HyperSparse Neural Networks: Shifting Exploration to Exploitation through Adaptive Regularization

Patrick Glandorf*, Timo Kaiser*, Bodo Rosenhahn
Institute for Information Processing (tnt)
L3S - Leibniz Universität Hannover, Germany
{glandorf, kaiser, rosenhahn}@tnt.uni-hannover.de

Abstract

Sparse neural networks are a key factor in developing resource-efficient machine learning applications. We propose the novel and powerful sparse learning method Adaptive Regularized Training (ART) to compress dense into sparse networks. Instead of the commonly used binary mask during training to reduce the number of model weights, we inherently shrink weights close to zero in an iterative manner with increasing weight regularization. Our method compresses the pre-trained model "knowledge" into the weights of highest magnitude. Therefore, we introduce a novel regularization loss named HyperSparse that exploits the highest weights while conserving the ability of weight exploration. Extensive experiments on CIFAR and TinyImageNet show that our method leads to notable performance gains compared to other sparsification methods, especially in extremely high sparsity regimes up to 99.8% *model sparsity. Additional investigations provide new insights into the patterns that are encoded in weights with high magnitudes.*[1]

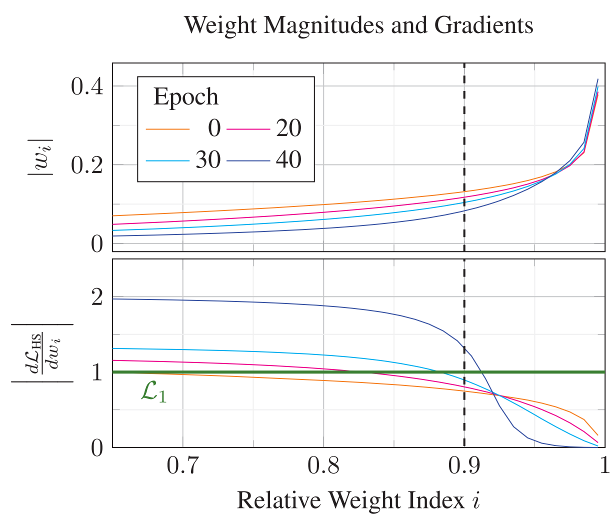


Figure 1: Magnitude of weights with their corresponding gradients at different epochs derived from our *HyperSparse* loss sorted by the weight magnitude. The weights and gradients belong to a ResNet-32 trained on Cifar-100, where the desired pruning rate is $\kappa = 90\%$. The smallest weight w_κ that remains after pruning is marked by a dashed line. Note that we added the gradient for the $\mathcal{L}_1$ loss in green.

1. Introduction

Recent years have shown tremendous progress in the field of machine learning based on the use of neural networks (NN). Alongside the increasing accuracy in nearly all tasks, also the computational complexity of NNs increased, *e.g.*, for Transformers [4, 6] or Large Language Models [1]. The complexity causes high energy costs, limits the applicability for cost efficient systems [9], and is counterproductive for the sake of fairness and trustworthiness due to dwindling interpretability [37].

Facing these issues, recent years have also led to a growing community in the field of sparse NNs [11]. The goal is to find small subgraphs (*a.k.a* sparse NNs) in well performing NNs that have similar or comparable capabilities regarding the main tasks while being significantly less complex and therefore cheaper and potentially better interpretable. Standard methods usually create sparse NNs by obtaining a binary mask that limits the number of used weights in a NN [19, 33, 41]. The most prominent method is *Iterative Magnitude Pruning* (*IMP*) [15] that is based on the *Lottery Ticket Hypothesis* (*LTH*) [8]. Assuming that important weights have high magnitudes after training, it trains a dense NN and removes an amount of elements from the mask that correspond to the lowest weights. Afterward, the sparse NN is reinitialized and retrained from scratch. The process is iterated until a sparsity level is reached.

The assumption of magnitude pruning that highest weights in dense NNs encode most important decision rules for a diverse set of classes is problematic, because it is not guaranteed. Removed weights that are potentially useful

*These authors contributed equally to this work
[1]Code available at https://github.com/GreenAutoML4FAS/HyperSparse

to the prediction can no longer be reactivated during fine-tuning. In the worst case, a "layer collapse" can prohibit a useful forward propagation [36]. The lack of exploration ability still persists in the more accurate but resource consuming iterative *IMP* approach.

Reviving the key ideas of *Han et al*. [9] and *Narang et al*. [26] (comparable to [25]), we introduce a lightweight and powerful method called *Adaptive Regularized Training* (*ART*) to obtain highly sparse NNs, which implicitly "removes" weights with increasing regularization until a desired sparsity level is reached. *ART* strongly regularizes the weights before magnitude pruning. First, a dense NN is pre-trained until convergence. In the second stage, the NN is trained with an increasing and weight decaying regularization until the hypothetical magnitude pruned NN performs on par with the dense counterpart. Lastly, we apply magnitude pruning and fine-tune the NN without regularization. Avoiding binary masks in the second stage allows exploration and regularization forces the exploitation of weights that remain in the sparse NN. We introduce the new regularization approach *HyperSparse* for the second stage that overcomes static regularization like *Lasso* [38] or *Weight Decay* [43] and adapts to the weight magnitude by penalizing small weights. *HyperSparse* balances the exploration/exploitation tradeoff and thus increases the accuracy while leading to faster convergence in the second stage. The combination of our regularization schedule and *HyperSparse* improves the classification accuracy and optimization time significantly, especially in high sparsity regimes with up to 99.8% zero weights. We evaluate our method on CIFAR-10/100 [18] and TinyImageNet [5] with ResNet-32 [10] and VGG-19 [32].

Moreover, we analyze the gradient and weight distribution during regularized training, showing that *HyperSparse* leads to faster convergence to sparse NNs. The experiments also shows that the claim of [33], that optimal sparse NNs can be obtained via simple weight distribution heuristics, does not hold in general. Finally, we analyze the process of compressing dense NNs into sparse NNs and show that the highest weights in NNs do not encode decision rules for a diverse set of classes with equal priority.

In summary, this paper

- introduces *HyperSparse*, a superior adaptive regularization loss that implicitly promotes configurable network sparsity by balancing the exploration and exploitation tradeoff.
- introduces the novel framework *ART* to obtain sparse networks using regularization with increasing leverage, which improves the optimization time and classification accuracy of sparse neural networks, especially in high sparsity regimes.
- analyzes the continuous process of compressing patterns from dense to sparse neural networks.

2. Related Work

Sparse Learning methods that find binary masks to remove a predefined amount of weights can be categorized as static or dynamic (*e.g*., in [3, 11, 13]). According to [3], in dynamic sparse training *"[...] removed elements [from masks] have chances to be grown back if they potentially benefit to predictions"* whereas static training incorporates fixed masks.

