Multicoated Supermasks Enhance Hidden Networks

Yasuyuki Okoshi [*1] **Ángel López García-Arias** [*1] **Kazutoshi Hirose** [1] **Kota Ando** [1] **Kazushi Kawamura** [1]
Thiem Van Chu [1] **Masato Motomura** [1] **Jaehoon Yu** [*1]

Abstract

Hidden Networks (Ramanujan et al., 2020) showed the possibility of finding accurate subnetworks within a randomly weighted neural network by training a connectivity mask, referred to as supermask. We show that the supermask stops improving even though gradients are not zero, thus underutilizing backpropagated information. To address this issue, we propose a method that extends Hidden Networks by training an overlay of multiple hierarchical supermasks—a Multicoated Supermask. This method shows that using multiple supermasks for a single task achieves higher accuracy without additional training cost. Experiments on CIFAR-10 and ImageNet show that Multicoated Supermasks enhance the tradeoff between accuracy and model size. A ResNet-101 using a 7-coated supermask outperforms its Hidden Networks counterpart by 4%, matching the accuracy of a dense ResNet-50 while being an order of magnitude smaller. Code available at: `https://github.com/yasu0001/multicoated-supermasks`

1. Introduction

Since the *Lottery Ticket Hypothesis* (Frankle & Carbin, 2019) showed that overparametrized dense networks contain a subnetwork as effective as the original, multiple studies have followed up on its analysis to deepen the understanding of neural network connectivity pruning. One of the most significant line of follow-up studies found that, not only could these subnetworks be identified in the early stages of the training phase (You et al., 2020), but even before training, at initialization time (Lee et al., 2019; 2020; Wang et al., 2020a;b; Tanaka et al., 2020; Hayou et al., 2021; Frankle et al., 2021; Sreenivasan et al., 2022b).

This line of research took a leap forward when Zhou et al. (2019) discovered that winning tickets can be found just by pruning, without the need of any weight training. In this ablation study, which clarifies the role of each component composing winning tickets, Zhou et al. demonstrate the existence of binary masks—*supermasks*—that, when applied to a randomly initialized network, uncover a subnetwork that achieves high accuracy without the need of ever updating weights. The discovery that weights do not need to be learned, which has come to be known as the *Strong Lottery Ticket Hypothesis*, broke common sense regarding neural network training. *Hidden Networks* (Ramanujan et al., 2020) further refined the idea of supermasks with an algorithm for learning a supermask (`edge-popup`), thus removing the stochasticity from the algorithm proposed by Zhou et al. This approach achieved an inference accuracy comparable to that of a dense network. A following series of papers provided theoretical ground to this phenomenon by analyzing the bounds of necessary overparametrization (Malach et al., 2020; Pensia et al., 2020; Orseau et al., 2020) in addition to showing the robustness of these subnetworks to binarization (Diffenderfer & Kailkhura, 2021; Sreenivasan et al., 2022a).

Supermasks have drawn attention to the intriguing fact that winning tickets are present in randomly initialized neural networks from the beginning. But that is not its only virtue, as they also offer a fresh point of view for the deployment of neural networks. The most practical feature of Hidden Networks is that they can be reconstructed from a small amount of information: unlearned weights are obtained from a random number generator, and the only data that needs to be stored are the random seed and a sparse supermask. These features can be exploited for implementing efficient inference accelerators on specialized hardware (Hirose et al., 2022).

There are compression, quantization, and pruning methods that provide similar advantages, such as pruning connections based on importance (Han et al., 2015), Deep Compression (Han et al., 2016), and coreset-based neural network compression (Dubey et al., 2018), but Hidden Networks are unique in that supermasks handle quantization and pruning

[*]Equal contribution [1]Tokyo Institute of Technology, Japan. Correspondence to: Yasuyuki Okoshi <okoshi.yasuyuki@artic.iir.titech.ac.jp>, Ángel López García-Arias <lopez@artic.iir.titech.ac.jp>, Jaehoon Yu <yu.jaehoon@artic.iir.titech.ac.jp>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

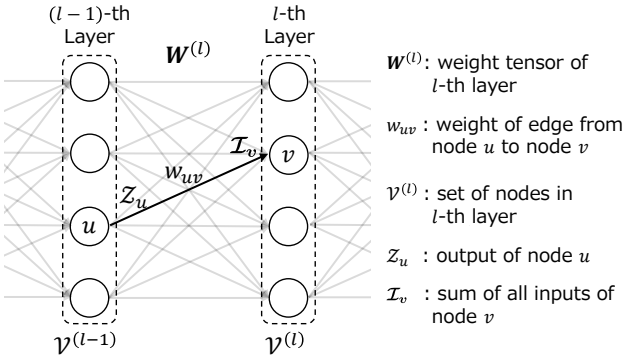


Figure 1. A fully connected layer and its notation.

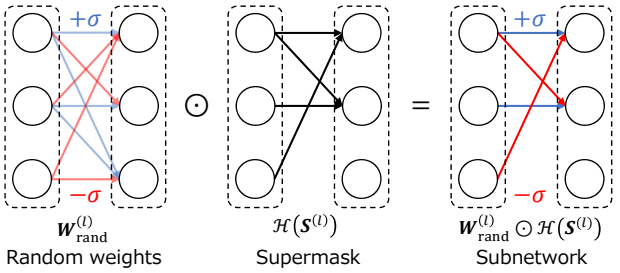


Figure 2. Structure of Hidden Networks with binary weights randomly initialized to $\pm\sigma$, where σ is the standard deviation of Kaiming normal distribution. This initialization method is referred to as Signed Kaiming Constant by Ramanujan et al. (2020).

simultaneously. Furthermore, Hidden Networks make possible to hot swap supermasks for applying the same model to different tasks.

Nonetheless, Hidden Networks has room for improvement in accuracy, task scalability, and model compression. The original work on Hidden Networks provides no solution for compensating the inference accuracy loss other than using wider channels, which require models several times larger for only a slight accuracy improvement. Using iterative pruning with weight reinitialization (Chijiwa et al., 2021), or ternary supermasks for learning both sign and connectivity (Koster et al., 2022), showed some promising results, but they both increase the learning cost and require some form of weight learning. Supermasks in Superposition (Wortsman et al., 2020) succeeded in handling multiple tasks with constant computation cost by using superposed multiple supermasks, and Hidden Fold Networks (López García-Arias et al., 2021) reduced the model size of Hidden Networks by folding them into a recurrent structure.

