APP: Anytime Progressive Pruning

Diganta Misra [*1 2 3] **Bharat Runwal** [*2 4] **Tianlong Chen** [5] **Zhangyang Wang** [5] **Irina Rish** [1]

Abstract

With the latest advances in deep learning, several methods have been investigated for optimal learning settings in scenarios where the data stream is continuous over time. However, training sparse networks in such settings has often been overlooked. In this paper, we explore the problem of training a neural network with a target sparsity in a particular case of online learning: the anytime learning at macroscale paradigm (ALMA). We propose a novel way of progressive pruning, referred to as *Anytime Progressive Pruning* (APP); the proposed approach significantly outperforms the baseline dense and Anytime OSP models across multiple architectures and datasets under short, moderate, and long-sequence training. Our method, for example, shows an improvement in accuracy of $\approx$ 7% and a reduction in the generalisation gap by $\approx$ 22%, while being $\approx 1/3$ rd the size of the dense baseline model in few-shot restricted imagenet training. The code and experiment dashboards can be accessed at `https://github.com/landskape-ai/Progressive-Pruning` and `https://wandb.ai/landskape/APP`, respectively.

1. Introduction

Supervised learning has been one of the most well-studied learning frameworks for deep neural networks, where the learner is provided with a dataset $\mathcal{D}_{x,y}$ of samples(x) and corresponding labels(y); and the learner is expected to predict the label y by learning on x usually by estimating $p(y|x)$. In an offline learning environment (Ben-David et al., 1997), the learner has access to the complete dataset $\mathcal{D}_{x,y}$, while in a standard online learning setting (Sahoo et al., 2017; Bottou et al., 1998) the data arrive in a stream over time, assuming that the rate at which samples arrive is the same as that of the learner's processing time to learn from them. There are several fine-grained types of learning from a stream of data, including, but not limited to, continual learning (Van de Ven & Tolias, 2019; Thrun, 1995; Ring, 1998), active online learning (Baram et al., 2004; Settles, 2009), and anytime learning (Grefenstette & Ramsey, 1992; Ramsey & Grefenstette, 1994). In an anytime learning framework, the learner has to have good performance at any point in time, while gradually improving its performance over time upon observing new data that subsequently arrive.

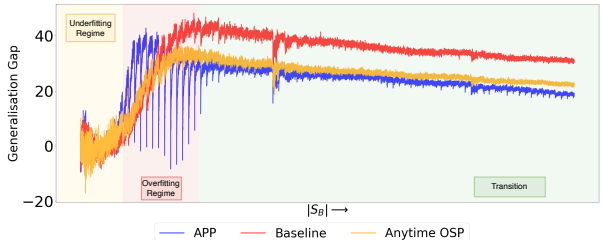

Figure 1: **Non-Monotonic Transition:** generalisation gap as a function of the number of megabatches ($|S_B| = 100$) in the long-sequence ALMA setting with full replay for ResNet-50 backbone on the CIFAR-10 dataset.

In this work, we are interested in exploring the training of sparse neural networks (pruned) in the ALMA setting (Caccia et al., 2021). Pruning (Blalock et al., 2020; Luo et al., 2017; Wang et al., 2021) of over-parameterized deep neural networks has been studied for a long time. Pruning deep neural networks leads to a reduction in inference time and memory footprint. Although early pruning work focused exclusively on pruning weights after pre-training the dense model for a certain number of iterations, extensive research has recently been conducted on pruning the model at initialization, that is, finding the lottery ticket (Frankle & Carbin, 2018; Frankle et al., 2019a;b; Malach et al., 2020) from a dense model at the start without pre-training the dense model (Lee et al., 2018; Wang et al., 2020a). However, few studies (Chen et al., 2020) have investigated training of sparse neural networks (pruned) in online settings. Thus, our objective is to answer the following question:

"*Given a dense neural network and a target sparsity, what*

[*]Equal contribution [1]Mila - Quebec AI Institute [2]Landskape AI [3]Morgan Stanley [4]IIT-Delhi [5]VITA,UT-Austin. Correspondence to: Diganta Misra <diganta@landskape.ai>, Bharat Runwal <bharat@landskape.ai>.