Static methods are usually based on *Frankle et al*. [8], who introduce *LTH* and show that well performing sparse NNs in random initialised NNs can be found after dense training via magnitude pruning. The magnitude pruning method is improved by *IMP* [15] that iterates the process. Replacing the time consuming training procedure, methods like *SNIP* [19] or *GraSP* [41] find sparse NNs in random initialized dense NNs using a single network prediction and its gradients. To also address the risk of layer collapse during pruning, *SynFlow* [36] additionally conserves the total flow in the network. Contrary to the latter works, *Su et al*. [33] claim that appropriate sparse NNs do not depend on data or weight initialization and provide a general heuristic for the distribution of weights.

Different from static methods, dynamic methods prune and re-activate zero elements in the binary mask. The weights that are reactivated can be selected randomly [24] or determined by the gradients [2, 3, 7]. For example, *RigL* [7] iteratively prunes weights with low magnitude and therefore reactivates weights with highest gradient. Also, modern dynamic methods utilize continuous masks. For example, *Tai et al*. [35] relax the *IMP* framework by introducing a parameterized softmask to obtain a weighted average between *IMP* and *Top-KAST* [14]. Similar, [23, 30] relaxes the binary mask and optimizes its $\mathcal{L}_0$-norm. Another way is to inherently prune the model, *e.g*., by reducing the gradients of weights with small magnitude [31]. Compared to static methods, *Liu et al*. [21, 22] show that dynamic sparse training methods overcomes most static methods by allowing weight exploration.

Another property to distinguish modern sparse learning methods is the complexity during mask generation, *e.g*., as done by *Schwarz et al*. [31]. The more resource efficient *sparse→sparse* methods sustain sparse NNs during training [3, 7, 19, 31, 33, 36, 41], whereas *dense→sparse* methods utilize all parameters before finding the final mask [8, 14, 15, 23, 30, 35].

However, as explained later, our approach belongs to *dense→sparse* methods that inherently reduce the model complexity without masking before magnitude pruning to obtain a static sparse mask for fine-tuning. We want to mention the primary works of *Han et al*. [9], *Narang et al*. [26] and *Molchanov et al*. [25] whose combination is a role model for us. *Han et al*. use $\mathcal{L}_1$ and $\mathcal{L}_2$ regularization to reduce the number of non-zero elements during training. Their early framework uses regularization without bells and whistles and has no ability to control the sparsity level. *Narang et al.* and *Molchanov et al.* remove weights in fine-grained portions with an increasing removal-threshold, but do not incorporate weight exploration.

Interpretability and Understanding of machine learning is closely related to sparse learning and is also addressed in this paper. There is an increasing number of works in recent years that utilize sparse learning for other benefits, for example, to find interpretable correlations between feature- and image-space [37] or to visualize inter-class ambiguities [17]. The work of *Paul et al.* [27] gives details about the early learning stage which is crucial, *e.g.*, to determine memorization of label noise [16]. They show that most data is not necessary to obtain suitable subnetworks. The general relationship between *LTH* and generalization is investigated in [29]. *Varma et al.* [34] show that sparse NNs are better suited in data limited and noisy regimes. On the other hand *Hooker et al.* [12] show that sparse NNs have non-trivial impact on ethical bias by investigating which samples are "forgotten" first during network compression. The underlying research question of the latter work is altered to *"Which samples are compressed first?"* and discussed in this paper.

3. Method

Sparsification aims to reduce the number of non-zero weights in a NN. To address this problem, we use a certain schedule for regularization such that small weights converge to zero and our model implicitly becomes sparse. In Sec. 3.1, we formally define the sparsification problem. Then, we present *Adaptive Regularized Training* (ART) in Sec. 3.2, which iteratively increases the leverage of regularization to maximize the number of close-to-zero weights. Moreover, we introduce our regularization loss *HyperSparse* in Sec. 3.3 that is integrated in *ART*. It simultaneously allows the exploration of new topologies while exploiting weights of the final sparse subnetwork.

3.1. Preliminaries

We consider a NN $f(W,x)$ with topology f and weights W that is trained to classify images from a dataset $S=\{(x_n,y_n)\}_{n=1}^{N}$, where y_n is the ground truth class to an image sample x_n. The training is structured in epochs, which are iterative optimizations of the weights $W=\{w_1,\dots,w_D\}$ over all samples in S to minimize the loss objective $\mathcal{L}$. The obtained weights after epoch e are denoted as W_e, with W_0 denoting the weights before optimization. Furthermore, the classification accuracy of a NN is measured by a rating function $\psi(W)$.

The goal in sparsification is to reduce the cardinality of W by removing a pre-defined ratio of weights κ, while maximizing $\psi(W)$. The network is pruned by the Hadamard product $m\odot W$ of a binary mask $m\in[0,1]^D$ and the model-weights W. The mask is usually created by applying magnitude pruning $m=\nu(W)$ [2, 3, 8, 15], which is a technique that sets the κ-lowest weights to zero.

3.2. Adaptive Regularized Training (ART)

Regularization losses like the L_1-norm ($Lasso$-regression) [38] or L_2-norm [43] are used to prevent overfitting by shrinking the magnitude of weights. We use this effect in *ART* for sparsification, as weights with low magnitude have low effect on changing the output and thus can be removed with only little impact on $\psi(W)$.

Regularization during training can be expressed as a mixed loss

$$\mathcal{L}_{\text{total}}=\mathcal{L}_{\text{class}}+\lambda_{\text{init}}\cdot\eta^{e}\cdot\mathcal{L}_{\text{reg}}\,, \tag{1}$$

where $\mathcal{L}_{\text{class}}$ is the classification loss and $\mathcal{L}_{\text{reg}}$ the regularization loss. The gradient of $\mathcal{L}_{\text{reg}}$ shrinks a set of weights to approximately zero and creates a inherent sparse network of an undefined pruning rate [38]. Increasing η leverages the regularization $\mathcal{L}_{reg}$ in an ascending manner, but current approaches use a fixed regularization rate $\eta=1$ [3, 9, 25].