This paper addresses these issues by proposing *Multicoated Supermasks*, which use multiple supermasks for a single task to form a scalar mask that grants the random weights of strong tickets with virtually the same expressive power as trained weights. Multicoated Supermasks raise the accuracy of the original Hidden Networks without additional computational cost for training, while also offering a better tradeoff between model size and accuracy than wide channel models.

Additionally, we present an analysis of the score evolution during the training of Hidden Networks that reveals an underusage of backpropagated gradients, a finding that forms the underpinning of the proposed method.

The remainder of the paper is organized as follows. Section 2 recapitulates Hidden Networks and analyzes their shortcomings. This serves as the mathematical basis for Section 3, where the proposed Multicoated Supermasks are presented before being explored and evaluated on Section 4. Finally, Section 5 concludes this paper.

2. Recapitulation of Hidden Networks

Hidden Networks is an intriguing approach that integrates pruning and quantization into a single form by learning connectivity instead of weights. Since our proposed method shares the mathematical basis of Hidden Networks, we recap its details in this section .

After reviewing the structure and notation of Hidden Networks in Section 2.1, Section 2.2 summarizes the mathematical proof of Hidden Networks using a fully connected (FC) layer (illustrated in Figure 1), similarly to Ramanujan et al. This proof serves as basis in Section 3 for proving the proposed method in a similar manner. Additionally, in contrast to the original paper, which mainly focused on how swapping edges affects the loss, we also provide an analysis of the score transition after swapping stops, leading to the underlying idea of the proposed method.

2.1. Structure of Hidden Networks

Figure 1 describes the structure of an FC layer and the notation used in this paper. The outputs $\boldsymbol{z}^{(l)} = [\mathcal{Z}^{(l)}_1, \mathcal{Z}^{(l)}_2, \ldots, \mathcal{Z}^{(l)}_{|\mathcal{V}^{(l)}|}]^\top$ of the l-th layer are calculated as

$$\boldsymbol{z}^{(l)} = \mathrm{ReLU}(\boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)}), \tag{1}$$

where $\boldsymbol{W}^{(l)}$ is the weight matrix of layer l. In Hidden Networks, represented in Figure 2, the weight matrix $\boldsymbol{W}^{(l)}$ is obtained from two matrices: a random weight matrix and a supermask matrix, which can be written as

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)}_{\text{rand}} \odot \mathcal{H}(\boldsymbol{S}^{(l)}), \tag{2}$$

where $\odot$ is the Hadamard product operator, and $\boldsymbol{W}^{(l)}_{\text{rand}}$ is a weight matrix randomly initialized by sampling uniformly from $\{-\sigma, \sigma\}$, in which σ is the standard deviation of Kaiming normal distribution (He et al., 2015). $\mathcal{H}$ is the supermask

generating function, defined as

$$\mathcal{H}(\boldsymbol{S}^{(l)}) = \begin{bmatrix} h(s^{(l)}_{1,1}) & h(s^{(l)}_{2,1}) & \cdots & h(s^{(l)}_{U,1}) \\ h(s^{(l)}_{1,2}) & h(s^{(l)}_{2,2}) & \cdots & h(s^{(l)}_{U,2}) \\ \vdots & \vdots & \ddots & \vdots \\ h(s^{(l)}_{1,V}) & h(s^{(l)}_{2,V}) & \cdots & h(s^{(l)}_{UV}) \end{bmatrix}, \quad (3)$$

in which $h(s_{uv})$ is a step function that determines mask values based on the score s_{uv} and a threshold score s_t:

$$h(s_{uv}) = \begin{cases} 1 & |s_{uv}| \geq s_t \\ 0 & \text{otherwise} \end{cases}. \quad (4)$$

Here, s_{uv} is the score assigned to the corresponding weight w_{uv}, and $\boldsymbol{S}^{(l)} \in \mathbb{R}^{U \times V}$, in Eq. (2) and Eq. (3), is the matrix formed by these scores.

The most distinct feature of Hidden Networks is that it uses randomly initialized frozen weights and selects a subset of them based on their scores. This makes the density of the supermask, $k\%$, a pivotal hyperparameter. Given $k\%$ and a tuple of sorted scores, written as

$$\begin{aligned} &\text{sort}(\boldsymbol{S}^{(l)}) = (s'_1, s'_2, s'_3, \cdots, s'_{U \times V}) \quad (5) \\ &\text{s.t.} \quad |s'_1| \geq |s'_2| \geq |s'_3| \geq \cdots \geq |s'_{U \times V}|, \end{aligned}$$

the threshold score is calculated as $s_t = |s'_t|$, where

$$t = \lfloor k\% \times U \times V \rfloor \quad (6)$$

is used to select the top-$k\%$ scores $|s_{uv}| \geq s_t$, i.e., $s'_{\leq t}$.

2.2. `Edge-Popup` and Scores

In the forward pass, `edge-popup` (Ramanujan et al., 2020)—the training algorithm of Hidden Networks—only uses the weights selected by the supermask, while in the backward pass it applies backpropagation on the scores instead of the weights (i.e., weights are never updated). The authors of the original paper provided a simple and clear proof of the efficacy of `edge-popup`, which we briefly review to then expand it to the proposed Multicoated Supermasks in Section 3.

Using the notation in Figure 1, the calculation of a FC layer on Hidden Networks can be written as

$$\mathcal{I}_v = \sum_{u \in \mathcal{V}^{(l-1)}} h(s_{uv}) w_{uv} \mathcal{Z}_u, \quad (7)$$

where $\mathcal{V}^{(l)} = \{v^{(l)}_1, \ldots, v^{(l)}_{n_l}\}$. In the forward pass, $h(s_{uv}) = 1$ if s_{uv} is one of the top-$k\%$ scores; otherwise $h(s_{uv}) = 0$. In the backwards pass, backpropagation uses straight-through estimation (STE) instead.

As mentioned above, Hidden Networks minimizes the loss by updating scores instead of weights. From Eq. (7), the differential of the loss w.r.t. the score is defined as

$$\frac{\partial \mathcal{L}}{\partial s_{uv}} = \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} \frac{\partial \mathcal{I}_v}{\partial s_{uv}} = \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} w_{uv} \mathcal{Z}_u. \quad (8)$$

Notice that the step function h can be omitted by assuming STE. Therefore, to decrease the loss, the score $\tilde{s}_{uv}$ can be updated as

$$\tilde{s}_{uv} = s_{uv} - \lambda \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} w_{uv} \mathcal{Z}_u, \quad (9)$$

where λ is the learning rate, and Hidden Networks takes the absolute value to ensure positive scores.

