the SLTs that can be found. (Specifically, the upper bound is given by $1-L_r$.) Thus, these ratios allow us to freely control the proportion of pre-pruning and locking ratios and memory size reduction ratio for the desired SLT sparsity.

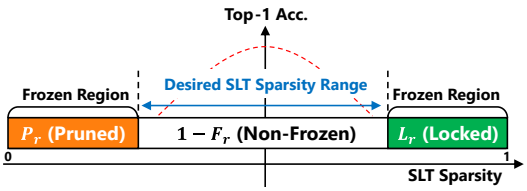


Figure 3: Pre-pruning and locking set the bounds of the SLT sparsity that can be found. These optimal bounds are investigated in Section 2.2.

Frozen network construction: We assume that the l-th layer of network with depth L has weights $\boldsymbol{w}^{(l)} \in \mathbb{R}^{n^{(l)}}$. The number of frozen parameters is determined by the layer-wise *freezing ratios* $p_f^{(1)}, ..., p_f^{(L)}$, where each $p_f^{(l)}$ is defined as the sum of the layer-wise *pre-pruning ratio* $p_p^{(l)}$ and the layer-wise *locking ratio* $p_l^{(l)}$, and $p_f^{(l)}, p_p^{(l)}, p_l^{(l)} \in [0, 1]$.

To prune $p_p^{(l)} n^{(l)}$ parameters and lock $p_l^{(l)} n^{(l)}$ parameters, we generate two random masks, a pre-pruning mask $\boldsymbol{m}_p^{(l)} \in \{0,1\}^{n^{(l)}}$ and a locking mask $\boldsymbol{m}_l^{(l)} \in \{0,1\}^{n^{(l)}}$, so that they satisfy $||1 - \boldsymbol{m}_p^{(l)}|| = p_p^{(l)} n^{(l)}$, $||\boldsymbol{m}_l^{(l)}|| = p_l^{(l)} n^{(l)}$, and $(1 - \boldsymbol{m}_p^{(l)}) \cdot \boldsymbol{m}_l^{(l)} = 0$ (i.e., we require pre-pruning and locking to be implemented without overlap). The layer-wise weights frozen with these masks are calculated as $\boldsymbol{w}_f^{(l)} := (\boldsymbol{m}_p^{(l)} \odot (1 - \boldsymbol{m}_l^{(l)}) + \boldsymbol{m}_l^{(l)}) \odot \boldsymbol{w}^{(l)}$. These masks are fixed during training, and we search for SLTs only in the parts where $\boldsymbol{m}_p^{(l)} \odot (1 - \boldsymbol{m}_l^{(l)})$ is one.

Setting the layer-wise ratios: Our method considers the following two existing strategies for determining the layer-wise pre-pruning ratio from the desired global pre-pruning ratio of the network:

- *Erdős-Rényi-Kernel* (ERK): The pre-pruning ratio of layer l is proportional to the scale $(C_{in}^{(l)} + C_{out}^{(l)} + k_h^{(l)} + k_w^{(l)})/(C_{in}^{(l)} \cdot C_{out}^{(l)} \cdot k_h^{(l)} \cdot k_w^{(l)})$, where $C_{in}^{(l)}$, $C_{out}^{(l)}$, $k_h^{(l)}$, and $k_w^{(l)}$ denote input channels, output channels, kernel height, and kernel width of the layer l, respectively [8].
- *Edge Per Layer* (EPL): Each layer's pre-pruning ratio is set so that they all have the same number of remaining weights [30, 11].

The same strategy is employed to determine the layer-wise pre-pruning and freezing ratios from their respective global ratios, and then the locking ratios are calculated as the difference between the corresponding freezing and pre-pruning ratios.

Freezing pattern encoding for model compression: The freezing pattern can be encoded during inference as a ternary mask—a *freezing mask*—that indicates whether a parameter is pruned, locked, or part of the supermask. For example, encoding pre-pruning as -1, locking as $+1$, and supermask inclusion as 0, the layer-wise freezing mask can be encoded as $\boldsymbol{m}_l^{(l)} + (\boldsymbol{m}_p^{(l)} - 1) \in \{-1, 0, 1\}^{n^{(l)}}$. Since this freezing mask is also random and fixed, it can be regenerated from its seed and ratios, similarly to the random weights. Furthermore, the supermask size is reduced by excluding from it the frozen parameters, so the SLTs found by this method can be compressed in inference to an even smaller memory size than those produced by the existing SLT literature [15, 27, 22], reducing costly off-chip memory access on specialized neural inference accelerators [15, 5], and thus offering an opportunity to perform even faster and more energy-efficient inference processing.

2.2 Optimal Pruning:Locking Propotion for Freezing

Here, we perform a preliminary investigation of the optimal pruning:locking proportion for each given desired SLT sparsity by varying the proportion with a fixed freezing ratio of the network. Figure 4 explores different configurations of a 80% freezing ratio—the situation where the supermask memory size is 20% of the SLT in the dense source network—on a Conv6 network and compares the performance of the found SLTs on CIFAR-10. As expected, the best-performing SLT of each prune:lock configuration is found at the center of the non-frozen region, where the number of candidate subnetworks is maximized. In other words, when the non-frozen region accounts for $S \in [0, 1]$ of the entire source network, the optimal SLT sparsity is $k = P_r + S/2$, where $P_r = \sum_l p_p^{(l)} n^{(l)} / \sum_k n^{(k)}$ is the global pre-pruning ratio of the network. Conversely, for a given freezing ratio F_r of the network and a desired SLT sparsity k, the optimal position of the frozen region is set by the pre-pruning ratio $P_r = k - (1 - F_r)/2$ and the corresponding locking ratio $L_r = F_r - P_r$. In the cases where this would result in $P_r < 0$ or $L_r < 0$, we choose a best-effort approach that keeps the desired freezing ratio and sets the bounds to $P_r = 0$ or $L_r = 0$, respectively.