After unregularized training of a dense NN to convergence, *ART* employs the standard regularization framework and modifies it by setting $\eta>1$ and a low initialisation of λ_{init}. Subsequently, the regularization loss $\mathcal{L}_{\text{reg}}$ has almost no effect on $\mathcal{L}_{\text{total}}$ in the beginning, but starts to shrink weights without much impact on $\mathcal{L}_{\text{class}}$ to zero. However, it allows every weight w_i to potentially get a high magnitude such that w_i is shifted into the sparse NN of highest weights (exploration). With increasing regularization, the influence of the gradient $\frac{d\mathcal{L}_{\text{reg}}}{dw_i}$ on w_i increases and is more likely to overcome the gradient $\frac{d\mathcal{L}_{\text{class}}}{dw_i}$. Regularization impedes proper exploration of small weights by pulling the magnitude to zero. On the other hand, the larger weights

Algorithm 1 Adaptive Regularized Training (ART)

Parameter: Pre-trained weights W_{pre}, initial rate λ_{init}, rating function $\psi(W)$, magnitude pruning $\nu(W)$, increasing factor $\eta>1$, classification loss $\mathcal{L}_{\text{class}}$, regularization loss $\mathcal{L}_{\text{reg}}$, training data S, optimizer SGD$(W,\mathcal{L},S)$
Result: Best weights for fine-tuning W_{best}

$W_0, W_{\text{best}} \leftarrow W_{\text{pre}}$
$e \leftarrow 0$
while $\psi(\nu(W_{\text{best}})\odot W_{\text{best}})<\psi(W_e)$ **do**
  $W_{e+1}\leftarrow \text{SGD}(W_e,\mathcal{L}_{\text{class}}+\lambda_{\text{init}}\cdot\eta^{e}\cdot\mathcal{L}_{\text{reg}},S)$
  if $\psi(\nu(W_{e+1})\odot W_{e+1})>\psi(\nu(W_{\text{best}})\odot W_{\text{best}})$ **then**
    $W_{\text{best}}\leftarrow W_{e+1}$
  end if
  $e\leftarrow e+1$
end while

need to be exploited to conserve the classification results. Therefore, our increasing regularization continually shifts the exploration/exploitation tradeoff from exploration to exploitation. The method allows reordering weights to find better topologies, but forces to exploit the highest weights regarding the classification task. Due to the increasing number of weights that are approximately zero, the dense model converges to a inherently sparse model. We stop the regularized training if the NN with best pruned weights $\psi(\nu(W_{\text{best}}) \odot W_{\text{best}})$ has higher accuracy than with the latest unpruned weights $\psi(W_e)$ and choose W_{best} as our candidate for fine-tuning.

The overall training pipeline is defined as follows:

- Step 1: Pre-train dense model until convergence without regularization.
- Step 2: Remove weights implicitly using *ART* as described in algorithm 1.
- Step 3: Apply magnitude pruning and fine-tune pruned network until convergence.

ART relaxes the iterative *IMP* approach that prunes the least important weights over certain iterations. Analogous to the increasing pruning ratio in standard iterative methods, we iteratively increase the amount of weights that are close to zero and thus approximate a binary mask implicitly.

3.3. HyperSparse Regularization

The latter Section 3.2 describes the process of shrinking weights in W by penalizing with ascending regularization. A drawback of this procedure is that also weights that remain after pruning are penalized by the regularization. This negatively affects the exploitation regarding the main task. Thus, remaining weights should not be penalized. On the other hand, if small weights are strongly penalized, the desired exploration property of dynamic pruning methods to "grow" back these elements is restricted. To address this tradeoff between exploitation and exploration, we introduce the sparsity inducing adaptive regularization loss *HyperSparse*.

Incorporating the *Hyper*bolic Tangent function applied on the magnitude denoted as $t(\cdot) = \tanh(|\cdot|)$ for simplicity, the *HyperSparse* loss is defined as

$$\mathcal{L}_{\text{HS}}(W) = \frac{1}{A} \sum_{i=1}^{|W|} \left(|w_i| \sum_{j=1}^{|W|} t(s \cdot w_j) \right) - \sum_{i=1}^{|W|} |w_i| \quad \text{with} \quad A := \sum_{w \in W} t(s \cdot w) \quad \text{and} \quad \forall w \in W: \quad \frac{dA}{dw} = 0, \tag{2}$$

where A is treated as a pseudo-constant in the gradient computation and s is an alignment factor that is described later. The regularization penalizes weights depending on the gradient and can vary for different weights. The gradient of *HyperSparse* with respect to a weight w_i is approximately

$$\frac{d\mathcal{L}_{\text{HS}}(W)}{dw_i} = \text{sign}(w_i) \cdot \frac{t'(s \cdot w_i) \cdot \sum_{j=1}^{|W|} |w_j|}{\sum_{j=1}^{|W|} t(s \cdot w_j)}, \quad \text{with} \quad w_i, w_j \in W, \quad t'(\cdot) \in (0, 1]. \tag{3}$$

The derivative $t' = \frac{dt}{dw_i}$ converges towards 1 for small magnitudes $|w_i| \approx 0$ and towards 0 for large magnitudes $|w_i| \gg 0$. Thus, the second term in Eq. (3) is adaptive to the weights and highly penalizes small magnitudes, but is breaking down to zero for large ones. Details for the gradient calculation and analysis can be found in the supplementary material, Sec. D.

The alignment factor s is mandatory to exploit the aforementioned properties for the sparsification task with a specific pruning rate κ. Since $\mathcal{L}_{\text{HS}}$ is dependent on the weights magnitude, but there is no determinable value range for weights, our loss $\mathcal{L}_{\text{HS}}$ is not guaranteed to adapt reasonably to a given W. For example, considering a fixed $s = 1$ and all weights in W are close to zero, the gradient from Eq. (3) results into nearly the same value for every weight. Therefore, we adapt s to the smallest weight $|w_\kappa|$ that would remain after magnitude pruning, such that $t'''(s \cdot w_\kappa) = 0$, which is the point of inflection of t'. According to this alignment, the gradients in Eq. (3) of remaining weights $|w| \geq |w_\kappa|$ are shifted closer to 1 and are increased for weights $|w| \leq |w_\kappa|$, while adhering a smooth gradient from remaining to removed weights. Moreover, the denominator in Eq. (3) decreases over time, if more weights in W are close to zero subsequent to ascending regularization. The gradient for different weight distributions of a NN based on *HyperSparse* is shown in Fig. 1 and visualizes the described gradient behavior of adaptive weight regularization.

4. Experiments

This section presents experiments showing that our proposed method *ART* outperforms comparable methods, especially in extreme high sparsity regimes. Our experimental setup is described in Sec. 4.1. In the subsequent section, we show that *HyperSparse* has a large positive impact on the optimization time and classification accuracy. This improvement is explained by analyzes of the tradeoff between exploration and exploitation, the gradient and weight distribution in Sec. 4.3 and 4.4. Finally, we analyze and discuss the compression behaviour during regularized training and derive further insights about highest magnitude weights in Sec. 4.5.