The remaining question is whether selecting the edges with the top-$k\%$ scores decreases the loss or not. If edge (i, p) replaces edge (j, p) at a gradient update, it means that $s_{ip} < s_{jp}$ before the swap and that $\tilde{s}_{ip} > \tilde{s}_{jp}$ after the update. Then, the following inequality holds:

$$\begin{aligned} &(s_{ip} < s_{jp}) \wedge (\tilde{s}_{ip} > \tilde{s}_{jp}) \\ &\Leftrightarrow \tilde{s}_{ip} - s_{ip} > \tilde{s}_{jp} - s_{jp} \\ &\Leftrightarrow -\lambda \tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{ip} \mathcal{Z}_i > -\lambda \tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{jp} \mathcal{Z}_j \quad \because \text{ from (9)} \\ &\Leftrightarrow \tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} (w_{ip} \mathcal{Z}_i - w_{jp} \mathcal{Z}_j) < 0. \quad (10) \end{aligned}$$

As with all gradient-based optimization methods, the loss $\mathcal{L}(\tilde{\mathcal{I}}_p)$ can be approximated with its Taylor expansion, as shown in Figure 3. When $\tilde{\mathcal{I}}_p$ is within the open neighborhood of $\mathcal{I}_p$, $\mathcal{L}(\tilde{\mathcal{I}}_p)$ can be approximated as

$$\begin{aligned} \mathcal{L}(\tilde{\mathcal{I}}_p) &= \mathcal{L}(\mathcal{I}_p + (\tilde{\mathcal{I}}_p - \mathcal{I}_p)) \\ &\approx \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p} (\tilde{\mathcal{I}}_p - \mathcal{I}_p) \\ &= \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p} (w_{ip} \mathcal{Z}_i - w_{jp} \mathcal{Z}_j). \quad (11) \end{aligned}$$

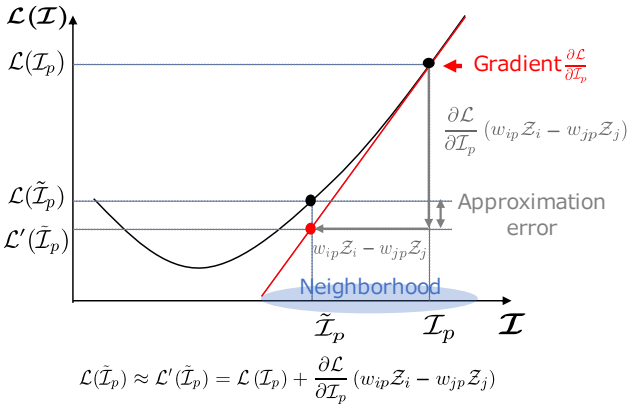


Figure 3. First-order Taylor expansion of the loss on Hidden Networks. To keep the approximation error small, $\tilde{\mathcal{I}}_p$ needs to be within the neighborhood of $\mathcal{I}_p$.

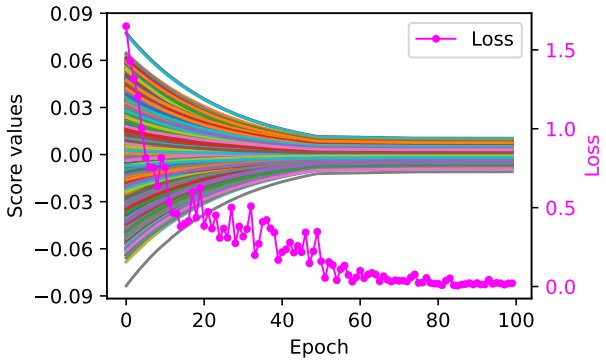


Figure 4. Evolution of loss and scores with weight decay during training. Solid lines correspond to 10 000 scores chosen randomly from the last convolutional layer of ResNet-18.

From Eq. (10), it is obvious that $\mathcal{L}(\tilde{\mathcal{I}}_p) < \mathcal{L}(\mathcal{I}_p)$, proving that `edge-popup` decreases the loss by swapping edges.

However, what would happen to scores if there was no swapping? If swapping stops, the score gradients $\frac{\partial \mathcal{L}}{\partial s_{uv}}$ are constant, and therefore scores will constantly increase or decrease until swapping occurs, as no swapping means no change in the subnetwork nor in the loss. In practice, however, this does not happen due to the weight decay applied to scores. Figure 4 shows that most scores reach a plateau after 50 epochs, after which they do not constantly increase or decrease. But this does not mean that their gradients are zero: it is the effect of the balance between the score increase and the weight decay, i.e., their speeds of increment and decay are almost equal. Although the continuously decreasing loss reveals that the subnetwork still undergoes slight changes, these underutilized gradients can be exploited for further profit. The underlying idea of the proposed method is to take advantage of these non-zero score gradients for enhancing the performance of Hidden Networks.

3. Multicoated Supermasks

This section proposes a method that extends Hidden Networks to use multiple supermasks—bundled in a Multicoated Supermask—instead of a single supermask. Although using multiple supermasks for multiple tasks (one mask for each task) has been proposed before by Wortsman et al. (2020), this is the first paper, to the best of our knowledge, discussing multiple supermasks for a single task.

The proposed method, illustrated in Figure 5, employs N supermasks—N coats—of descending density. That is, additional coats select a subset of the connections selected by the previous coat. Because of this information redundancy, and since all supermask coats use identical scores, this method has no additional training cost compared with Hidden Networks. Additionally, this redundancy is exploited to compress the Multicoated Supermasks.