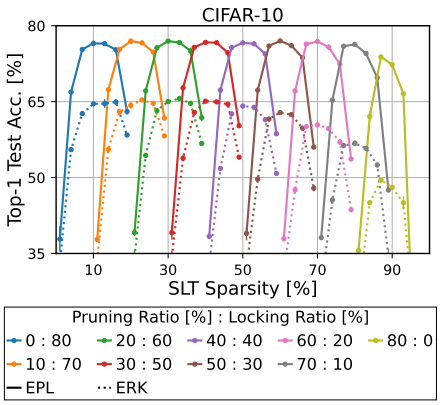


Figure 4: Different prune:lock proportions of a 80% freezing ratio using a Conv6.

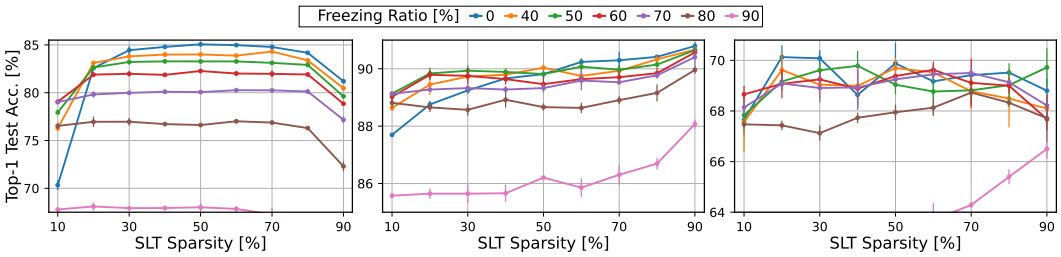
Additionally, Figure 4 compares the two strategies for setting ratios considered in Section 2.1, showing that EPL outperforms ERK in all cases. Consequently, hereafter, the proposed method sets the global ratios in order to position the frozen region center as close as possible to the desired SLT sparsity, and then sets the layer-wise ratios using EPL.

2.3 SLT Existence in Frozen Networks

One question comes to mind here: *does SLT exist that approximates a given target network, even if the parameters are randomly frozen (i.e., pruned or locked) in advance?* It has been shown by Gadhikar et al. [11] that an SLT that approximates a given target network exists in the pre-pruned source network if the source network is sufficiently wider and deeper than the target, but it is not known whether such an SLT exists in the situation of a frozen source network. Here, we provide a theoretical result indicating that an SLT capable of approximating a target network exists in a frozen network. This result is proved by extending the subset-sum approximation lemma (Lemma A.1) to the case where some parameters are locked (for detailed proof, see Appendix B).

Lemma 2.1 (Subset-Sum Approximation in Randomly Locked Networks)**.** *Let $X_1, ..., X_n$ be as in Lemma A.1, and $M'_1, ..., M'_n \sim Ber(q)$ be independent, Bernoulli distributed random variables with $q \in (0, 1)$. Then, except with exponentially small probability, any $z \in [-1, 1]$ can be approximated by the sum of $\sum_{i=1}^{n} M'_i X_i$ and a subset-sum of $(1 - M'_i)X_i$ if n is sufficiently large.*

Then, by combining Lemma 2.1 and Lemma A.2, we extend the subset-sum approximation to the situation where some random variables are frozen (i.e., pruned or locked).



(a) Conv6 (CIFAR-10) (b) ResNet-18 (CIFAR-10) (c) GIN (OGBN-Arxiv)

Figure 5: Impact of the freezing ratio on different architectures. Pruning and locking ratios are set following Section 2.2.

Lemma 2.2 (Subset-Sum Approximation in Frozen Networks)**.** *Let $X_1, ..., X_n$ be as in Lemma A.1, $M_1, ..., M_n \sim Ber(p)$ be as in Lemma A.2, and $M'_1, ..., M'_n \sim Ber(q)$ be as in Lemma 2.1. Then, except with exponentially small probability, any $z \in [-1, 1]$ can be approximated by the sum of $\sum_{i=1}^{n} M_i M'_i X_i$ and a subset-sum of $M_i(1 - M'_i)X_i$ if n is sufficiently large.*

Finally, by applying Lemma 2.2 to the Theorem 2.5 in Gadhikar et al. [11] instead of Lemma A.2, it follows that an SLT approximating a target network exists within a frozen network.

Theorem 2.3 (SLT Existence in Frozen Networks)**.** *Let a target network f_T with depth L and a partially frozen source network f_S with depth $L+1$ be given. Assume that the source network is randomly frozen with pruning ratio p_l and locking ratio q_l for each l-th layer. Also assume that these networks use the ReLU activation function and are initialized with a uniform distribution $U[-1, 1]$. Then, except with exponentially small probability, a subnetwork f_{SLT} exists in the frozen source network f_S such that f_{SLT} approximates the target network f_T if the width of f_S is sufficiently large.*

3 Experiments

This section demonstrates that freezing reduces the SLT memory size in a broad range of situations by evaluating it on image classification and graph node classification. We evaluate the impact of the freezing ratio on various network architectures, identifying three scenarios. Then, we explore trade-offs between accuracy and model memory size for different network widths and architectures. The detailed experimental setting and additional experiments are in Appendix C.1.

3.1 Varying Desired SLT Sparsity at Different Freezing Ratios

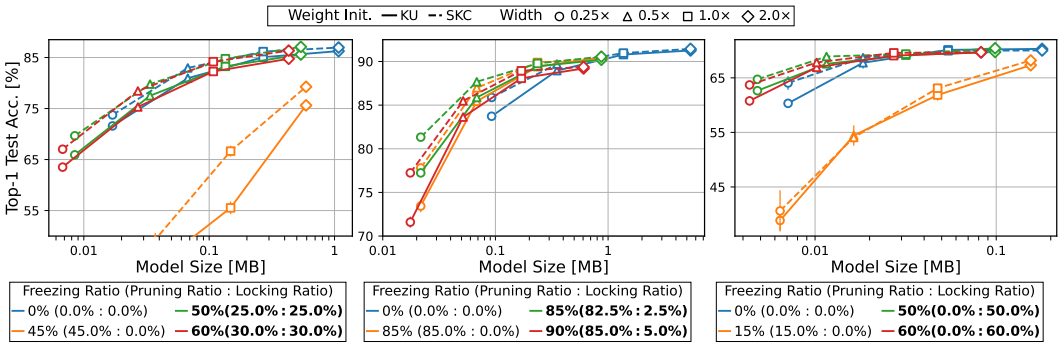
This section investigates the effect of the freezing ratio of the network on desired SLT accuracy. We identify three scenarios, represented in Figure 5.

