Data-Free Model Pruning at Initialization via Expanders

James Stewart Umberto Michieli Mete Ozay
Samsung Research UK
{j2.stewart, u.michieli, m.ozay}@samsung.com

Abstract

In light of the enormous computational resources required to store and train modern deep learning models, significant research has focused on model compression. When deploying compressed networks on remote devices prior to training them, a compression scheme cannot use any training data or derived information (e.g., gradients). This leaves only the structure of the network to work with, and existing literature on how graph structure affects network performance is scarce. Recently, expander graphs have been put forward as a tool for sparsifying neural architectures. Unfortunately, however, existing models can rarely outperform a naïve random baseline. In this work, we propose a stronger model for generating expanders, which we then use to sparsify a variety of mainstream CNN architectures. We demonstrate that accuracy is an increasing function of expansion in a sparse model, and both analyse and elucidate its superior performance over alternative models.

1. Introduction

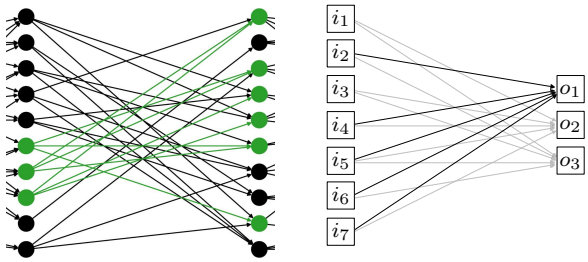
State-of-the-art results in deep learning are often achieved by highly over-parameterized models consisting of millions, and even billions, of parameters. Such models consume vast computational resources for both training and storage, making them infeasible to transmit to, store on, and train using resource constrained devices. Extensive research [1, 5, 9, 15] has therefore focused on model compression to alleviate the computational resources that they consume, whilst maintaining performance. In settings where models are transmitted to edge devices to train on local private data (*e.g.*, federated learning), a server-side compression algorithm may not use any data, which leaves the structure of the network to work with.

Early work on pruning at initialization (PaI) [21, 27, 28] proposed pruning metrics defined with respect to the gradient of the loss, and therefore depends on data. For other relevant work and background on PaI, see the survey papers [6, 29]. In settings where PaI should take place independently of both the task and data, more structural approaches are necessary. It has long been understood how certain structural properties of neural networks impact their performances [18, 20]. Outside the context of compression, the general graph structure of neural networks has also been studied [30]; however, relatively little attention has been placed on the graph structure of sparse neural networks.

In this work, we complement recent research [6, 11, 25, 27] on how sparse graph structure and connectivity can be harnessed as a tool for model compression. We investigate the role played by graph *expansion* in the performance of sparse models. We empirically demonstrate that classification accuracy is an increasing function of graph expansion, in highly sparse regimes. With this as motivation, we propose a stronger model (RReg) for constructing sparse expander layers, and observe their improved properties at the scale of typical vision architectures. We demonstrate, for a variety of architectures and datasets, that RReg consistently outperforms a random baseline [23], whereas existing models [11, 25] often do not, particularly in highly sparse regimes. Finally, we show the practical relevance of highly sparse models by exhibiting expander sparsified ResNets with fewer parameters and superior performance than their fully-connected alternatives.

2. Expander Graphs

Preliminaries To capture the graph structure of a neural network, we follow the approach of [25], which models individual layers of a neural network as bipartite graphs. Whilst more complicated models are possible, such as relational graphs [30], we believe that the straightforward approach best captures the intuition behind the relationship between graph structure and behaviour of the layers of a neural network. We achieve a sparse architecture by masking weights; that is, we associate a binary mask to each layer of the network. For a linear layer with n_0 input units and n_1 output units, $M \in \{0,1\}^{n_0 \times n_1}$ indicates the non-zero parameters of the layer. For a convolutional layer with n_0 input channels and n_1 output channels, a binary mask M indicates which input channels each entire filter is applied to. In both cases, the layer corresponds to a bipartite graph $G = (V_G^0, V_G^1, E_G)$, with $V_G^0 = \{1, \dots, n_0\}$, $V_G^1 = \{1, \dots, n_1\}$, and



(a) **Linear** layer: graph expansion allows subsets of input neurons to interact with larger subsets of output ones (highlighted in green).

(b) **Conv** layer: a filter is applied to input channels $\{i_j\}_{j=2,4,5,6,7}$, to obtain output channel o_1, as determined by the graph structure.