3.1. Formulation

With the proposed Multicoated Supermasks, Eq. (2) can be rewritten as

$$\boldsymbol{W}^{(l)} = \sum_{k_n \in \mathcal{K}} \mathcal{H}(\boldsymbol{S}^{(l)}, k_n) \odot \boldsymbol{W}^{(l)}_{\text{rand}}, \tag{12}$$

where $\mathcal{K}$ is a set of N supermask densities. Denoting the element of matrix $\boldsymbol{A}$ at (x, y) as $(\boldsymbol{A})_{xy}$, the coordinates of non-zero elements in a supermask coat can be written as

$$\mathbb{H}^{(l)}_k = \left\{ (u, v) \middle| k > 0,\ \left(\mathcal{H}(\boldsymbol{S}^{(l)}, k) \right)_{uv} \neq 0 \right\}. \tag{13}$$

Then, the following statement holds between $\mathcal{H}_k$ with different density k_n:

$$k_i < k_j \Rightarrow \mathbb{H}^{(l)}_{k_i} \subset \mathbb{H}^{(l)}_{k_j}, \tag{14}$$

meaning that the non-zero elements of a supermask coat with smaller k_n are a subset of those in a coat with larger k_n, and so is the corresponding set of coordinates (i.e., this method "applies several coats" of the learned supermask). These non-zero indices of each supermask coat are all the information necessary for constructing the desired subnetwork, and the redundancy of the subsets can be exploited for compressing them.

For calculating the output of a FC layer with a Multicoated Supermask, Eq. (7) is extended to

$$\mathcal{I}_v = \sum_{u \in \mathcal{V}^{(l-1)}} w_{uv} \mathcal{Z}_u \sum_{n=1}^{N} h_{k_n}(s_{uv}) \quad \text{s.t.} \quad k_1 > k_2 > \cdots > k_N > 0, \tag{15}$$

where N is the number of total coats, and k_n is the density of the n-th coat. Notice that when $N = 1$, Eq. (15) is identical to Eq. (7), allowing to consider Hidden Networks as a special case of the proposed method.

The score update rule in Eq. (9) is rewritten just by introducing N as

$$\tilde{s}_{uv} = s_{uv} - \lambda \frac{\partial \mathcal{L}}{\partial \mathcal{I}_v} N w_{uv} \mathcal{Z}_u, \tag{16}$$

assuming STE over the summation of step functions in Eq. (15). Therefore, Multicoated Supermasks can be proved in the same manner with Hidden Networks based on Eq. (15) and Eq. (16). We divide the proof in two possible scenarios: the case of adding an additional coat, and the case of a swap between two existing coats.

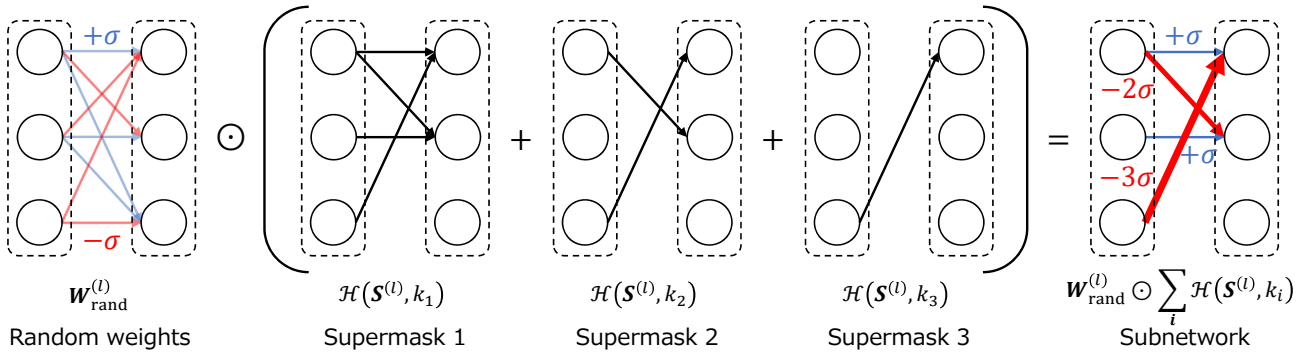


Figure 5. An example of Multicoated Supermask with three coats and Signed Kaiming Constant weights. Each coat consists of a supermask with a different density top-k%: in this figure $k_1 > k_2 > k_3$. Since all coats use the same score matrix $\boldsymbol{S}^{(l)}$, all edges in the k_3 coat are included in the k_2 coat, which in turn is a subset of the k_1 coat.

In the case of adding a new coat, the loss $\mathcal{L}(\tilde{\mathcal{I}}_p)$ is approximated as

$$\begin{aligned}\mathcal{L}(\tilde{\mathcal{I}}_p) &\approx \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p}(\tilde{\mathcal{I}}_p - \mathcal{I}_p) \\ &= \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p}\{c w_{ip}\mathcal{Z}_i - (c-1)w_{ip}\mathcal{Z}_i\} \\ &= \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{ip}\mathcal{Z}_i, \end{aligned} \tag{17}$$

where $c \in \{0, 1, \ldots, N\}$ represents the number of coats accumulated in Eq. (15), and the term $\frac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{ip}\mathcal{Z}_i$ is less than 0 because

$$\begin{aligned} &\tilde{s}_{ip} - s_{ip} > 0 \\ \Leftrightarrow\ &-\lambda \tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} N w_{ip}\mathcal{Z}_i > 0 \quad \because \text{ from (16)} \\ \Leftrightarrow\ &\tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{ip}\mathcal{Z}_i < 0. \end{aligned} \tag{18}$$

Consequently, the updated loss $\mathcal{L}(\tilde{\mathcal{I}}_p)$ becomes smaller than the previous loss $\mathcal{L}(\mathcal{I}_p)$ by adding an additional coat.

In the case of an edge swap happening between two existing coats, the loss $\mathcal{L}(\tilde{\mathcal{I}}_p)$ can be written as

$$\begin{aligned}\mathcal{L}(\tilde{\mathcal{I}}_p) &\approx \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p}(\tilde{\mathcal{I}}_p - \mathcal{I}_p) \\ &= \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p}\{c w_{ip}\mathcal{Z}_i - (c-1)w_{ip}\mathcal{Z}_i\} \\ &\quad + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p}\{(c-1)w_{jp}\mathcal{Z}_j - c w_{jp}\mathcal{Z}_j\} \\ &= \mathcal{L}(\mathcal{I}_p) + \frac{\partial \mathcal{L}}{\partial \mathcal{I}_p}\{w_{ip}\mathcal{Z}_i - w_{jp}\mathcal{Z}_j\}, \end{aligned} \tag{19}$$

which is identical to Eq. (11). As with Eq. (10), the updated loss $\mathcal{L}(\tilde{\mathcal{I}}_p)$ is also smaller than the loss before the update $\mathcal{L}(\mathcal{I}_p)$, as

$$\begin{aligned} &\tilde{s}_{ip} - s_{ip} > \tilde{s}_{jp} - s_{jp} \\ \Leftrightarrow\ &-\lambda N \tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{ip}\mathcal{Z}_i > -\lambda N \tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p} w_{jp}\mathcal{Z}_j \\ \Leftrightarrow\ &\tfrac{\partial \mathcal{L}}{\partial \mathcal{I}_p}(w_{ip}\mathcal{Z}_i - w_{jp}\mathcal{Z}_j) < 0. \end{aligned} \tag{20}$$

Therefore, in both the case of adding additional coats and the case of a swap between coats, the Multicoated Supermask approach works.