DyNN workshop at the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

should be the optimal way of pruning the model in ALMA setting?"

In summary, our contributions can be summarized by the following two points.

- We provide the first comprehensive study of deep neural network pruning in the ALMA setting; henceforth to this extent, we propose a novel approach of progressive pruning which we term **Anytime Progressive Pruning**(APP).
- We further investigate the APP training dynamics compared to baselines in the ALMA setting with a varied number of megabatches using C-10, C-100[1], and Restricted ImageNet datasets.

2. Related Work

2.1. Pruning

Pruning (LeCun et al., 1990; Han et al., 2015a) as one of the effective model compression techniques is widely explored in the field of efficient machine learning. It trims down the parameter redundancy in modern over-parameterized deep neural networks, aiming at substantial resource savings and unimpaired performance. Depending on the granularity of the removed network components, classical pruning methods can be categorized into unstructured (Han et al., 2015a; LeCun et al., 1990; Han et al., 2015b) and structural pruning (Liu et al., 2017; Zhou et al., 2016), where the former removes parameters irregularly and the latter discards substructures such as convolution filters or layers. In addition to the above post-training pruning, it can also be flexibly applied before network training, such as SNIP (Lee et al., 2019), GraSP (Wang et al., 2020b) and SynFlow (Tanaka et al., 2020) or during training (Zhang et al., 2018; He et al., 2017).

Recent closely related work (Chen et al., 2021) defines pruning in sequential learning as a dynamical system and proposes two effective lifelong pruning algorithms to identify high-quality subnets, leading to superior trade-offs between efficiency and lifelong learning performance. Furthermore, (Golkar et al., 2019) prunes neurons with low activity and (Sokar et al., 2020) compresses the sparse connections of each task during training to overcome the problem of forgetting.

2.2. Revisiting ALMA

In this section, we revisit the ALMA learning framework as conceptualized in (Caccia et al., 2021). In ALMA, the model f_θ is provided with a stream of S_B of $|S_B|$ consecutive batches of samples under the assumption that there exists an underlying data distribution $\mathcal{D}_{x,y}$ with input $x \in \mathbb{R}^d$ and target labels $y \in \{1, ..., C\}$. Each megabatch $\mathcal{M}_t$ consists of $N \gg 0$ i.i.d. samples randomly drawn from $\mathcal{D}_{x,y}$, for $t \in \{1, ..., S_B\}$. Therefore, the stream S_B is the ordered sequence $S_B = \{\mathcal{M}_1, ..., \mathcal{M}_{|S_B|}\}$ where $|S_B|$ represents the total number of megabatches in the stream. Thus, the model $f_\theta : \mathbb{R}^d \to \{1, ..., C\}$ is trained by processing a *mini-batch* of $n \ll N$ samples at a specified time of each megabatch $\mathcal{M}_t$ and iterating multiple times over each mega-batch before having access to the next mega-batch. In ALMA, it is assumed that the rate at which megabatches arrive is slower than the training time of the model on each megabatch and, therefore, the model can iterate over the megabatches at its disposal based on its discretion to maximize performance.

In ALMA, the authors conducted a conclusive study using different baselines, two of them being (a) ensemble and (b) dynamic growing. In both cases, the complexity of the model parameters was gradually increased to allocate sufficient capacity to accommodate the newly arrived megabatches. However, in this paper, we investigate the effect of progressively decreasing the parametric complexity of the model via pruning and subsequently training a sparse neural network in an ALMA setting.