In the cases where the optimal SLT sparsity is found at intermediate sparsity—e.g., around 50% for Conv6 in Figure 5a—pruning and locking can be applied with equally high ratios. Results show that the supermask memory size can be reduced by 40% with a small impact on accuracy, and by 70% with still a moderate accuracy drop of 5 points.

Applying the much larger ResNet-18 to the same task results in much stronger overparametrization. Therefore, optimal SLTs are found in the higher sparsity range, as revealed by Figure 5b, benefiting from much higher pruning than locking. Even though 90% of the memory size is reduced in this scenario, we can find 90% sparse SLTs with 88.1% accuracy.

As an example of the scenario where optimal SLT sparsities are found in the denser range, benefiting from a higher locking ratio, we evaluate GIN in Figure 5c. Compared with the best-performing SLT found in the dense GIN, of 70.1% accuracy and 20% sparsity, by freezing 50% of the memory size, our method finds similarly performing SLTs of 69.8% accuracy with 40% sparsity.

Interestingly, with low SLT sparsity (e.g., 10% sparsity) in all three scenarios, despite the reduced parameters to be optimized, SLTs within frozen networks achieve higher accuracy than SLTs within dense networks. These results imply that parameter freezing at an appropriate ratio has the effect of avoiding the inclusion of low-grade local optimal solutions in the search space. While searching for SLTs in a dense network by `Edge-Popup` leads to convergence to a local optimal solution [10], a moderate random parameter freezing may reduce the number of less accurate local optimal solutions and facilitate convergence to a more accurate local optimal solution within the reduced search space. In other words, we conjecture that if the network is properly frozen, the local optimal solution for the reduced search space is close to the global optimal solution of the entire search space.



(a) Conv6 (SLT Sparsity 50%) (b) ResNet-18 (SLT Sparsity 90%) (c) GIN (SLT Sparsity 20%)

Figure 6: Compared to sparse (●) or dense (●) source networks, freezing achieves better accuracy-to-model memory size trade-off (top-left is better).

3.2 Accuracy-to-Model Memory Size Trade-Off

The freezing mask compression scheme proposed in Section 2.1 allows to reduce the model memory size during inference by regenerating both the random weights and the freezing mask with random number generators. Here we consider this compression and investigate the accuracy-to-model size trade-off offered by the SLTs found in the frozen networks. Model size refers to the total memory size of model parameters that need to be stored. The random weights and the frozen parts of the supermask are excluded, since they can be regenerated from seed, whereas each non-frozen element of the supermask and each learned batchnorm parameter are counted as 1 and 32 bits, respectively. Furthermore, we also compare the Kaiming Uniform (KU) weight initialization used so far with the binary weights provided by the Signed Kaiming Constant (SKC) initialization [31], which can be exploited for reduced computational cost in neural engines [15]. For SKC, we scale weights by $1/\sqrt{1-k_l}$, where k_l is the sparsity of each layer, as proposed by Ramanujan et al. [31].

Figure 6 explores varying the width of the source network to analyze its impact on accuracy and model size. SLT sparsity is fixed to that of the best performing SLT found in a dense source network in Figure 5: 50% in Conv6, 90% in ResNet-18, and 20% in GIN. Compared to the SLTs found in dense or sparse source networks, SLTs found by our method achieve similar or higher accuracy for similar or smaller model size, thus improving the accuracy-to-model size trade-off in all scenarios.

Empirically, it is known that SLTs within an SKC-initialized dense network achieve better performance than with continuous random weights [31, 27, 22, 39]. Our results show that such a trend can also be observed in frozen source networks. Nonetheless, in all source networks, we find that this improvement is smaller the wider the source network is, suggesting that the requirement for source networks of larger width is weaker in the case of binary weights.

3.3 ImageNet Experiments

Finally, we evaluate our method using larger models on a large-scale dataset: deeper and wider ResNets on ImageNet. Since SLTs with 80% sparsity achieve the highest accuracy in a dense source ResNet-50 (for details, see Appendix C.2), we compare the three methods using 80% SLT sparsity.

Figure 2 compares the accuracy-to-model size trade-off in finding the 80% sparsity SLTs between the proposed and conventional methods with SKC using ImageNet. Despite the more challenging setting and the significant 70% memory size reduction, our method (green) finds SLTs that are more accurate than pre-pruning-only methods (orange) for the same model size. This result demonstrates that the effective combination of parameter pruning and locking at initialization can improve the SLT memory efficiency even on large-scale datasets and models.

4 Conclusion

This paper capitalizes on the fact that pre-pruning a randomly weighted network reduces the supermask memory size, but identifies that doing so limits the search to a suboptimal sparsity region. This problem is tackled by freezing (i.e., pruning or locking) some parameters at initialization, excluding them from the search. Freezing allows to the search for SLTs in the optimal sparsity region, while further reducing the model size. Experimental results show that SLTs found in frozen networks improve the accuracy-to-model size trade-off compared to SLTs found in dense [31, 34] or sparse networks [11]. Interestingly, although for weight training random parameter locking has only been

found useful for reducing accuracy degradation in network compression [42, 37], we identify scenarios in SLT training where they can be used for raising accuracy. Our method can be interpreted as being capable of generating more useful SLT information from a random seed than previous methods, offering an opportunity for reducing off-chip memory access in specialized SLT accelerators [15]. Additionally, the reduced number of parameters to be optimized may be exploited for training cost reduction, which remains for future work.

Acknowledgments and Disclosure of Funding

This work was supported in part by JSPS KAKENHI Grant Number JP23H05489.

References

[1] Rebekka Burkholz. Most activation functions can win the lottery without excessive depth. *Proc. Adv. Neural Inform. Process. Syst.*, 35:18707–18720, 2022.