4.1. Experimental Setup

We evaluate *ART* on the datasets CIFAR-10/100 [18] and TinyImageNet [5] to cover different complexities, given by

$\psi\uparrow$		ResNet-32 ⟶						VGG-19 ⟶					
		$\kappa=90\%$	$\kappa=95\%$	$\kappa=98\%$	$\kappa=99\%$	$\kappa=99.5\%$	$\kappa=99.8\%$	$\kappa=90\%$	$\kappa=95\%$	$\kappa=98\%$	$\kappa=99\%$	$\kappa=99.5\%$	$\kappa=99.8\%$
CIFAR-10	No Mask	94.70±0.19						93.84±0.12					
	SNIP [19]	92.72±0.18	91.35±0.15	88.02±0.27	83.94±0.39	71.64±7.46	23.72±21.11	93.63±0.25	93.36±0.20	76.12±21.96	10.00±0.00	10.00±0.00	10.00±0.00
	GraSP [41]	92.86±0.19	91.80±0.23	89.00±0.24	85.63±0.28	80.25±0.67	62.56±11.25	92.97±0.04	92.79±0.24	92.16±0.14	91.27±0.15	51.45±38.46	10.00±0.00
	SRatio [33]	93.02±0.17	91.85±0.16	88.91±0.16	85.97±0.22	80.73±0.37	64.38±0.50	93.86±0.19	93.58±0.20	92.33±0.24	91.14±0.21	89.14±0.15	43.64±20.04
	LTH [8]	92.68±0.32	91.45±0.19	88.48±0.15	85.99±0.30	81.19±0.40	69.34±0.43	93.71±0.17	93.31±0.15	41.17±42.76	10.00±0.00	10.00±0.00	10.00±0.00
	IMP [15]	**94.69±0.17**	**94.00±0.18**	91.35±0.18	87.35±0.55	82.00±0.34	69.12±0.50	<u>93.96±0.17</u>	**94.02±0.07**	93.48±0.24	91.29±0.29	25.43±34.50	10.00±0.00
	RigL [7]	94.21±0.10	93.07±0.22	90.65±0.17	86.50±0.83	62.89±5.18	32.78±4.11	93.48±0.13	92.92±0.14	91.41±0.15	89.08±0.37	84.79±0.90	70.81±1.10
	ART + $\mathcal{L}_1$	94.20±0.16	93.14±0.16	91.34±0.46	88.18±1.07	84.52±1.24	79.35±1.85	**93.97±0.13**	93.82±0.10	**93.85±0.12**	<u>93.10±0.23</u>	<u>92.17±0.25</u>	90.42±0.50
	ART + $\mathcal{L}_2$	93.49±0.21	92.91±0.24	89.60±0.73	85.80±2.38	82.24±0.60	71.73±0.88	93.18±0.18	92.65±0.40	79.38±4.92	78.85±8.74	72.68±2.67	56.28±26.33
	ART + $\mathcal{L}_{HS}$ (no preTrain)	93.13±0.13	92.85±0.18	<u>91.79±0.14</u>	<u>90.79±0.30</u>	<u>89.01±0.21</u>	**84.64±0.51**	93.58±0.12	93.53±0.09	93.15±0.12	92.56±0.08	92.12±0.13	<u>91.24±0.09</u>
	ART + $\mathcal{L}_{HS}$	<u>94.22±0.20</u>	<u>93.76±0.18</u>	**92.69±0.22**	**91.16±0.28**	**89.35±0.23**	<u>84.45±0.55</u>	93.93±0.20	<u>93.83±0.10</u>	<u>93.75±0.23</u>	**93.51±0.15**	**92.91±0.10**	**91.62±0.19**
CIFAR-100	No Mask	74.60±0.14						72.88±0.34					
	SNIP [19]	69.78±0.22	65.54±0.26	53.20±0.30	37.45±1.42	14.76±3.35	04.52±2.16	72.76±0.20	71.50±0.27	25.34±9.16	1.00±0.00	1.00±0.00	1.00±0.00
	GraSP [41]	69.64±0.38	66.84±0.14	59.59±0.30	49.42±1.04	36.46±2.73	15.62±3.20	71.10±0.13	70.39±0.17	68.25±0.45	65.84±0.36	59.56±0.47	1.10±0.10
	SRatio [33]	69.80±0.18	67.08±0.41	60.44±0.32	51.60±0.63	38.57±0.75	18.35±0.97	72.84±0.32	71.67±0.19	68.84±0.38	65.00±0.22	51.16±2.67	1.02±0.04
	LTH [8]	69.23±0.31	66.80±0.49	60.28±0.10	51.92±0.11	40.18±0.28	20.31±1.63	72.55±0.27	70.46±0.26	9.80±15.98	1.00±0.00	1.00±0.00	1.00±0.00
	IMP [15]	<u>73.91±0.37</u>	71.21±0.36	64.67±0.29	55.89±0.34	41.53±0.74	14.97±0.69	**73.92±0.33**	**73.77±0.32**	70.99±0.34	4.03±4.69	1.00±0.00	1.00±0.00
	RigL [7]	73.09±0.29	71.46±0.37	64.46±0.36	45.58±1.78	21.80±1.54	8.47±4.24	72.00±0.24	70.42±0.30	67.48±0.36	63.31±0.51	55.56±1.33	24.57±13.21
	ART + $\mathcal{L}_1$	73.16±0.45	70.98±0.48	66.10±0.76	59.36±2.08	50.50±3.43	37.43±1.64	73.16±0.20	<u>72.80±0.20</u>	<u>71.23±0.25</u>	<u>69.18±0.22</u>	<u>65.71±0.63</u>	59.08±1.07
	ART + $\mathcal{L}_2$	71.39±0.60	68.21±1.25	58.49±3.93	56.61±0.88	47.11±1.00	28.73±1.18	61.54±3.91	55.22±7.34	44.42±6.14	39.40±4.78	26.94±23.75	29.78±16.25
	ART + $\mathcal{L}_{HS}$ (no preTrain)	72.49±0.35	<u>71.57±0.36</u>	<u>69.08±0.12</u>	<u>65.48±0.28</u>	<u>59.49±0.53</u>	**48.63±0.66**	71.49±0.42	70.24±0.67	68.57±0.38	67.59±0.47	65.59±0.17	<u>61.66±0.50</u>
	ART + $\mathcal{L}_{HS}$	**74.08±0.13**	**72.85±0.31**	**70.08±0.37**	**65.86±0.26**	**59.58±0.26**	<u>48.31±0.53</u>	<u>73.23±0.24</u>	72.70±0.41	**71.97±0.13**	**70.83±0.23**	**69.02±0.36**	**64.53±0.24**
TinyImageNet	No Mask	62.87±0.27						61.41±0.12					
	SNIP [19]	55.23±0.47	48.78±0.40	34.93±0.83	23.20±1.41	12.25±1.50	3.19±1.52	61.47±0.16	59.00±0.20	4.77±4.23	0.50±0.00	0.50±0.00	0.50±0.00
	GraSP [41]	56.16±0.25	51.52±0.47	40.32±2.24	28.41±1.26	15.81±2.30	4.29±3.73	60.50±0.08	58.97±0.14	56.70±0.12	53.12±0.49	43.76±0.40	0.51±0.03
	SRatio [33]	55.19±0.35	51.70±0.48	44.04±0.36	34.14±0.12	8.31±1.26	1.98±0.29	61.21±0.19	59.10±0.32	55.94±0.24	51.13±0.34	39.76±0.32	0.50±0.00
	LTH [8]	55.72±0.22	52.22±0.48	43.73±0.85	33.22±0.39	20.78±0.40	7.65±0.58	59.91±0.59	58.74±0.43	56.38±0.16	54.02±0.60	46.78±0.81	2.89±2.32
	IMP [15]	<u>60.71±0.24</u>	56.97±0.26	47.29±0.57	33.21±0.11	8.59±0.67	2.40±0.34	**62.42±0.32**	61.28±0.23	57.39±0.10	54.26±0.26	47.19±0.28	3.10±0.96
	RigL [7]	59.29±0.21	55.53±0.16	44.72±1.34	26.07±1.59	8.76±0.30	4.51±0.37	61.47±0.29	**61.69±0.41**	<u>59.41±0.53</u>	54.59±0.68	47.11±0.62	20.81±0.99
	ART + $\mathcal{L}_1$	58.00±0.49	55.30±0.94	46.59±0.45	39.34±1.32	29.80±3.53	18.06±2.89	61.29±0.24	60.21±0.31	57.04±0.53	<u>54.61±0.77</u>	51.27±1.80	43.59±1.33
	ART + $\mathcal{L}_2$	56.94±0.76	51.40±0.99	43.03±2.23	35.19±1.42	24.62±1.96	7.79±0.59	60.62±0.69	51.10±5.94	47.96±7.73	45.90±8.76	30.32±17.94	5.55±11.30
	ART + $\mathcal{L}_{HS}$ (no preTrain)	57.96±0.39	<u>57.01±0.34</u>	<u>53.27±0.32</u>	<u>47.32±0.46</u>	<u>40.53±0.26</u>	**28.96±0.69**	60.95±0.27	59.67±0.17	56.72±0.48	53.79±0.33	<u>51.66±0.19</u>	<u>47.49±0.17</u>
	ART + $\mathcal{L}_{HS}$	**60.97±0.18**	**58.78±0.28**	**53.92±0.14**	**47.97±0.42**	**40.68±0.82**	<u>28.95±0.52</u>	<u>61.55±0.24</u>	<u>61.36±0.31</u>	**59.79±0.25**	**58.01±0.21**	**55.34±0.22**	**49.44±0.18**