3. Anytime Progressive Pruning

In this section, we formally introduce our proposed method *Anytime Progressive Pruning*(APP) as defined in the algorithm 1. For each megabatch $\mathcal{M}_t \in S_B$, we construct the replay inclusive megabatch $\mathcal{M}_t$ by taking the union of all previous megabatches along with the current megabatch and then create a small sample set π_t of size $0.2 * |\mathcal{M}_t|$ to be used to prune the model to $0.8^{\delta_t} \times 100\%$ sparsity. Here, δ_t is obtained from a predetermined list δ of uniformly spaced values that denote the target sparsity levels for each megabatch in the stream S_B. After pruning the model, we train it on the $\mathcal{M}_t$ megabatch and evaluate it on a held-out test set.

Algorithm 1 Training APP in the ALMA setting

Require: $f_\theta^{t=0}, \tau, S_B \Longleftrightarrow \{\mathcal{M}_1, ..., \mathcal{M}_{|S_B|}\}$
$t \leftarrow 1$
$\delta \leftarrow \{start = 1, end = \tau, steps = |S_B|\}$
while $t \leq |S_B|$ **do**
  SNIP set(π_t) $\leftarrow \emptyset$
  $\mathcal{M}_t \leftarrow \bigcup_{i=1}^{t} \mathcal{M}_i$
  pruning state $\leftarrow 0.8^{\delta_t}$
  SNIP set $\leftarrow \pi_t \subset \mathcal{M}_t \mid \frac{|\pi_t|}{|\mathcal{M}_t|} = 0.2$
  $f_\theta^t \leftarrow$ SNIP(f_θ^{t-1}, SNIP set, pruning state)
  f_θ^t.*train*($\mathcal{M}_t$)
end while

To evaluate APP, we use primarily 2 baselines:

[1]C-10 and C-100 denote CIFAR-10 and 100 respectively.

1. **Baseline**: This denotes the model at full parametric capacity trained and fine-tuned on all megabatches in the stream S_B using stochastic gradient descent in an ALMA setting.

2. **Anytime OSP**: This denotes one-shot pruning (OSP) to the target sparsity $0.8^{\tau} \times 100\%$ at the initialization of f_θ and then subsequently training on all mega-batches in the stream S_B in an ALMA setting. Thus, Anytime OSP models have the lowest parametric complexity since the start of training on the first megabatch in the stream S_B. We use the same pruner of choice (SNIP) by default for both APP and Anytime OSP. Similarly to APP, we prune the model at initialization using a small randomly selected subset π_1 of the first megabatch $\mathcal{M}_1$ of size $0.2 * |\mathcal{M}_1|$.

∗ **Cumulative Error Rate (CER)**: Along with test accuracy, we use CER to evaluate the methods described above, which can be defined by the following equation.

$$CER = \sum_{t=1}^{S_B} \sum_{j=1}^{|T_{x,y}|} \mathbb{1}(\mathcal{F}_t(x_j) \neq y_j) \qquad (1)$$

Here, $T_{x,y}$ represents the held-out test set used for evaluation, $\mathcal{F}_t$ represents the trained model at the t -th megabatch, and $\mathcal{F}_t(x_j)$ represents the prediction on the j-th index sample of the test set $T_{x,y}$ compared to the true label for that sample y_j. CER provides strong information on whether the learner is a good anytime learner, as it is expected to minimize CER at each megabatch training in the stream S_B.

In addition, we also note the generalisation gap as the difference between the training and the validation accuracy. This gives a notion of whether the model is over- or under-fitting.

3.1. Results

3.1.1. Short sequence ALMA ($|S_B| = 8$)

We start by analyzing the results shown in Fig. 2. Each megabatch M_t consists of 6250 samples and the target sparsity was set to $\tau = 4.5$.

For all models, we observed a strong performance improvement for APP compared to baseline and Anytime OSP in all metrics: test accuracy, CER, and generalisation gap. For example, with ResNet-50 (R50) in C-100, APP improved the test accuracy by 17.97% and 11.12%, reduced CER by 9927 and 5533 and decreased the generalisation gap by 20.49% and 14.79% compared to baseline and Anytime OSP. For C-10, we use a noncyclic step decay learning rate policy which reduces the learning rate only for the first megabatch ($\mathcal{M}_1$) and subsequently stays constant for all remaining megabatches. However, for C-100, we used a cyclic step decay learning rate policy, where the learning rate resets to it's initial value when starting on a new megabatch. In Fig. 2, we show the results for using magnitude and random pruning instead of SNIP for APP and, based on the observations, we make SNIP the default pruner of choice due to its stability and strong performance.