[2] Rebekka Burkholz. Convolutional and residual networks provably contain lottery tickets. In *Proc. Int. Conf. Mach. Learn.*, pages 2414–2433. PMLR, 2022.

[3] Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. In *Proc. Int. Conf. Learn. Repr.*, 2022.

[4] Yiming Chen, Guodong Yin, Mingyen Lee, Wenjun Tang, Zekun Yang, Yongpan Liu, Huazhong Yang, and Xueqing Li. Hidden-ROM: A compute-in-rom architecture to deploy large-scale neural networks on chip with flexible and scalable post-fabrication task transfer capability. In *Proc. IEEE Int. Conf. Comput.-Aided Design*, pages 1–9, 2022.

[5] Yung-Chin Chen, Shimpei Ando, Daichi Fujiki, Shinya Takamaeda-Yamazaki, and Kentaro Yoshioka. HALO-CAT: A hidden network processor with activation-localized CIM architecture and layer-penetrative tiling. *arXiv preprint arXiv:2312.06086*, 2023.

[6] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the lottery ticket hypothesis for convolutional neural networks. In *Proc. Int. Conf. Learn. Repr.*, 2021.

[7] James Diffenderfer and Bhavya Kailkhura. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. In *Proc. Int. Conf. Learn. Repr.*, 2021.

[8] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proc. Int. Conf. Mach. Learn.*, pages 2943–2952. PMLR, 2020.

[9] Damien Ferbach, Christos Tsirigotis, Gauthier Gidel, and Joey Bose. A general framework for proving the equivariant strong lottery ticket hypothesis. In *Proc. Int. Conf. Learn. Repr.*, 2023.

[10] Jonas Fischer and Rebekka Burkholz. Plant 'n' seek: Can you find the winning ticket? In *Proc. Int. Conf. Learn. Repr.*, 2022.

[11] Advait Harshal Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why random pruning is all we need to start sparse. In *Proc. Int. Conf. Mach. Learn.*, pages 10542–10570. PMLR, 2023.

[12] Soufiane Hayou, Jean-Francois Ton, Arnaud Doucet, and Yee Whye Teh. Pruning untrained neural networks: Principles and analysis. *arXiv preprint arXiv:2002.08797*, 2020.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1026–1034, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, pages 770–778, 2016.

[15] Kazutoshi Hirose, Jaehoon Yu, Kota Ando, Yasuyuki Okoshi, Ángel López García-Arias, Junnosuke Suzuki, Thiem Van Chu, Kazushi Kawamura, and Masato Motomura. Hiddenite: 4K-PE hidden network inference 4D-tensor engine exploiting on-chip model construction achieving 34.8-to-16.0 TOPS/W for CIFAR-100 and ImageNet. In *Proc. IEEE Int. Solid-State Circuits Conf.*, volume 65, pages 1–3. IEEE, 2022.

[16] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Proc. Adv. Neural Inform. Process. Syst.*, 33:22118–22133, 2020.

[18] Tianjin Huang, Tianlong Chen, Meng Fang, Vlado Menkovski, Jiaxu Zhao, Lu Yin, Yulong Pei, Decebal Constantin Mocanu, Zhangyang Wang, Mykola Pechenizkiy, and Shiwei Liu. You can have better graph neural networks by not training weights at all: Finding untrained GNNs tickets. In *Learn. of Graphs Conf.*, 2022.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.*, pages 448–456. pmlr, 2015.

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, Toronto, 2009.

[21] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proc. AAAI Conf. on Artif. Intell.*, volume 32, 2018.

[22] Ángel López García-Arias, Yasuyuki Okoshi, Masanori Hashimoto, Masato Motomura, and Jaehoon Yu. Recurrent residual networks contain stronger lottery tickets. *IEEE Access*, 11: 16588–16604, 2023. doi: 10.1109/ACCESS.2023.3245808.

[23] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proc. Int. Conf. Learn. Repr.*, 2017.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. Int. Conf. Learn. Repr.*, 2019.

[25] George S Lueker. Exponentially small bounds on the expected optimum of the partition and subset sum problems. *Random Struct. Algor.*, 12(1):51–62, 1998.

[26] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *Proc. Int. Conf. Mach. Learn.*, pages 6682–6691. PMLR, 2020.

[27] Yasuyuki Okoshi, Ángel López García-Arias, Kazutoshi Hirose, Kota Ando, Kazushi Kawamura, Thiem Van Chu, Masato Motomura, and Jaehoon Yu. Multicoated supermasks enhance hidden networks. In *Proc. Int. Conf. Mach. Learn.*, pages 17045–17055, 2022.

[28] Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Proc. Adv. Neural Inform. Process. Syst.*, 33:2925–2934, 2020.

[29] Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Proc. Adv. Neural Inform. Process. Syst.*, 33:2599–2610, 2020.

[30] Ilan Price and Jared Tanner. Dense for the price of sparse: Improved performance of sparsely initialized networks via a subspace offset. In *Proc. Int. Conf. Mach. Learn.*, pages 8620–8629. PMLR, 2021.

[31] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, pages 11893–11902, 2020.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115:211–252, 2015.

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

[34] Kartik Sreenivasan, Jy-yong Sohn, Liu Yang, Matthew Grinde, Alliot Nagle, Hongyi Wang, Eric Xing, Kangwook Lee, and Dimitris Papailiopoulos. Rare gems: Finding lottery tickets at initialization. *Proc. Adv. Neural Inform. Process. Syst.*, 35:14529–14540, 2022.

[35] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Proc. Adv. Neural Inform. Process. Syst.*, 33:6377–6389, 2020.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proc. Adv. Neural Inform. Process. Syst.*, 30, 2017.

[37] Paul Wimmer, Jens Mehnert, and Alexandru Condurache. Freezenet: Full performance by reduced storage costs. In *Proc. Asian Conf. Comput. Vis.*, 2020.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc. Int. Conf. Learn. Repr.*, 2019.