Table 1: Classification accuracy of sparse NNs for varying pruning rates κ based on our proposed method *ART* with $\mathcal{L}_1$, $\mathcal{L}_2$, and *HyperSparse* regularization $\mathcal{L}_{HS}$ compared to dense models, and masks obtained by *SNIP* [19], *GraSP* [41], *SRatio* [33], *LTH* [8], *IMP* [15] and *RigL* [7]. The best accuracy per configuration is highlighted, the second is underlined. It shows that our method outperforms *IMP* significantly in the domain of high sparsity. We recommend the pdf version and zooming in.

#Epochs ↓		ResNet-32 ⟶			VGG-19 ⟶		
	κ:	90%	98%	99.5%	90%	98%	99.5%
CIFAR-10	$\mathcal{L}_1$	34.2±3.1	68.2±4.8	94.2±8.0	5.8±0.4	24.2±2.4	56.0±3.0
	$\mathcal{L}_2$	75.6±63.6	49.6±92.45	116.6±19.5	116.2±5.3	175.8±15.8	178.4±9.7
	$\mathcal{L}_{HS}$	**26.6±1.8**	**55.4±2.3**	**77.8±1.9**	**4.0±0.0**	**18.2±1.6**	**42.8±1.6**
CIFAR-100	$\mathcal{L}_1$	53.2±2.8	77.5±4.1	101.3±9.6	11.2±0.4	47.7±2.1	66.5±4.6
	$\mathcal{L}_2$	112.0±7.6	141.8±17.0	120.0±11.4	153.2±2.9	168.8±6.9	75.4±122.7
	$\mathcal{L}_{HS}$	**39.2±0.84**	**63.4±0.5**	**88.2±0.8**	**8.2±0.4**	**35.8±0.4**	**55.8±0.4**
Tiny-ImageNet	$\mathcal{L}_1$	52.0±3.4	81.6±3.5	101.2±10.8	20.8±0.4	43.2±3.5	59.0±8.0
	$\mathcal{L}_2$	110.2±7.3	129.8±11.6	93.2±45.5	27.6±81.1	148.4±22.3	107.4±94.6
	$\mathcal{L}_{HS}$	**36.6±1.3**	**67.8±3.8**	**100.0±9.3**	**14.4±0.5**	**34.0±0.0**	**52.3±1.5**

Table 2: Number of epochs with regularization to obtain the final mask, evaluated for multiple datasets, network topologies, and pruning rates κ. It shows that our *HyperSparse* $\mathcal{L}_{HS}$ loss reduces the training time significantly.

a varying number of class labels. Furthermore, we use different model complexities, where ResNet-32 [10] is a simple model with 1.8 M parameters and VGG-19 [32] is a complex model with 20 M parameters. Note that we use the implementation given in [33]. As explained in Sec. 3.2, we group our training in 3 steps. First we train our model for 60 epochs until convergence (step 1), using a constant learning rate of 0.1. In the following regularization step, we initialize the regularization with $\lambda_{\text{init}} = 5 \cdot 10^{-6}$, $\eta = 1.05$, and use the same learning rate as used in pre-training. The fine-tuning-step (step 3) is similar to [33], as we train for 160 epochs in CIFAR-10/100 and for 300 epochs on Tiny-ImageNet, using a learning rate of 0.1 and apply a multiplied decay of 0.1 at 2/4 and 3/4 of the total number of epochs. We also adapt the batch size of 64 and weight-decay of 10^{-4}. All experiments are averaged over 5 runs.

We compare our method *ART* to *SNIP* [19], *Grasp* [41], *SRatio* [33], and *LTH* [8] similar as done in [33, 40]. In addition we evaluate *IMP* [15] and *RigL* [7] as dynamic pruning methods. For comparability, all competitors in our experiments are trained with the same setup as given in the fine-tuning-step. To improve the performance of *RigL*, we extend the training duration by 360 epochs. Further details are given in the supplementary material, Sec. A.

4.2. Sparsity Level

In this section, we compare the performances of *ART* to other methods on different sparsity levels $\kappa \in \{90\%, 95\%, 98\%, 99\%, 99.5\%, 99.8\%\}$, using different datasets and models. To demonstrate the advantages of our novel regularization loss, we additionally substitute *HyperSparse* with $\mathcal{L}_1$ [38] and $\mathcal{L}_2$ [43]. Table 1 shows the resulting accuracies with standard deviations.