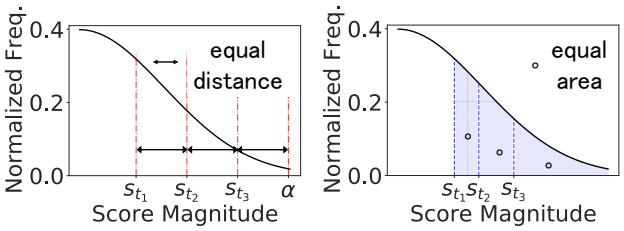
So far we have discussed Multicoated Supermasks for fully connected neural networks. Due to the similarity with Hidden Networks, Multicoated Supermasks also extends to convolutional neural networks in almost the same way as Hidden Networks (Ramanujan et al., 2020, Section B.2), except for the additional N from the STE of the step functions used in Multicoated Supermasks.

3.2. Narrowing Down the Hyperparameter Space

Although Multicoated Supermasks can be trained with `edge-popup` as well, it is necessary to determine a density k_n for each of the N coats. To avoid the additional cost of exploring a bigger hyperparameter space, we propose two simple strategies for determining k_n from the total density of the model top-k% (i.e., the density of the first coat, k_1): Linear and Uniform.

These two methods, portrayed in Figure 6, assume that the magnitude of learned scores follows a folded normal distribution (although, in reality, the concentration of scores around zero is slightly more prominent), and each uses a different way of partitioning this probability density function to obtain the necessary k_n. These two methods are compared experimentally in Section 4.2.

Linear method (Figure 6a) partitions the score magnitude segment $[s_{t_1}, \alpha]$ at even score magnitude intervals, where s_{t_1} is the threshold score with density k_1, and we define



(a) Linear (b) Uniform

Figure 6. Two methods for setting the density k_n of each coat n: (a) uses equidistant threshold scores s_{t_n}; (b) considers area under the curve of a folded normal distribution. $\alpha = s_{t_1} + 3\sigma_s$, where σ_s is the scores' standard deviation.

$\alpha = s_{t_1} + 3\sigma_s$ as upper bound for partitioning, in which σ_s is the standard deviation of the score distribution. The range of $3\sigma_s$ from s_{t_1} covers most of the scores range regardless of s_{t_1}. Then, the threshold score s_{t_n} for each density k_n is calculated by

$$s_{t_n} = s_{t_1} + (\alpha - s_{t_1})\frac{n}{N}, \tag{21}$$

and then, using Eq. (5) and Eq. (6),

$$k_n = t_n/UV. \tag{22}$$

Uniform (Figure 6b), on the other hand, uses even portions of area under the curve to partition the score magnitude segment $[s_{t_1}, \infty)$. Thus, k_n is simply defined as

$$k_n = k_1\frac{n}{N}. \tag{23}$$

3.3. Multicoated Supermask Encoding

Since weights are random, it is only necessary to store the supermasks and the random seed used at training for generating weights. Supermask coats with higher density contain those with lower density (see Eq. (14)), for an additional coat n it only necessary to store the elements located at the coordinates of non-zero elements of the previous coat $n-1$, i.e., $\mathbb{H}^{(l)}_{k_n-1}$. Therefore, the total number of bits of a unary encoded Multicoated Supermask is

$$\begin{cases} UV, & \text{for } N = 1 \\ \left(1 + \sum_{n=2}^{N} k_{n-1}\right) UV, & \text{for } N > 1 \end{cases}. \tag{24}$$

The model sizes reported in Section 4 assume this encoding.

3.4. Single- and Multicoated Supermasks Comparison

As mentioned above, Hidden Networks can be considered a special case of Multicoated Supermasks that only use the first coat. Since additional coats are subsets of this first coat, they do not modify network connectivity. However, the scaling they provide expands the parameter search space, leading the learning algorithm to better quality models by exploiting the backpropagated information more effectively. As will be discussed in Section 4, Multicoated Supermasks achieve this by partially restoring the relationship between a connection's importance and its magnitude in the activation's linear combination—which Hidden Networks broke into score and random weight, respectively.

Since all coats are computed from the first one, multicoating has a trivial impact on training cost. Compared with the binary supermasks of Hidden Networks, encoding all coats into a single Multicoated Supermask makes them scalar supermasks. In specialized hardware, this may entail a negligible added cost in its application to the weights, but none in standard processors. The only significant inconvenience introduced by this method is the additional memory needed for storing more coats. However, as demonstrated in Section 4, increasing supermask size is more efficient than increasing model size, whether it is with extra layers or wider channels, as it results in models with similar memory size and less computational cost but higher accuracy.

Previous work found success in reusing a single random network for multiple tasks by training a supermask for each task and hot swapping them accordingly (Wortsman et al., 2020). Multicoated Supermasks differs in that it uses multiple supermasks for a single task. However, these two approaches are compatible: one Multicoated Supermask can be trained for each mask, raising the accuracy for each task while retaining the capacity of hot-swapping supermasks.

4. Evaluation

This subsection evaluates the performance of Multicoated Supermasks on image classification and compares it to Hidden Networks using the experimental settings described in Section 4.1. First, a small-scale dataset is used to investigate the behaviour of the proposed method: Section 4.2 compares the different methods explained in Section 3 for setting the density of each supermask coat, while Section 4.3 explores the relationship between total model density and accuracy. Then, a large-scale dataset is used to evaluate the performance of the proposed method and its size to accuracy tradeoff by testing different supermask sizes in Section 4.4 and different model sizes in Section 4.5.