[39] Jiale Yan, Hiroaki Ito, Ángel López García-Arias, Yasuyuki Okoshi, Hikari Otsuka, Kazushi Kawamura, Thiem Van Chu, and Masato Motomura. Multicoated and folded graph neural networks with strong lottery tickets. In *Learn. of Graphs Conf.*, 2023.

[40] Sangyeop Yeo, Yoojin Jang, Jy-yong Sohn, Dongyoon Han, and Jaejun Yoo. Can we find strong lottery tickets in generative models? In *Proc. AAAI Conf. on Artif. Intell.*, volume 37, pages 3267–3275, 2023.

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

[42] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Proc. Adv. Neural Inform. Process. Syst.*, 32, 2019.

[43] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, pages 3599–3608, 2021.

A Preliminaries

This section outlines the background of strong lottery tickets (SLTs) within dense or sparse networks and SLT search algorithms.

A.1 Strong Lottery Tickets in Dense Networks

SLTs [42, 31, 26] are subnetworks within a randomly weighted neural network that achieve high accuracy *without any weight training*. Compared with learned dense weight models, SLTs can be reconstructed from a small amount of information: since the random weights can be regenerated from their seed, it is only necessary to store the binary supermask and the seed [15]. SLT search algorithms for deep neural networks [42, 31, 43, 34] find SLTs by updating weight scores, which are then used to generate the supermask, instead of updating weights. For example, the `Edge-Popup` algorithm [31, 34] used in this paper finds SLTs by applying a supermask generated from the connections with the top-$k\%$ scores. Such as this top-k, these methods determine the SLT sparsity to be explored as a hyperparameter explicitly or implicitly.

SLT Existence via Subset-Sum Approximation Based on the subset-sum approximation (Lemma A.1), which was first introduced into the SLT context by Pensia et al. [29], previous works [1, 2, 6, 29] showed that an SLT that approximates an arbitrary target network exists in a dense source network if it is logarithmically wider and constantly deeper than the target network. In particular, Burkholz [1] proved that a source network with depth $L+1$ and larger width than the target network contains an SLT that can approximate the target network with depth L. Given a set of random variables and a target value, the subset-sum problem consists of finding a subset whose total value approximates the target. Lueker [25] showed that such a subset exists with high probability if the number of random variables is sufficiently large:

Lemma A.1 (Subset-Sum Approximation [25])**.** *Let $X_1, ..., X_n \sim U(-1, 1)$ be independent, uniformly distributed random variables. Then, except with exponentially small probability, any $z \in [-1, 1]$ can be approximated by a subset-sum of X_i if n is sufficiently large.*

A.2 Strong Lottery Tickets in Sparse Networks

Recently, Gadhikar et al. [11] revealed that SLTs also exist within sparse source networks, i.e., random networks that have been *randomly pruned at initialization* (see Figure 1, center). They showed its existence experimentally as well as theoretically, as outlined later: an SLT that approximates a given target network exists with high probability in a sparse source network that is sufficiently wider and deeper than the target.

SLT Existence in Sparse Networks To prove the existence of SLTs within sparse networks, Gadhikar et al. [11] extended the subset-sum approximation (Lemma A.1) to the situation where randomly chosen variables are permanently pruned at initialization:

Lemma A.2 (Subset-Sum Approximation in Sparse Networks [11])**.** *Let $X_1, ..., X_n$ be as in Lemma A.1, and $M_1, ..., M_n \sim Ber(p)$ be independent, Bernoulli distributed random variables with $p \in (0, 1)$. Then, except with exponentially small probability, any $z \in [-1, 1]$ can be approximated by a subset-sum of $M_i X_i$ if n is sufficiently large.*

By applying this extended lemma instead of Lemma A.1 to the SLT existence theorem presented by Burkholz [1], Gadhikar et al. [11] proved the SLT existence in sparse networks as Theorem B.1. Thus, by extending the subset-sum approximation lemma, the conventional SLT existence proof can be easily extended to various network settings.

B Proof for Strong Lottery Tickets (SLTs) Existence in Frozen Networks

This section describes the proofs of the lemma and theorem introduced in the manuscript. For reference, we present a theorem presented by Gadhikar et al. [11] in advance.

Theorem B.1 (SLT Existence in Sparse Networks [11])**.** *Let input data $\mathcal{D}$, a target network f_T with depth L, and a source network f_S with depth $L+1$, parameters $\boldsymbol{\theta}_S$, and edge probabilities p_l be given.*

Assume that these networks have ReLU activation function, and each element of $\boldsymbol{\theta}_S$ is uniformly distributed over $[-1, 1]$. Then, with probability at least $1 - \delta$, there exists a binary mask $\boldsymbol{m}_S$ so that each output component i is approximated as $\max_{\boldsymbol{x}\in\mathcal{D}} \|f_{T,i}(\boldsymbol{x}) - f_{S,i}(\boldsymbol{x}; \boldsymbol{\theta}_S \odot \boldsymbol{m}_S)\| < \epsilon$ if

$$n_{S,l} \geq C \frac{n_{T,l}}{\log(1/1 - p_{l+1})} \log\left(\frac{1}{\min\{\epsilon_l, \delta/\rho\}}\right) \tag{1}$$

for $l \geq 1$,where

$$\epsilon_l \coloneqq \frac{\epsilon}{n_{T,L}L}\left[(1 + B_{l-1}\left(1 + \frac{\epsilon}{L}\right)\prod_{k=l+1}^{L-1}\left(\left\|\boldsymbol{\theta}_T^{(l)}\right\|_\infty + \frac{\epsilon}{L}\right)\right]^{-1}, \tag{2}$$

$$B_l \coloneqq \sup_{\boldsymbol{x}\in\mathcal{D}} \left\|\boldsymbol{x}_T^{(l)}\right\|_1, \tag{3}$$

$$\rho = \frac{C N_T^{1+\gamma}}{\log\left(1/(1 - \min_l\{p_l\})\right)^{1+\gamma}} \log\left(\frac{1}{\min\{\min_l\{\epsilon_l\}, \delta\}}\right) \tag{4}$$

for any $\gamma \geq 0$. Moreover, we need to satisfy

$$n_{S,0} \geq Cd \log\left(\frac{1}{\min\{\epsilon_1, \delta/\rho\}}\right), \tag{5}$$

where $C > 0$ denotes a generic constant that is independent of $n_{T,l}, L, p_l, \delta$, and ϵ.