Figure 1. Illustration of bipartite expander graph representations of sparse layers.

$E_G = \{(i,j) \mid i \in V_G^0, j \in V_G^1, M_{i,j} = 1\}$, where $M_{i,j}$ is the element in the i^{th} row and j^{th} column of M.

Definition 1 (α-expander)**.** *Let $d \geq 3$ (degree) be an integer and let $\alpha > 0$ be a real number. We say that a d-regular bipartite graph $G = (V_G^0, V_G^1, E_G)$ is an α-expander if, for all $i \in \{0,1\}$ and all $S \subseteq V_G^i$ with $|S| \leq |V_G^i|/2$, we have that $|\partial S| \geq \alpha |S|$, where ∂S denotes the set of vertices connected to S.*

The above definition already illustrates why expander graphs are of interest as a tool for data-free model compression – expanders are simultaneously sparse yet highly connected. When modelling a layer of a neural network as a bipartite expander, greater values of α allow subsets of neurons to interact with a larger subset of other neurons, therefore promoting a better feature shareability and flow of information through the network, as depicted in Fig. 1.

We may also give a spectral definition of the class of d-regular expanders. For an n-vertex d-regular bipartite graph, let $\lambda_1(G) \geq \lambda_2(G) \geq \cdots \geq \lambda_n(G)$ denote the eigenvalues of its adjacency matrix, and define $\lambda(G) = \max_{i \neq 1} |\lambda_i(G)|$.

Definition 2 (ϵ-expander)**.** *Let $d \geq 3$ be an integer and let $\epsilon > 0$ be a real number. We say that an n-vertex d-regular bipartite graph is an ϵ-expander if it satisfies $\lambda(G) \leq \epsilon d$.*

We can define a class of expanders using either of Def. 1 and 2, and the ways in which they relate are well-understood [13]. The spectral definition of an expander allows us to conveniently compare the expansion properties of different graph classes, and understand the best possible expansion properties that a class of graphs can have. It is known [24] that for every ϵ and d, there exists n such that all d-regular graphs G with at least n vertices satisfy $\lambda(G) > 2\sqrt{d-1} - \epsilon$. A d-regular graph G is called a *Ramanujan graph* if it satisfies $\lambda(G) \leq 2\sqrt{d-1}$, and such graphs therefore give the best possible expanders [13]. We

Table 1. Comparison between random models at degree d.

Model	**Regularity**	α	**Edges**
Random [4, 23]	random	low	$d \cdot n/2$
X-Net [25]	d-left-regular	medium	$d \cdot n/2$
RReg (ours)	d-regular	high	$d \cdot n/2$

are interested in constructing graphs with the strongest possible expansion properties, so that we can employ them as effective sparsifiers for the layers of a neural network. We analyze two existing models from the literature: the random left-regular model [25] (X-Net) and the random graph model [4, 23] (Random). We then propose a stronger model based on the **R**andom **Reg**ular graph model (RReg).

Methodology The RReg model, for fixed n and d, chooses a random n-vertex d-regular graph uniformly at random from the set of all such graphs (Tab. 1). The X-Net model [25], for fixed n and d, chooses a random n-vertex d-left-regular (every left vertex has degree d) graph uniformly at random from the set of all such graphs. Finally, the random model [23], for fixed n and d, connects every pair of vertices with an edge independently with probability $2d/n$.

With high probability, RReg graphs are *essentially* Ramanujan, and achieve optimal expansion [2, 24]. These optimal expansion properties provide our motivation for their use, and we demonstrate empirically that the RReg model produces stronger expanders than both X-Net and Random.

The random baseline [23] is crucially important, and has so far been overlooked by existing work [11, 25]. In graph terms, it can be viewed as a version of the Erdős-Rényi random graph model [4], which is known to produce poor expanders. In [23], it is shown that, in certain sparsity regimes, randomly pruned architectures can match the performance of their fully-connected equivalents. Another recent work [11] develops metrics to demonstrate that existing work on expander-based PaI often cannot be separated from a random baseline. Whilst they do not propose a model for overcoming this, we do so by equipping our random graph model with a simple procedure for sampling vertex subsets, in order to estimate α and thereby allow us optimize the expansion properties of our generated graphs

We note that theoretical results on expanders apply *asymptotically* – the properties of random graphs and the probabilities they occur with are, strictly speaking, only valid for graphs of *large enough* size. Despite this, we empirically verify that the expansion properties of the various random graph models that we appeal to, do indeed align with their theoretical properties. Further, the current state of the art is achieved every year by deeper and wider architectures [16, 32].