Our method *ART* combined with *HyperSparse* outper-

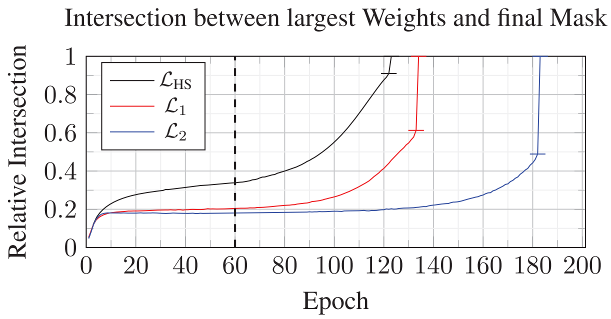


Figure 2: Intersection of the set of weights with highest magnitude during training and the final mask measured during *ART* with ResNet-32, CIFAR-100, pruning rate $\kappa = 98\%$ and different regularization losses. Horizontal bars mark the intersection one epoch before pruning and the dashed line at epoch 60 indicates the start of regularization. Our *HyperSparse* loss reduces the optimization time and the high intersection before pruning suggests a higher stability during regularization, which leads to better exploitation.

formes the methods *SNIP* [19], *Grasp* [41], *SRatio* [33], *LTH* [8] and *RigL* [7] on all sparsity levels. Considering the high sparsity of 99%, 99.5% and 99.8%, all competitors drop drastically in accuracy, even to the minimal classification bound of random prediction for SNIP and *LTH* using VGG-19. However, *ART* is able keep high accuracy even on extreme high sparsity levels. In comparison to the regularization losses $\mathcal{L}_1$ and $\mathcal{L}_2$, our *HyperSparse* loss achieves higher accuracy in nearly all settings and even minimizes the variance. If we skip the Pre-train-step (step 1) of *ART*, the performance slightly drops. However, *ART* without pretraining still has good results.

Moreover, we present the number of trained epochs for the regularization phase (step 2) in Tab. 2. In almost all cases, *HyperSparse* requires less epochs to terminate compared to $\mathcal{L}_1$ and $\mathcal{L}_2$ and converges faster to a well performing sparse model. As a second aspect, *ART* dynamically varies the training-length to the sparsity level, model and data complexity. Thus, *ART* trains longer if higher sparsity is required or the model has more parameters and is more complex like VGG-19. In comparison of the two datasets CIFAR-10 and CIFAR-100, which have the same number of training samples and thus the same number of optimization steps per epoch, *ART* extends the training-length for the more complex classification problem in CIFAR-100.

ART trains the model for 60 epochs in pre-training (step 1) and 160 epochs in fine-tuning (step 3). Considering the dynamic training-length in step 2, the epochs of *ART* using $\mathcal{L}_{\text{HS}}$ sum up from 226.2 to 301.2 epochs in mean. In comparison, iterative pruning methods are computationally much more expensive, since each model is trained multiple times. For example, IMP [15] requires 860 epochs on CIFAR-10/100 in our experiments.

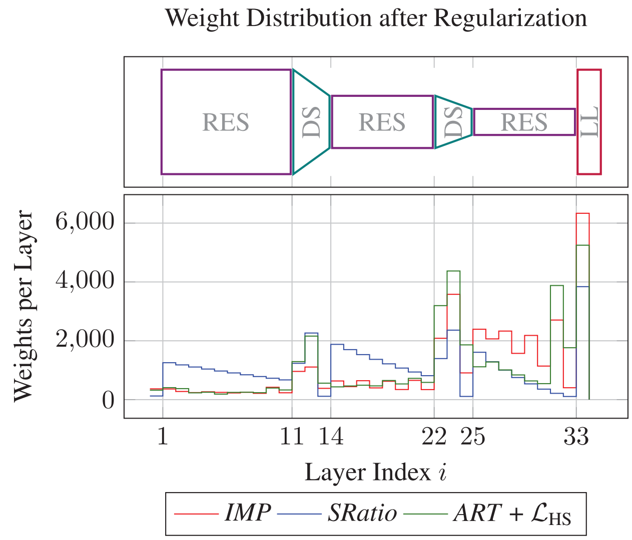
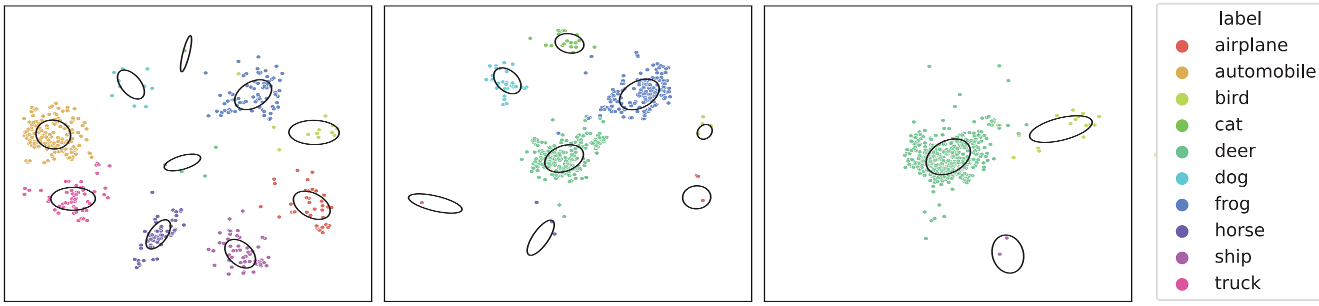


Figure 3: Distribution of weights per layer after pruning in a ResNet-32 model that is trained on CIFAR-100 with pruning rate $\kappa = 98\%$. Layer index i describes the execution order. We group the model in residual blocks (RES), downsampling blocks (DS) and the linear layer (LL). Our method distributes the weights comparable to *IMP* [15], but it has more weights in the downsampling layers.