4.1. Experimental Settings

We evaluate Multicoated Supermasks for image classification using the CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015) datasets. In both cases residual networks (He et al., 2016) are trained for 100 epochs using stochastic gradient descent (SGD) with

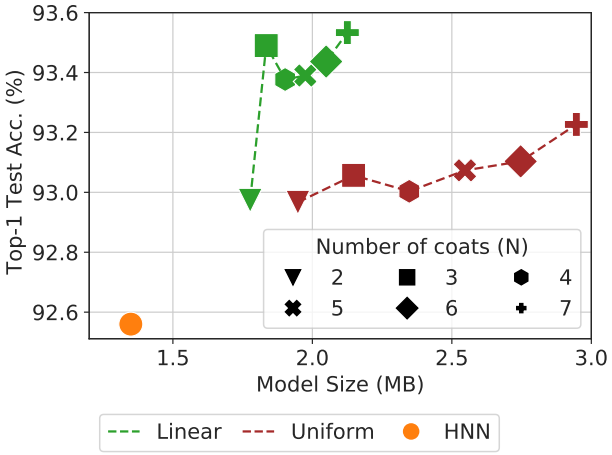


Figure 7. Comparison between the proposed methods for setting k_n, using ResNet-18 on CIFAR-10 with and a fixed $k_1 = 30\%$. Linear produces smaller and more accurate models.

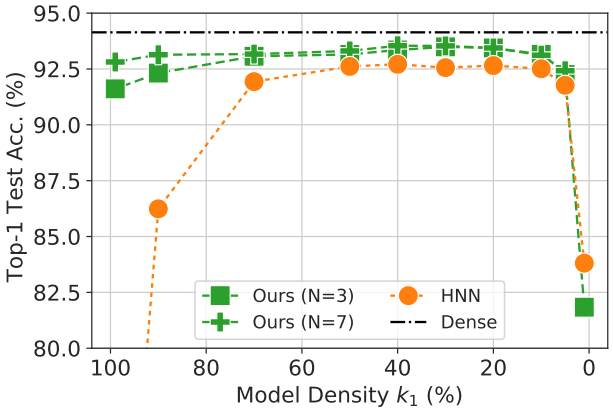


Figure 8. Varying the value of density $k_1\%$ while setting the rest of k_n with the Linear method. ResNet-18 on CIFAR-10 with N-coated supermasks.

weight decay of 0.0001 and momentum of 0.9. Wide ResNets (Zagoruyko & Komodakis, 2016) are used for wider channel models. In CIFAR-10 experiments, the learning rate is decreased by 0.1 after 50 and 75 epochs starting from 0.1 with a batch size of 128; in ImageNet experiments, the learning rate is reduced using cosine annealing starting from 0.1, with a batch size of 256. Following Ramanujan et al. (2020), we use Signed Kaiming Constant initialization (see Section 2.1) with a scaling factor of $\sqrt{1/k_1}$, depending on the sparsity k_1. However, instead of non-affine batchnorm, our experiments use affine batch normalization (i.e., batchnorm learnable parameters are updated). All models and experiments are implemented using MMClassification (MMClassification Contributors, 2020), a toolbox based on PyTorch (Paszke et al., 2019). All reported model sizes for models using supermasks use the encoding described in Section 3. Reported accuracy is the average of three runs for CIFAR-10 experiments, and a single run for ImageNet experiments.

4.2. Setting Each Coat's Density

Figure 7 compares the two methods for setting k_n proposed in Section 3.2 (Linear and Uniform) by testing them on CIFAR-10 using ResNet-18 and a fixed $k_1 = 30\%$. This figure shows that the proposed method is effective: although only slightly, both methods surpass the accuracy of the baseline Hidden Networks, and accuracy grows with the number of coats. Notably, the biggest gap in accuracy is observed between the single-coated and the double-coated supermasks.

Even though the difference in accuracy between the two methods is narrow, Linear consistently outperforms Uniform. Furthermore, since Linear prefers larger additional threshold scores, it produces sparser coats, and thus significantly smaller models. Since Linear produces both the smallest and most accurate models, it is the method used hereafter.

This result suggests that the additional expressive power of Multicoated Supermasks comes from partially restoring the linear relationship between connectivity strength—scores—and weight magnitude of conventional weight learning. For all practical purposes, when applied to models initialized with Signed Kaiming Constant (see Section 2.1), a Multicoated Supermask extends the dynamic range of effective weights without weight learning, functioning as a concurrent blend of pruning and quantization.

4.3. Effect of Total Density on Accuracy

Figure 8 shows the results of investigating the relationship between the total supermask density top-$k\%$ (i.e., the density of the first coat, k_1) and the accuracy of the resulting subnetwork. It shows that Multicoated Supermasks achieve higher accuracy than the single-coated Hidden Networks for practically any density value. The best results are obtained in the range $40 \leq k_1 \leq 20$, with a maximum at $k_1 = 40\%$ of 93.54%, close to the dense model's 94.14%.

It can also be appreciated that, although both single-coated and multicoated supermasks degenerate critically at high sparsity values, additional coats prevent accuracy degradation on dense supermasks ($k_1 > 70\%$). This result proves the hypothesis that Multicoated Supermasks help to mitigate the impact of using random weights.

4.4. Supermask Size to Accuracy Tradeoff

This section evaluates how supermask size affects accuracy and model size using ResNet-50 on ImageNet. The size of

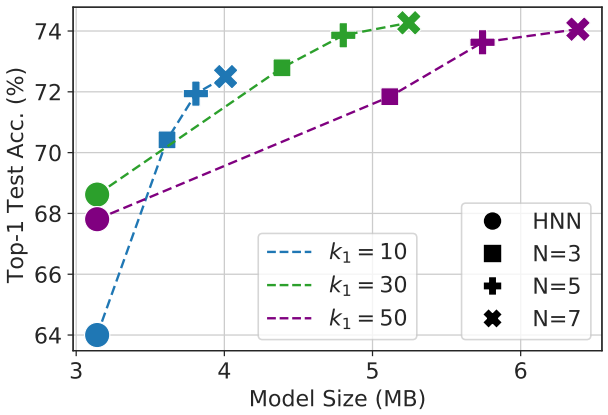


Figure 9. Effect of supermask size (determined by first coat density $k_1\%$ and number of coats N) on accuracy and model size using ResNet-50 and ImageNet. HNN: Hidden Networks (i.e., $N = 1$).