We first extend the lemma of the subset-sum approximation shown by Lueker [25] to the case where some random variables are always included in the subset-sum.

Lemma B.2 (Subset-Sum Approximation in Randomly Locked Networks)**.** *Let $X_1, \cdots, X_n \sim U(-1, 1)$ be independent, uniformly distributed random variables, and $M_1, \cdots, M_n \sim \mathrm{Ber}(q)$ be independent, Bernoulli distributed random variables with $q \in (0, 1)$. Let $\varepsilon, \delta > 0$. Then, with probability at least $1 - \delta$, for any $z \in [-1, 1]$, there exists indices $I \subset \{1, \cdots, n\}$ such that $|z - \sum_{i=1}^n M_i X_i - \sum_{i\in I}(1 - M_i)X_i| \leq \varepsilon$ if*

$$n \geq C \log\left(\frac{8}{\varepsilon\delta}\right), \tag{6}$$

where $C > 0$ is a constant.

Proof. Let $M_1, \cdots, M_n \sim \mathrm{Ber}(q)$ and $m \coloneqq \sum_i M_i$. By Hoeffding's inequality, we have

$$\mathbb{P}\left(|m - qn| \leq \varepsilon_M n\right) \geq 1 - 2\exp\left(-\frac{\varepsilon_M^2 n}{2q(1-q)}\right) \tag{7}$$

for $\varepsilon_M > 0$. Thus, if we set $n \geq \frac{2q(1-q)\log(2/\delta)}{\varepsilon_M^2}$ and $\varepsilon_M \coloneqq \frac{\min(q,1-q)}{2}$, we have

$$(2q - \beta)n \leq m \leq \beta n \tag{8}$$

with $\beta \coloneqq q + \varepsilon_M = \min(\frac{3q}{2}, \frac{1+q}{2}) \in (0, 1)$, with probability at least $1 - \delta$. In particular, the number of non-vanishing terms in the sum $\sum_{i\in I}(1 - M_i)X_i$ is $n - m \geq (1 - \beta)n$ as long as each X_i is non-zero.

Now fix $M_1, \cdots, M_n \sim \mathrm{Ber}(q)$ with $(2q - \beta)n \leq m \leq \beta n$. The goal is to approximate $z \in [-1, 1]$ and $\sum_{i=1}^n M_i X_i$ by subset sum from $\{(1 - M_i)X_i : M_i = 0\}$. For simplicity, we split it into two parts:

$$\{(1 - M_i)X_i : M_i = 0\} = \{X_1', \cdots, X_{n_1}'\} \cup \{X_1'', \cdots, X_{n_2}''\}, \tag{9}$$

where the former part is used for approximating $z \in [-1, 1]$ and the latter part for approximating $\sum_{i=1}^n M_i X_i$.

To approximate $z \in [-1, 1]$, we can directly apply Corollary 2.5 from Lueker [25]:

$$\mathbb{P}\left(\forall z \in [-1, 1], \exists I \subset \{1, \cdots, n_1\} \text{ s.t. } |z - \sum_{i\in I} X_i'| \geq \varepsilon\right) \geq 1 - \delta \tag{10}$$

whenever $n_1 \geq C \log(2/\varepsilon\delta)$.

To approximate $\sum_{i=1}^n M_i X_i$, we have to evaluate its norm. By Hoeffding's inequality on X_i's with $M_i \neq 0$, we have

$$\mathbb{P}\Big(\Big|\sum_{i=1}^n M_i X_i\Big| \leq \alpha m\Big) \geq 1 - 2\exp\left(-\frac{3\alpha^2 m}{2}\right), \tag{11}$$

for any fixed $\alpha > 0$, whose value will be specified later. Thus $\left|\sum_{i=1}^n M_i X_i\right| \leq \alpha m \leq \alpha\beta n$ holds with probability at least $1-\delta$ whenever $m \geq \frac{2\log(2/\delta)}{3\alpha^2}$. Since $m \leq \beta n$ holds, $n \geq \frac{2\log(2/\delta)}{3\alpha^2\beta}$ is enough.

From the proof of Corollary 3.1 in Lueker [25], for any $\gamma \in (0, \frac{1}{4})$, we know that

$$\mathbb{P}\left(\forall z \in [-\gamma n_2, \gamma n_2], \exists I \subset \{1, \cdots, n_2\} \text{ s.t. } |z - \sum_{i\in I} X_i''| \geq \varepsilon\right) \tag{12}$$

$$\geq 1 - \delta - 2\exp\left(-\frac{(1-4\gamma)^2 n_2}{64}\right) \tag{13}$$

whenever $n_2 \geq C\log(\frac{2}{\varepsilon\delta})$. Thus if

$$n_2 \geq \max\left(C\log(\frac{2}{\varepsilon\delta}), \frac{C'}{(1-4\gamma)^2}\log(\frac{2}{\delta})\right) \tag{14}$$

holds, we can approximate any $z \in [-\gamma n_2, \gamma n_2]$ by subset sum of $X_1'', \cdots, X_{n_2}''$ with probability $1-2\delta$.

Now we assume $(\alpha\beta/\gamma)n \leq n_2$. For example, if we set $\alpha = \frac{1}{10}$ and $\gamma = \frac{1}{5}$, the assumption is satisfied when we split $\{(1-M_i)X_i\}$ by $n_1 = n_2 = \frac{n}{2}$ in (9). Then the approximation (12) can be applied to $z = \sum_{i=1}^n M_i X_i$ with high probability since $|\sum_{i=1}^n M_i X_i| \leq \alpha m \leq \alpha\beta n \leq \gamma n_2$ holds.