4.3. Exploration and Exploitation aware Gradient

The training-schedule of *ART* allows to explore new topologies of sparse networks, while compressing the dense network into the remaining weights that are exploited to minimize the loss $\mathcal{L}_{\text{class}}$. To reduce the tradeoff between exploration and exploitation, our regularization loss *HyperSparse* penalizes small weights with a higher regularization and forces the most weights to be close to zero, while preserving the magnitude of weights that remain after pruning. To highlight the beneficial behaviour of *HyperSparse*, this section visualizes and analyzes the gradient. Fig. 1 shows the values and the corresponding gradients of all weights, sorted by the weights magnitude. Note that we only focus on the second step of *ART*, where the regularization is incorporated. Epoch 0 represents the first epoch using regularization.In the lower subfigure, we observe that the gradient of *HyperSparse* with respect to weights larger than $|w_\kappa|$ is closer to 0 than for smaller weights. In comparison, $\mathcal{L}_1$ remains constantly 1 for all weights. The effect of increasing regularization of small weights is stronger for networks with more weights close to zero and therefore amplifies over time, since increasing regularization shrinks the weights magnitude. For example, epoch 40 shows higher gradients for small weights compared to epoch 0, while having more weights with lower magnitude. The pruning-rate κ dependent $\mathcal{L}_{\text{HS}}$ increases the gradient for small weights $|w| < |w_\kappa|$ over time but conserves the low gradient of larger weights $|w| > |w_\kappa|$ approximately at 0 to favor



(a) Dense Model ($\kappa = 0\%$) (b) Low Sparsity ($\kappa = 90\%$) (c) High Sparsity ($\kappa = 99.8\%$)

Figure 4: First 5% CIFAR-10 samples that are compressed into the remaining highest weights after pruning with $\kappa \in \{0\%, 90\%, 99.8\%\}$ deduced by the CP-metric. While dense networks learn samples approximately uniform-distributed over classes, the highest weights compress decision rules only for a subset of classes in the early learning stage. Note that we sampled by factor 10 for visualization purposes and ellipses represent the double standard deviation of cluster centers.

CP	Human Label Errors	CIFAR-10 Class					
		deer	bird	cat	truck	airplane	horse
Dense Model (0%)	0	0.482	0.554	0.639	0.400	0.452	0.429
	1	0.548	0.632	0.722	0.479	0.547	0.493
	2	0.653	0.741	0.808	0.588	0.676	0.710
	3	0.760	0.823	0.862	0.695	0.783	0.769
Low Sparsity (90%)	0	0.166	0.499	0.494	0.567	0.580	0.524
	1	0.217	0.573	0.601	0.613	0.671	0.570
	2	0.292	0.678	0.721	0.676	0.790	0.737
	3	0.392	0.772	0.796	0.710	0.841	0.808
High Sparsity (99.8%)	0	0.056	0.296	0.387	0.622	0.798	0.824
	1	0.061	0.317	0.411	0.647	0.835	0.835
	2	0.069	0.342	0.439	0.677	0.882	0.893
	3	0.081	0.363	0.469	0.699	0.902	0.929

Table 3: *Compression Position* (see Sec. 4.5) for dense NNs (during pre-training) and κ pruned NNs (during regularization) for six CIFAR-10 classes. Samples of a class are split into 4 subsets according to the number of human label errors in CIFAR-N to indicate the difficulty. In sparse networks, different classes are compressed at different times and difficult samples are compressed later. All classes and pruning rates can be found in the supplementary material, Tab. 2.

exploitation. During optimization, the gradient remains smooth and increases slowly for weights that are smaller, but close to $|w_\kappa|$. This favors exploration in the domain of weights close to w_κ. Therefore, the model becomes inherently sparse and the behaviour shifts continuously from exploration to exploitation.

4.4. Reordering Weights

We use the regularization loss with ascending leverage to find a reasonable set of weights, that remain after pruning. We implicitly do this by shrinking small weights close to zero. During training, weights are reordered and thus can change the membership from the set of pruned to remaining weights, and vice versa. We analyze the reordering procedure in Fig. 2, which shows the intersection of the intermediate and final mask over all epochs, using different regularization losses in *ART*. The model is pre-trained to convergence without regularization for the first 60 epochs (step 1) and with regularization in further epochs (step 2). Fine-tuning is not visualized (step 3). After pre-training, the highest weights only intersects up to 20% with the final mask obtained by $\mathcal{L}_1$ and $\mathcal{L}_2$, while *HyperSparse* leads to an intersection of approximately 35%. This results show *HyperSparse* changes less parameter while reordering weights, which implies that more structures from the dense model are exploited. It also shows that *HyperSparse* has a significantly smaller learning duration than $\mathcal{L}_1$ and $\mathcal{L}_2$. The horizontal bars point to the intersection before last training epoch and show that $\mathcal{L}_1$ and $\mathcal{L}_2$ only intersect by 60% and 50%, while *HyperSparse* is getting very close to the final mask with more than 90% intersection. This indicates that *HyperSparse* finds a more stable set of high valued weights and reduces exploration, as the mask has less variation in the final epochs. More results for other training settings are shown in the supplementary material, Sec. B.

Moreover, we analyze the resulting weight distribution of our method and compare it to *IMP* [15] and *SRatio* [33]. Fig. 3 shows the number of remaining weights per layer for ResNet-32 that consists of three scaling levels, which end up with the linear layer (LL). Each scaling level consists of four residual blocks (RES), which are connected by a downsampling-block (DS). The basic topology of *ART* and *IMP* looks similar, since both methods show a constant keep-ratio over the residual blocks. Furthermore, *ART* and *IMP* use more parameters in downsampling and linear layers. We conclude that these two layer types require more weights and consequently are more important to the model. The higher accuracy discussed earlier suggest that our method exploit these weights better. To show that this results are also obtained on other datasets, models, and sparsity levels, we describe further weight distributions in the supplementary material, Sec. C and show that the number of parameters in the linear layer decreases drastically for a small set of classes in CIFAR-10. Moreover, the compared method *SRatio* assumes that suitable sparse networks can be obtained using handcrafted keep-ratios per layer. It has a quadratic decreasing keep-ratio that can be observed in Fig. 3. As shown in Tab. 1, our method *ART* performs significantly better than *SRatio* and therefore we deduce that fixed keep-ratios have an adverse effect on performance. Reordering weights during training favors well performing sparse NNs, especially in high sparsity regimes.

4.5. What do networks compress first?

Along with the introduction of *ART*, we are faced with the question of which patterns are compressed first into the large weights that remain after magnitude pruning during regularization. This question is in contrast to Hooker's question *"What Do Compressed Deep Neural Networks Forget?"* [12] and challenges the fundamental assumption of magnitude pruning, which assumes large weights to be most important. In this section, we analyze the chronological order of how samples are compressed and introduce the metric *Compression Position* (CP) to determine it.

According to our method, regularization starts at epoch e_S and ends at e_E and therefore the weights W have different states $\mathcal{W} = \{W_e\}_{e=e_S}^{e_E}$ during training. We measure the individual accuracy over time ψ_I reached by the sparse network for a training sample $(x, y) \in S$, defined by

$$\psi_\text{I}(x, y, f, \mathcal{W}) = \frac{\left|\left\{W_e \in \mathcal{W} \mid f(\nu(W_e) \odot W_e, x) = y\right\}\right|}{e_E - e_S}. \quad (4)$$

After computing the individual accuracy for all samples $\Psi = \left\{\psi_\text{I}(x_n, y_n, f, \mathcal{W})\right\}_{n=1}^{N}$ and sorting Ψ in descending order, the metric $\text{CP}(x, y, f, \mathcal{W})$ describes the relative position of $\psi_\text{I}(x, y, f, \mathcal{W})$ in $\text{sort}(\Psi)$. In other words, early compressed and correctly classified samples obtain a low CP close to 0, and those compressed later closer to 1.