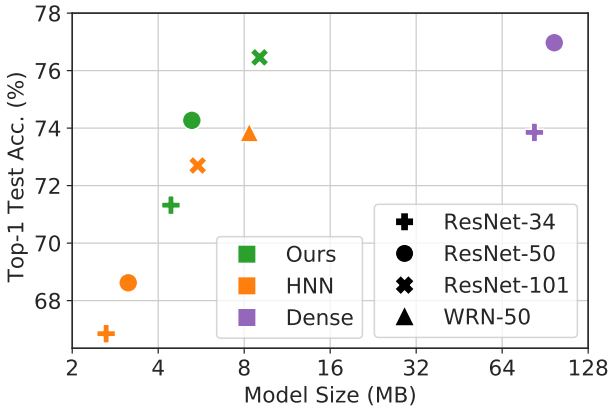


Figure 10. Comparison of Multicoated Supermasks ($N = 7$, $k_1 = 30\%$, Linear), Hidden Networks ($k_1 = 30\%$) and dense ResNet models on ImageNet. WRN: Wide ResNet. The proposed method outperforms Hidden Networks, and matches the dense model, despite the $10\times$ model size reduction.

Multicoated Supermasks depends on two hyperparameters: the number of coats N and the total density k_1. This experiment probes the best ranges of N and k_1 found in the previous subsections.

Figure 9 shows that not all ways of increasing supermask size are equal: while accuracy grows monotonically with number of coats, there is an optimal density value around $k_1 = 30\%$. Blending N and k_1 offers an almost linear tradeoff between accuracy and model size. The highest accuracy, obtained with $k_1 = 30$ and $N = 7$, overperforms Hidden Networks by 5.65%, while only requiring a $1.67\times$ bigger model size.

4.5. Model Size to Accuracy Tradeoff

Figure 10 evaluates the performance of Multicoated Supermasks ($N = 7$, $k_1 = 30\%$, Linear) when considering different model sizes, and compares it to the the baseline Hidden Networks ($k_1 = 30\%$) and dense models.

The enhanced expressive power granted by the additional supermask coats delivers an outstanding tradeoff between accuracy and model size. A multicoated ResNet-101 achieves higher accuracy (76.46%) than the best performing single-coated model, Wide ResNet-50 (73.85%), despite having similar model size (9.0 MB vs. 8.3 MB). Even more remarkably, this model's accuracy is only 0.5% lower than that of a dense ResNet-50, while having a model size $10.78\times$ smaller. These results prove that Multicoated Supermasks achieve higher accuracy for the same model size and smaller size for the same accuracy even on a large scale dataset, with no added training cost and negligible additional inference cost.

4.6. Effect of Weight Initialization

Figure 11 shows the impact on accuracy of using standard Kaiming initialization (KN, (He et al., 2015)) instead of Signed Kaiming Constant initialization (SKC, see Section 2.1). Similarly to Ramanujan et al. (2020), we observe that SKC achieves a higher accuracy than KN. However, we observe that, as the search space expands with the additional coats, this difference disappears.

4.7. Multicoated Supermasks vs. Multiple Supermasks

To demonstrate the effectiveness of using a Multicoated Supermask instead of multiple independent supermasks, we experiment with using an independent score tensor for each mask instead of shared scores. Figure 12 shows that, independently of the number of masks, shared scores result in higher accuracy than independent scores. Since the multiple coats have the combined effect of scaling each weight depending on its importance (i.e., its score), having multiple scores for each weight introduces redundant scores in the best case, and orthogonal scores in the worst case, which eliminate the scaling effect and produce denser subnets.

4.8. Comparison with Other Approaches

Table 1 compares the inference accuracy, inference FLOPS, and model size of the proposed method with pruning (Gale et al., 2019), and with two sparsity training methods: RigL (Evci et al., 2020) and MEST+EM&S (Yuan et al., 2021). Model sizes of the compared literature are calculated by considering nonzero elements as 32-bit (Zhu & Gupta, 2017). Compared with approaches with the same sparsity and computation, Multicoated Supermasks result in models

Table 1. Comparison on ImageNet of the proposed method with other sparsity training methods.

Method	Model	Sparsity	Inference GFLOPs	Model Size (MB)	Top-1 Acc. (%)
Dense	ResNet-50	-	8.2	98	77.1
Pruning (Gale et al., 2019)	ResNet-50	90	0.9	20	75.2
RigL (Evci et al., 2020)	ResNet-50	90	0.9	20	75.7
MEST+EM&S (Yuan et al., 2021)	ResNet-50	90	0.9	20	76.1
Ours (N=7)	ResNet-50	90	0.9	4	72.5
Ours (N=7)	ResNet-50	70	2.5	5	74.3
Ours (N=7)	ResNet-101	70	4.8	9	76.5

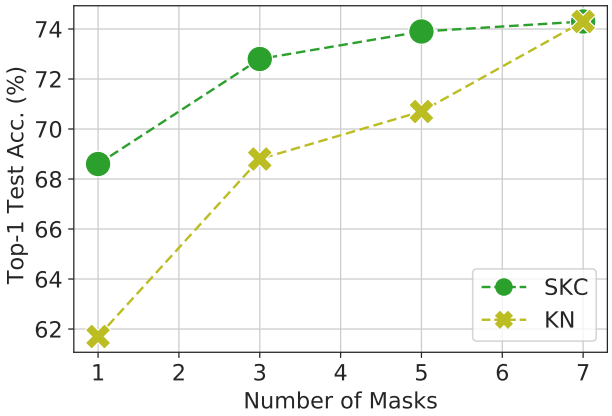


Figure 11. Impact of weight initialization on ImageNet using ResNet-50 ($k_1 = 30\%$, linear). SKC is signed Kaiming constant initialization, and KN is Kaiming normal initialization.

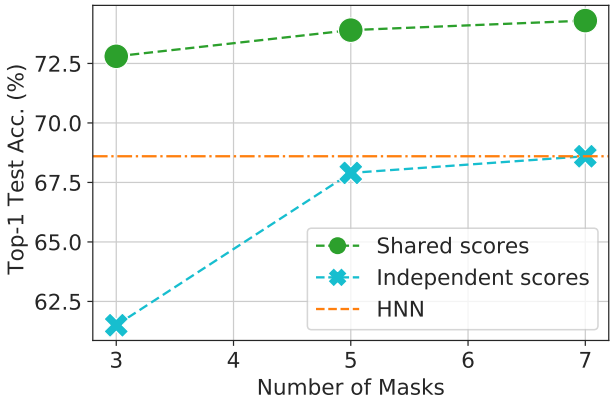


Figure 12. Impact of learning masks with independent scores instead of a shared score. ResNet-50 ($k_1 = 30\%$, Linear, SKC) on ImageNet.