3. Results

3.1. Experimental Setup

Datasets We consider popular benchmarks under variable dataset sizes and numbers of classes. We use: CIFAR [17] with 10 or 100 classes, containing $50k$ training and $10k$ testing samples; Tiny ImageNet [19] with 200 classes, containing $100k$ training and $10k$ testing samples; ImageNet [3] with $1k$ classes, containing approximately $1.3M$ training and $100k$ testing samples. We use the same batch size for each model/dataset pair – the largest batch size compatible with a GeForce RTX 2080 Ti GPU.

Metrics We evaluate performance along several metrics. Methods are ranked according to top-1 accuracy, Acc (%, ↑). Relative gain of Acc_2 against a baseline Acc_1 is measured in terms of (i) relative gain $\Delta_{\text{R}} = (\text{Acc}_2 - \text{Acc}_1)/\text{Acc}_1$; and (ii) room-aware relative gain $\Delta_{\text{RAR}} = (\text{Acc}_2 - \text{Acc}_1)/(100 - \text{Acc}_1)$. Sparsity is measured by the number of remaining parameters P (M, ↓) or by their percentage, PP (%, ↓) at a certain degree d (↓). Network connectivity is measured via combinatorial expansion α (↑). Model calibration, *i.e.*, the ability to predict probability estimates representative of the true correctness likelihood, is measured through the ECE score, as in [7].

Sparse graph generation We follow the masking approach described in Sec. 2. We implement sparse linear and convolutional layers in PyTorch, endowed with a mask that defines the sparsity pattern for the relevant layer. This mask is applied to to all forward and backward passes through the layer. Upon initialisation of the sparse model, we generate the mask for each layer by taking the biadjacency matrix of a NetworkX [8] graph object, which can be constructed according to any desired model.

The theoretical motivations described in Sec. 2 largely concern regular bipartite graphs, which are necessarily balanced (have the same number of left and right-hand vertices). In practice this is not always the case; however, in spite of this minor deviation from the theoretical setup, we empirically observe the expansion properties that we expect to see in the graphs generated by each of the models that we consider, as we show in Sec. 3.2.

3.2. Experimental Results

Empirical motivation We show in Fig. 2 that expander-based PaI improves accuracy, by sparsifying a network according to graphs with varying expansion, but same average degree d (and thus the same number of edges). This controlled experiment is conducted on CIFAR10 using a small 2-layer CNN (with 6 and 16 channels, respectively), where the penultimate linear layer is sparsified via 3-regular graphs with increasingly strong expansion properties.

Variable sparsity We evaluate on CIFAR10/100 in Tab. 2 for a VGG16 backbone [26]. Each horizontal block corresponds to a different level of sparsity: from top to bottom, the percentage of remaining parameters PP diminishes with the layer degree d. The mean standard deviation over all models over five runs is ~ 0.2. As shown in [11], we

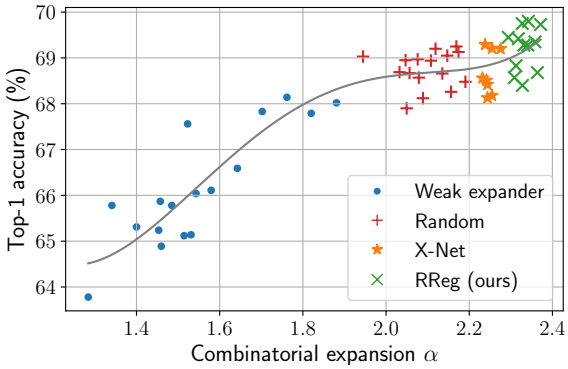


Figure 2. Accuracy is an increasing function of expansion for a small CNN on CIFAR10. A best fit line is shown in black. Degree $d = 3$ is constant across all sparse models.

Table 2. Results on CIFAR10/100 using VGG16 at variable sparsity levels. The original VGG model has $15M$ parameters. Δ_{R} and Δ_{RAR} are computed with respect to Random.