We calculate the CP metric for all samples in CIFAR-10 during training of dense, low, and high sparsity NNs. The compression behaviour for dense NNs is measured during the pre-training phase ($e_S = 0$ and $e_E = 60$) and for sparse NNs during regularization phase ($e_S = 60$ and $e_E = e_\text{max}$).

To show, which samples are compressed first into the remaining highest weights, the 5% samples with lowest CP are visualized in Fig. 4 in the latent space of the well known *CLIP* framework [28] mapped by *t-SNE* [39]. As commonly known, the dense model compresses easy samples of all classes in the early stages [16, 20], while the low sparsity model already loses some. In the high sparsity regime no discriminative decision rules are left at beginning of training, and the remaining classes are compressed step by step as the training continues (see supplementary material, Sec. E). In our experiments, we have seen continuously that there is a bias towards the class *deer*. We call this effect *"the deer bias"*, which must be reduced with regularisation. The *deer* bias suggests that large weights in dense NNs do not encode decision rules for all classes.

To quantify the above results, Tab. 3 shows the average CP for all samples belonging to a specific class. Additionally, we split the class sets into four subsets according to their difficulty. We estimate the difficulty of a sample by counting the human label errors that are made from three human annotators derived from CIFAR-N [42], *e.g.*, 2 means that two of three persons mislabeled the sample. The first observation is that the above mentioned separation of classes is confirmed, since CP values are similar in dense NNs, but diverge in sparse NNs. In high sparsity regimes, the *deer* bias is persistent before first samples of other classes are compressed. The classes *horse* and *airplane* are only included at the end of the training. The second observation is, that within a closed set of samples belonging to a class, difficult samples are compressed later. This nature is similar to the training process of dense NNs.

Implementation details and more fine-grained results are available in the supplementary material, Sec. E.

5. Conclusion

Our work presents *Adaptive Regularized Training* (*ART*), a method that utilizes regularization to obtain sparse neural networks. The regularization is amplified continuously and used to shrink most weight magnitudes close to zero. We introduce the novel regularization loss *HyperSparse* that induces sparsity inherently while maintaining a well balanced tradeoff between exploration of new sparse topologies and exploitation of weights that remain after pruning. Extensive experiments on CIFAR and TinyImageNet show that our novel framework outperforms sparse learning competitors. *HyperSparse* is superior to standard regularization losses and leads to impressive performance gains in extremely high sparsity regimes and is much faster. Additional investigations provide new insights about the weight distribution during network compression and about patterns that are encoded in high valued weights.

Overall, this work provides new insights into sparse neural networks and helps to develop sustainable machine learning by reducing neural network complexity.

6. Acknowledgments

This work was supported by the Federal Ministry of Education and Research (BMBF), Germany under the project AI service center KISSKI (grant no. 01IS22093C), the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122), and by the Federal Ministry of the Environment, Nature Conservation, Nuclear Safety and Consumer Protection, Germany under the project GreenAutoML4FAS (grant no. 67KI32007A).

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1

[2] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3

[3] Tianlong Chen, Zhenyu Zhang, pengjun wang, Santosh Balachandra, Haoyu Ma, Zehao Wang, and Zhangyang Wang. Sparsity winning twice: Better robust generalization from more efficient training. In *International Conference on Learning Representations (ICLR)*, 2022. 2, 3

[4] Yuren Cong, Michael Ying Yang, and Rosenhahn Bodo. Reltr: Relation transformer for scene graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2023. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2, 4

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1

[7] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning (ICML)*, 2020. 2, 5, 6

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 3, 5, 6

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2015. 1, 2, 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5

[11] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research (JMLR)*, 2021. 1, 2

[12] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. What do compressed deep neural networks forget? In *arXiv:1911.05248*, 2019. 3, 8

[13] Ajay Kumar Jaiswal, Haoyu Ma, Tianlong Chen, Ying Ding, and Zhangyang Wang. Training your sparse neural network better with any mask. In *International Conference on Machine Learning (ICML)*, 2022. 2

[14] Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[15] Karolina Gintare Jonathan Frankle, Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020. 1, 2, 3, 5, 6, 7

[16] Timo Kaiser, Lukas Ehmann, Christoph Reinders, and Bodo Rosenhahn. Blind knowledge distillation for robust image classification. In *arXiv:2211.11355*, 2022. 3, 8

[17] Timo Kaiser, Christoph Reinders, and Bodo Rosenhahn. Compensation learning in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2023. 3

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 2, 4

[19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 5, 6

[20] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 8

[21] Shiwei Liu, Tim Van der Lee, Anil Yaman, Zahra Atashgahi, Davide Ferraro, Ghada Sokar, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Topological insights into sparse neural networks. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2021. 2

[22] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning (ICML)*, 2021. 2

[23] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[24] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 2018. 2

[25] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017. 2, 3

[26] Sharan Narang, Greg Diamos, Shubho Sengupta, and Erich Elsen. Exploring sparsity in recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[27] Mansheej Paul, Brett W Larsen, Surya Ganguli, Jonathan Frankle, and Gintare Karolina Dziugaite. Lottery tickets on a data diet: Finding initializations with sparse trainable networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. 8

[29] Keitaro Sakamoto and Issei Sato. Analyzing lottery ticket hypothesis from PAC-bayesian theory perspective. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3

[30] Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[31] Jonathan Schwarz, Siddhant Jayakumar, Razvan Pascanu, Peter E Latham, and Yee Teh. Powerpropagation: A sparsity inducing weight reparameterisation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 2, 5

[33] Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 5, 6, 7

[34] Mukund Varma T, Xuxi Chen, Zhenyu Zhang, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Sparse winning tickets are data-efficient image recognizers. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3

[35] Kai Sheng Tai, Taipeng Tian, and Ser-Nam Lim. Spartan: Differentiable sparsity via regularized transportation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2

[36] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[37] Bodo Rosenhahn Thomas Norrenbrock, Marco Rudolph. Take 5: Interpretable image classification with a handful of features. In *Conference on Neural Information Processing Systems, Workshop Progress and Challenges in Building Trustworthy Embodied AI (NeurIPSW)*, 2022. 1, 3

[38] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the royal statistical society series b-methodological*, 1996. 2, 3, 5

[39] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research (JMLR)*, 2008. 8

[40] Vinay Kumar Verma, Nikhil Mehta, Shijing Si, Ricardo Henao, and Lawrence Carin. Pushing the efficiency limit using structured sparse convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. 5

[41] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2, 5, 6

[42] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations (ICLR)*, 2022. 8

[43] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization, 2018. 2, 3, 5