$5\times$ smaller, although slightly less accurate. By increasing the number of layers and density, the proposed method outperforms other works while still keeping the model size more than 50% smaller. These results show that Multicoated Supermasks are an efficient option for memory-constrained applications.

5. Conclusion

This work analyzes Hidden Networks (Ramanujan et al., 2020), showing that their limit in performance is due to supermask optimization stopping before score gradients converge to the minima. This finding that `edge-popup` is not uncovering the best possible connectivity pattern suggests that randomly weighted neural networks may be hiding even better performing subnetworks than previously thought, aligning with the conclusions of Fischer & Burkholz (2021).

We address this problem by enhancing the random subnetwork's expressive power with Multicoated Supermasks, which use multiple supermasks to scale random weights. This method delivers a better accuracy to model size trade-off, showing that increasing the number of supermasks can be more effective than adjusting sparsity or channel width for compensating the accuracy loss of models with random weights.

With a broader view, the high performance and small size of supermasked models point to a promising trend of lightweight neural networks combining pruning, quantization, and random weights.

Acknowledgement

This work was supported by JST CREST Grant Number JPMJCR18K2 and JSPS KAKENHI Grant Numbers JP18H05288, JP22H03555, Japan.

References

Chijiwa, D., Yamaguchi, S., Ida, Y., Umakoshi, K., and Inoue, T. Pruning randomly initialized neural networks with iterative randomization. In *Proc. Adv. Neural Inform. Process. Syst.*, 2021.

Diffenderfer, J. and Kailkhura, B. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by

pruning a randomly weighted network. In *Proc. Int. Conf. Learn. Repr.*, 2021.

Dubey, A., Chatterjee, M., and Ahuja, N. Coreset-based neural network compression. In *Proc. European Conf. Comput. Vis.*, pp. 454–470, 2018.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *Proc. Int. Conf. Mach. Learn.*, pp. 2943–2952, 2020.

Fischer, J. and Burkholz, R. Plant'n'seek: Can you find the winning ticket? *arXiv preprint arXiv:2111.11153*, 2021.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proc. Int. Conf. Learn. Repr.*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Pruning neural networks at initialization: Why are we missing the mark? In *Proc. Int. Conf. Learn. Repr.*, 2021.

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

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Proc. Adv. Neural Inform. Process. Syst.*, pp. 1135–1143, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *Proc. Int. Conf. Learn. Repr.*, 2016.

Hayou, S., Ton, J.-F., Doucet, A., and Teh, Y. W. Robust pruning at initialization. In *Proc. Int. Conf. Learn. Repr.*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, pp. 770–778, 2016.

Hirose, K., Yu, J., Ando, K., Okoshi, Y., López García-Arias, Á., Suzuki, J., Chu, T. V., Kawamura, K., and Motomura, M. Hiddenite: 4K-PE Hidden Network Inference 4D-Tensor Engine Exploiting On-Chip Model Construction Achieving 34.8-to-16.0TOPS/W for CIFAR-100 and ImageNet. In *Proc. IEEE Int. Solid-State Circuits Conf.*, volume 65, pp. 1–3, 2022.

Koster, N., Grothe, O., and Rettinger, A. Signing the supermask: Keep, hide, invert. In *Proc. Int. Conf. Learn. Repr.*, 2022.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Lee, N., Ajanthan, T., and Torr, P. Snip: Single-shot network pruning based on connection sensitivity. In *Proc. Int. Conf. Learn. Repr.*, 2019.

Lee, N., Ajanthan, T., Gould, S., and Torr, P. H. A signal propagation perspective for pruning neural networks at initialization. In *Proc. Int. Conf. Learn. Repr.*, 2020.

López García-Arias, Á., Hashimoto, M., Motomura, M., and Yu, J. Hidden-fold networks: Random recurrent residuals using sparse supermasks. In *Proc. Brit. Mach. Vis. Conf.*, 2021.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *Proc. Int. Conf. Mach. Learn.*, pp. 6682–6691, 2020.

MMClassification Contributors. Openmmlab's image classification toolbox and benchmark. https://github.com/open-mmlab/mmclassification, 2020.

Orseau, L., Hutter, M., and Rivasplata, O. Logarithmic pruning is all you need. In *Proc. Adv. Neural Inform. Process. Syst.*, pp. 2925–2934, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Proc. Adv. Neural Inform. Process. Syst.*, pp. 8024–8035, 2019.

Pensia, A., Rajput, S., Nagle, A., Vishwakarma, H., and Papailiopoulos, D. S. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. In *Proc. Adv. Neural Inform. Process. Syst.*, 2020.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, pp. 11893–11902, 2020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.*, 2015.

Sreenivasan, K., Rajput, S., Sohn, J.-Y., and Papailiopoulos, D. Finding nearly everything within random binary networks. In *Proc. Int. Conf. on Artif. Intell. and Stat.*, pp. 3531–3541, 2022a.

Sreenivasan, K., Sohn, J.-y., Yang, L., Grinde, M., Nagle, A., Wang, H., Lee, K., and Papailiopoulos, D. Rare gems: Finding lottery tickets at initialization. In *arXiv preprint arXiv:2202.12002*, 2022b.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Proc. Adv. Neural Inform. Process. Syst.*, volume 33, pp. 6377–6389, 2020.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. In *Proc. Int. Conf. Learn. Repr.*, 2020a.

Wang, Y., Zhang, X., Xie, L., Zhou, J., Su, H., Zhang, B., and Hu, X. Pruning from scratch. In *Proc. AAAI Conf. on Artif. Intell.*, volume 34, pp. 12273–12280, 2020b.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. *Proc. Adv. Neural Inform. Process. Syst.*, 33, 2020.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Toward more efficient training of deep networks. In *Proc. Int. Conf. Learn. Repr.*, 2020.

Yuan, G., Ma, X., Niu, W., Li, Z., Kong, Z., Liu, N., Gong, Y., Zhan, Z., He, C., Jin, Q., Wang, S., Qin, M., Ren, B., Wang, Y., Liu, S., and Lin, X. MEST: Accurate and fast memory-economic sparse training framework on the edge. In *Proc. Adv. Neural Inform. Process. Syst.*, 2021.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *Proc. Brit. Mach. Vis. Conf.*, 2016.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Proc. Adv. Neural Inform. Process. Syst.*, pp. 3597–3607, 2019.

Zhu, M. and Gupta, S. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.