			CIFAR10			CIFAR100		
Method	PP	d	Acc	Δ_{R}	Δ_{RAR}	Acc	Δ_{R}	Δ_{RAR}
Original	100	-	94.24	-	-	74.16	-	-
Random [23]	13.79 ±0.11	60	**93.50**	-	-	**72.97**	-	-
X-Net [25]			93.46	-0.04	-0.62	72.73	-0.33	-0.89
RReg (ours)			**93.50**	+0.00	+0.61	72.72	-0.34	-0.04
Random [23]	7.07 ±0.20	30	92.51	-	-	69.81	-	-
X-Net [25]			92.65	+0.15	+1.87	69.80	-0.01	-0.03
RReg (ours)			**93.05**	+0.58	+5.44	**70.15**	+0.49	+1.16
Random [23]	3.62 ±0.15	15	91.30	-	-	66.58	-	-
X-Net [25]			91.38	+0.09	+0.92	66.81	+0.35	+0.69
RReg (ours)			**91.50**	+0.22	+1.39	**67.72**	+1.71	+2.74
Random [23]	0.79 ±0.03	3	85.81	-	-	56.98	-	-
X-Net [25]			86.06	+0.29	+1.76	56.69	-0.51	-0.67
RReg (ours)			**87.02**	+1.41	+6.89	**59.61**	+4.62	+6.74

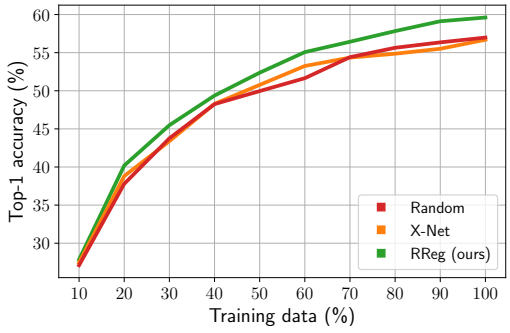


Figure 3. CIFAR100 Acc of a sparsified VGG16 with $d = 3$ over variable training data percentage.

Table 3. Results on Tiny-ImageNet with several architectures at a fixed degree $d = 3$. Δ_{R} is computed with respect to Random.

Method	VGG16		MN		RN18		RN34		RN50		RN101		RN152		WRN28-10		WRN40-14		Avg	
	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	PP [%]	Acc	Acc	Δ_{R}
Original	100	40.03	100	54.75	100	53.88	100	56.95	100	57.08	100	60.13	100	61.29	100	46.27	100	49.04	53.27	-
Random [23]	0.79 (±0.03)	22.84	19.33 (±0.10)	21.81	3.45 (±0.05)	44.02	2.33 (±0.02)	**47.34**	14.24 (±0.06)	46.77	8.52 (±0.08)	49.56	6.62 (±0.08)	49.05	1.04 (±0.02)	35.5	0.65 (±0.01)	39.51	39.63	-
X-Net [25]		23.20		19.49		42.69		46.97		45.36		49.45		49.47		35.57		40.21	39.16	-1.20
RReg (ours)		**25.31**		**34.26**		**44.30**		46.30		**48.27**		**51.04**		**51.21**		**36.50**		**40.54**	**41.94**	+5.11

Table 4. Results of RN50 on ImageNet at a fixed degree $d = 3$.

Method	PP	Acc	Δ_{R}
Original	100	68.69	-
Random [23]	14.24 (±0.06)	51.48	-
X-Net [25]		52.63	+2.23
RReg (ours)		**54.26**	+5.40

Table 5. Sparse architectures improve smaller networks (Tiny-ImageNet). Note: batch size is $\sim$ 2x higher than in Tab. 3.

Type	Model	P	Acc	Δ_{R}	Δ_{RAR}	Model	P	Acc	Δ_{R}	Δ_{RAR}
Orig	RN50	23.9	59.36	-	-	WRN28-4	5.9	43.16	-	-
RReg	RN152	21.9	62.01	+4.46	+6.52	WRN28-10	5.9	43.76	+1.39	+1.06

verify that is hard to separate existing methods (*e.g.*, X-Net) from a random baseline. Nevertheless, our method outperforms both the random baseline and X-Net almost everywhere (with one exception on CIFAR100). Looking at Δ_R and Δ_{RAR}, we observe that our method is most effective at high levels of sparsity (lower blocks).

Fig. 3 shows that our approach is consistently higher at variable percentage of training samples, therefore revealing its usefulness, even in low data regimes.

Additionally, we argue that expander sparsified architectures achieve higher confidence calibration [7] than fully-connected ones, as recently showed for post-training pruning techniques [22]. The fully-connected model has an ECE score of 0.044 and 0.141 on CIFAR10 and CIFAR100, respectively. On the other hand, RReg at high level of sparsity ($d = 3$) has an ECE score of 0.037 and 0.075, respectively. ECE reduction reflects a higher calibration, which gives the model higher correctness likelihood and interpretability.