By combining (10), (11), (14) with the fact that $n = m + n_1 + n_2$, and replacing δ with $\delta/4$, we obtain the desired results. □

By similar arguments as in Lueker [25] or Pensia et al. [29], we can easily generalize Lemma B.2 to the case where the distribution followed by X_i contains the uniform distribution. Also, by combining our proof and the proof of Lemma 2.4 in Gadhikar et al. [11], we can prove the following extension of Lemma B.2:

Lemma B.3 (Subset-Sum Approximation in Frozen Networks.)**.** *Let $X_1, ..., X_n$ be independent, uniformly distributed random variables so that $X \sim U(-1,1)$, $M_1, ..., M_n$ be independent, Bernoulli distributed random variables so that $M_i \sim Ber(p)$ for $p \in (0,1)$, and $M_1', ..., M_n'$ be independent, Bernoulli distributed random variables so that $M_i' \sim Ber(q)$ for $q \in (0,1)$. Let $\epsilon, \delta \in (0,1)$ be given. Then for any $z \in [-1,1]$ there exists a subset $I \subseteq \{1, ..., n\}$ so that with probability at least $1-\delta$ we have $\left|z - \sum_{i=1}^n M_i M_i' X_i - \sum_{i\in I} M_i (1 - M_i') X_i\right| \leq \epsilon$ if*

$$n \geq C\log\left(\frac{C'}{\varepsilon\delta}\right), \tag{15}$$

where $C, C' > 0$ is a constant.

Theorem B.4 (SLT Existence in Frozen Networks)**.** *Let $\mathcal{D}$ be input data, f_T be a target network with depth L, and f_S be a source network with depth $L+1$, $\boldsymbol{\theta}_S$ be a parameter of f_S, and $p_l, q_l \in (0,1)$. Assume that these networks have ReLU activation function, and each element of $\boldsymbol{\theta}_S$ is uniformly distributed over $[-1,1]$. Also assume that $\boldsymbol{\theta}_S$ is randomly pruned and locked with pruning ratio p_l and locking ratio q_l for each l-th layer. Then, with probability at least $1-\delta$, there exists a binary mask $\boldsymbol{m}_S$ so that each output component i is approximated as $\max_{\boldsymbol{x}\in\mathcal{D}} \|f_{T,i}(\boldsymbol{x}) - f_{S,i}(\boldsymbol{x}; \boldsymbol{\theta}_S \odot \boldsymbol{m}_S)\| < \epsilon$ if*

$$n_{S,l} \geq C_1 \log\left(\frac{C_2}{\min\{\epsilon_l, \delta/\rho\}}\right) n_{T,l}, \qquad (l \geq 1) \tag{16}$$

$$n_{S,0} \geq C_3 \log\left(\frac{C_4}{\min\{\epsilon_1, \delta/\rho\}}\right) d \qquad (l = 0) \tag{17}$$

where $C_i > 0$ are constants that include p_l, q_l, and ρ, ϵ_l are as defined in Theorem B.1.

Proof. By replacing Lemma 2.4 in Gadhikar et al. [11] with our Lemma B.3, we obtain the desired result as an extension of Theorem 2.5 in Gadhikar et al. [11]. □

C Additional Experimental Results

C.1 Experimental Settings

We evaluate the SLTs within frozen networks on image classification using the CIFAR-10 [20] and ImageNet [32] datasets, and on node classification using the OGBN-Arxiv [17] dataset. CIFAR-10 and ImageNet train data are split into training and validation sets with a 4:1 ratio, while for OGBN-Arxiv we use the default set split. We test the models with the best validation accuracy and report the mean of three experiment repetitions for CIFAR-10 and OGBN-Arxiv, and the result of one experiment for ImageNet. The standard deviation of experiments conducted more than once is plotted as error bars.

For image classification we employ the VGG-like Conv6 [33, 31], ResNet [14], and Wide ResNet [41] architectures, and for graph node classification the 4-layer modified GIN [38] architecture in Huang et al. [18], all implemented with no learned biases. ResNet and GIN use non-affine Batch Normalization [19], while Conv6 has no normalization layers. Random weights are initialized with the Kaiming Uniform distribution, while weight scores are initialized with the Kaiming Normal distribution [13].

SLTs are searched using an extension of `Edge-Popup` [31] that enforces the desired SLT sparsity globally instead of per-layer [34]. On CIFAR-10, scores are optimized for 100 epochs using stochastic gradient descent with momentum 0.9, batch size 128, weight decay 0.0001, and initial learning rates of 0.01 and 0.1 for Conv6 and ResNet-18, respectively. On ImageNet, scores are optimized by the same setting as ResNet-18 on CIFAR-10, but a 256 batch size. On OBGN-Arxiv, scores are optimized for 400 epochs using AdamW [24] with weight decay 0.0001 and initial learning rate 0.01. All experiments use cosine learning rate decay [23]. These can be adequately verified with two NVIDIA H100 SXM5 94GB GPUs.

C.2 Accuracy of SLT within Dense ResNet-50

This section introduces the preliminary experiment used for determining the main experimental setup. Figure 7 compares the accuracy of SLTs within a dense ResNet-50 source at different SLT sparsity using ImageNet. The SLT with 80% sparsity is the most accurate, reaching 66.8% accuracy.

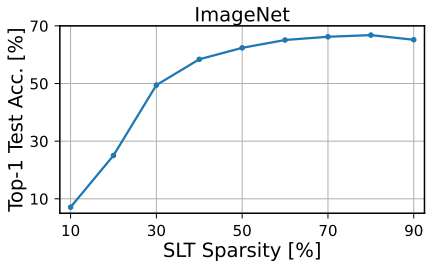


Figure 7: Accuracy comparison of SLTs of different sparsity within a dense ResNet-50 on ImageNet.

C.3 Result Analysis

Table 1 summarizes the presented results and compares the proposed method to weight training and SLT search in dense and sparse source networks for the same network architecture and SLT sparsity.

In the case of Conv6, compared to weight training our method provides reductions of 66.4× of the model size, in exchange of a small accuracy drop. Compared to the SLT found in the sparse source network, the SLT found in the frozen source network achieves much higher accuracy.