More backbones, samples, and classes We evaluate on Tiny-ImageNet and ImageNet at a high level of sparsity ($d = 3$). To prove the robustness of our approach to models with variable design choice, depth and width, we evaluate popular CNNs on Tiny-ImageNet in Tab. 3: VGG16 [26], MobileNet (MN) [14], ResNets (RN) [10] and Wide-ResNets (WRN) [31]. Our method consistently outperforms both Random and X-Net by over 5% relative gain.

Tab. 4 reports RN50 on ImageNet, where the relative gain is much higher than in previous cases – the higher task difficulty appears to increase the influence of network structure. Overall, our RReg proves to be applicable to multiple backbones and to datasets with varying numbers of samples (from 3k to $\sim$ 1.2M) and classes (from 10 to 1k).

Why do we need expander-sparsified architectures? PaI presents similar challenges to those encountered when pruning post-training. Sparse architectures (pruned either before or after training) are yet to fully reveal their advantages, mainly due to limitations of both hardware and libraries which hinder large-scale deployment of pruned models [6, 12, 29]. This leaves several exciting possible applications of PaI; for example, as a tool for neural architecture search that provides sparse networks from the outset. Nevertheless, we argue that our RReg PaI method reveals to be effective in designing sparse architectures that achieve higher accuracy at a same number of parameters than their shallower and narrower fully-connected counterparts. We investigate this claim in Tab. 5 for RN and a WRN-based architectures. An RN152 sparsified according to RReg significantly outperforms a fully-connected (shallower) RN50 by $\Delta_{\text{RAR}} = 6.52$. Interestingly, the original RN50 contains around 10% more parameters than the sparse RN152. Similar results also hold for the WRN.

4. Conclusions

In this work, we explored the use of expander graphs as sparsifiers for neural network architectures. We proposed a model (RReg) for generating sparse layers with optimal expansion properties. We have further identified the settings in which expansion impacts network performance, and captured this relationship by observing that classification accuracy is an increasing function of graph expansion. Our results show a consistent improvement over existing random graph models for data-free PaI (*e.g.* XNet, Random), for a range of image classification tasks and architectures. We believe that our work better explains the role that expansion plays, and gives compelling motivation for future research in this area. For example, application to larger-scale architectures (*e.g.* transformers, LLMs) and domains, proving theoretical guarantees around the performance of expander sparsified architectures, and exploring the graph structure of neural networks in general.

References

[1] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of Machine Learning and Systems*, 2:129–146, 2020. 1

[2] Gerandy Brito, Ioana Dumitriu, and Kameron Decker Harris. Spectral gap in random bipartite biregular graphs and applications. *Combinatorics, Probability and Computing*, 31(2):229–267, 2022. 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009. 3

[4] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1):17–60, 1960. 2

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1

[6] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 1, 4

[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1321–1330. PMLR, 2017. 3, 4

[8] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 3

[9] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 4

[11] Duc N.M Hoang, Shiwei Liu, Radu Marculescu, and Zhangyang Wang. Revisiting pruning at initialization through the lens of Ramanujan graph. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. 1, 2, 3

[12] 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. *The Journal of Machine Learning Research*, 22(1):10882–11005, 2021. 4

[13] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006. 2

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4

[15] Steven A Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989. 1

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 2

[17] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, Univ. Toronto, 2009. 3

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[19] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 3

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[21] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1

[22] Ruofeng Li. Calibration analysis of structured pruning methods and cascade classifier design based on pruned networks. 2022. 4

[23] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 1, 2, 3, 4

[24] Alon Nilli. On the second eigenvalue of a graph. *Discrete Mathematics*, 91(2):207–210, 1991. 2

[25] Ameya Prabhu, Girish Varma, and Anoop Namboodiri. Deep expander networks: Efficient deep networks from graph theory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–35, 2018. 1, 2, 3, 4

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 3, 4

[27] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6377–6389, 2020. 1

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

[29] Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. Recent advances on neural network pruning at initialization.

In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 23–29, 2022. 1, 4

[30] Jiaxuan You, Jure Leskovec, Kaiming He, and Saining Xie. Graph structure of neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10881–10891. PMLR, 2020. 1

[31] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2016. 4

[32] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12104–12113, 2022. 2