Even in the more challenging case of ResNet-18, SLTs found in a frozen network are 250.8× smaller than the trained-weight model. When comparing SLTs found in sparse and frozen networks with the same 85% freezing ratio, the accuracy is almost equivalent, demonstrating that the inclusion of

Table 1: Comparison between trained-weight networks and SLTs found in dense, sparse, and frozen networks.

Method (Source Net.)	Weight Init.	Sparsity [%]	Pruning Ratio [%]	Locking Ratio [%]	Top-1 Test Acc. [%]	Model Size [MB]
CONV6 & CIFAR-10						
Weight Training	KU	-	-	-	87.1	8.63
SLT (Dense)	SKC	50	0	0	86.2	0.27
SLT (Sparse)	SKC	50	45	0	66.7	0.15
SLT (Frozen)	SKC	50	25	25	**84.8**	**0.13**
RESNET-18 & CIFAR-10						
Weight Training	KU	-	-	-	92.4	42.63
SLT (Dense)	SKC	90	0	0	91.0	1.37
SLT (Sparse)	SKC	90	85	0	89.9	0.24
SLT (Frozen)	SKC	90	82.5	2.5	**89.8**	0.24
SLT (Frozen)	SKC	90	85	5	**88.9**	**0.17**
GIN & OGBN-ARXIV						
Weight Training	KU	-	-	-	70.1	1.452
SLT (Dense)	SKC	20	0	0	70.0	0.054
SLT (Sparse)	SKC	20	15	0	62.3	0.047
SLT (Frozen)	SKC	20	0	40	**69.2**	**0.036**
RESNET-50 & IMAGENET						
Weight Training	KU	-	-	-	74.4	97.49
SLT (Dense)	SKC	80	0	0	66.8	3.24
SLT (Sparse)	SKC	80	70	0	47.1	1.11
SLT (Frozen)	SKC	80	65	5	**55.2**	**1.11**
RESNET-34 & IMAGENET						
SLT (Dense)	SKC	80	0	0	62.5	2.66
SLT (Sparse)	SKC	80	70	0	40.7	0.84
SLT (Frozen)	SKC	80	65	5	**51.2**	**0.84**
RESNET-18 & IMAGENET						
SLT (Dense)	SKC	80	0	0	54.0	1.43
SLT (Sparse)	SKC	80	70	0	25.2	0.45
SLT (Frozen)	SKC	80	65	5	**39.4**	**0.45**
WIDE RESNET-50 & IMAGENET						
SLT (Dense)	SKC	80	0	0	70.8	8.46
SLT (Sparse)	SKC	80	70	0	57.7	2.72
SLT (Frozen)	SKC	80	65	5	**63.9**	**2.72**

locking does not introduce a degradation in accuracy even in the scenario that benefits more from pruning.

Interestingly, the SLT found in the frozen GIN model achieves comparable accuracy to the trained-weight network, even though the model size is reduced by $40.3\times$. The SLT found within a dense network also achieves accuracy comparable to that of the trained-weight network, but the SLT found by our method is smaller. Generally, graph neural networks (GNNs) suffer from a generalization performance degradation due to the over-smoothing problem [21]. Searching for SLTs within dense GNNs has been shown to mitigate the over-smoothing problem [18] and achieve higher accuracy. As our method is similarly accurate compared to dense networks, it is possible that parameter freezing offers an even stronger solution to the over-smoothing problem.

Although the ImageNet experiment falls into the scenario where a relatively sparse SLT is more accurate (like the ResNet-18 experiment on CIFAR-10), the SLT found in a sparse network is significantly less accurate. On the other hand, the inclusion of parameter locking shows a significant improvement in SLT accuracy. These results suggest that the number of highly accurate SLT patterns decreases as the difficulty of the problem increases, and the pruning:locking proportion affects the performance of the found SLTs more severely.

D Limitations

Our results have the following limitations: 1) Although the proposed method can effectively reduce the model memory size for inference in principle, actual hardware implementation remains for future work. The previous work of the SLT-specialized neural engine, Hiddenite [15], would be a promising direction for such future implementation. 2) This paper provides theoretical support for the existence of SLTs in frozen networks, but it does not provide their superiority in the approximation capability compared to random pruning, a special case of random freezing, which may depend on the sparsity nature of target networks. Also, the theoretical consideration of the appropriate proportion of these ratios is left for future work. 3) Even though we demonstrate our method on various model architectures including CNNs, ResNet families and GIN, following previous SLT work [42, 31, 22, 18, 39], we have to leave it for future work to apply our method to Transformers [36] because SLT itself has not yet been established for Transformers.

E Impact Statements

This paper presents work whose goal is to advance the field of machine learning. Among the many potential societal consequences of our work, we highlight its potential to reduce the computational cost of inference of neural networks, and thus their energy consumption footprint.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claim of this paper is to search for SLTs from frozen networks, and this claim and the findings from our experiments are described in the abstract and introduction (Section 1) sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the limitations section (Appendix D), we discuss experimental setups and evaluations not conducted in this paper, as well as the limitation of our theoretical contributions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: This paper supports the SLT existence within frozen networks by extending the subset-sum approximation lemma (Lemma A.1), which is a core part of the traditional SLT existence theorem, to the situation where the part of parameters are locked randomly. We provide a proof of this extended lemma in Appendix B, and other lemmas and theorems mentioned abstractly in the main paper are also mentioned specifically in Appendix B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All experimental settings in this paper are described in Appendix C.1. Additional settings specific to the experiment are also described in the corresponding sections (e.g., Section 3.2).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code to reproduce the experimental results cannot be submitted due to inability to attach supplemental materials. We can make them available on github after acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings in this paper are described in Appendix C.1. Additional settings specific to the experiment are also described in the corresponding sections (e.g., Section 3.2).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The plots of the experiments conducted more than once have the error bars using their standard deviation. This explanation is provided in Appendix C.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments in this paper can be verified with two NVIDIA H100 SXM5 94GB GPUs (Appendix C.1).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The impact of this paper is discussed in Appendix E.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: References to the experimental codes are cited in the corresponding paper, and the referenced URL is clearly indicated in the code released later.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Details of our method and experimental procedures are described in this paper (Section 2 and Appendix C.1).

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.