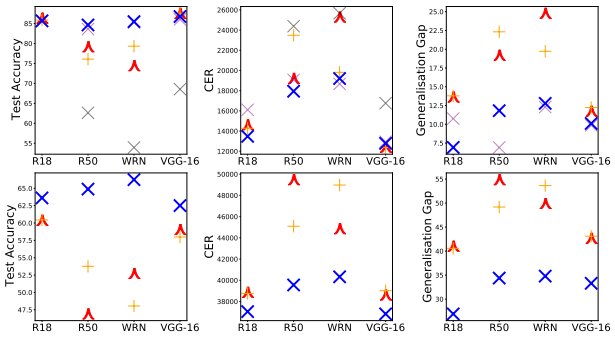

Figure 2: Top and Bottom Rows: L→R Test Accuracy(↑), CER(↓) and Generalisation Gap(↓) results for C-10 and C-100, respectively. 人, +, ×, × and × represent the baseline, Anytime OSP, APP (Snip), APP (Magnitude), and APP (Random), respectively.

3.1.2. Moderate and long sequence ALMA ($|S_B| = 25, 50, 100$)

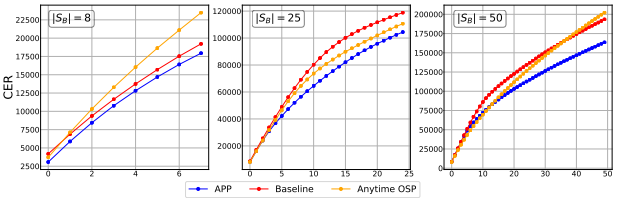

Figure 3: Change in CER during training of APP, Anytime OSP and Baseline using a ResNet-50 on C-10 as a function of megabatches($|S_B|$).

Similarly to short-sequence-based ALMA, we observed a strong improvement in performance while using APP compared to the Anytime OSP and baseline models. In particular, when $|S_B| = 100$, where each megabatch has $|\mathcal{M}_t| = 500$ samples, we report an improvement in CER by 105277 compared to the baseline model, which is equivalent to APP correctly classifying the test set $T_{x,y}$ of 10,000 samples 10 times compared to the baseline model throughout the training process on the complete stream $|S_B|$. Interestingly, as reported in Table 1, we find that the performance of the baseline has a high variation caused by the change in $|S_B|$ with a deviation in test accuracy of $\sigma = 4.95\%$, while APP is extremely stable and is less sensitive to the change in $|S_B|$ with a deviation in test accuracy of $\sigma = 1.15\%$ across

Table 1: Results on C-10 ALMA with varying $|S_B|$.

Method	$\|S_B\|$	$\|\mathcal{M}_t\|$	Test Accuracy(↑)	CER (↓)	Generalisation Gap(↓)
Baseline	25	2000	**82.69%**	118876	9.98%
Anytime OSP	25	2000	78.86%(-3.83 %)	110698 (-8178)	16.28%(+6.30%)
APP	25	2000	79.73%(-2.96 %)	**104435** (-14441)	**2.92%**(-7.06 %)
Baseline	50	1000	79.13%	193384	20.97%
Anytime OSP	50	1000	72.91%(-6.22 %)	202212 (+8828)	26.56%(-2.41 %)
APP	50	1000	**82.0%**(+2.87 %)	**163503** (-29881)	**14.71%**(-6.26%)
Baseline	100	500	70.87%	396572	28.97%
Anytime OSP	100	500	78.51%(+7.64 %)	315349 (-81223)	20.13%(-8.84 %)
APP	100	500	**82.32%**(+11.45 %)	**291295** (-105277)	**16.50%**(-12.47%)

$|S_B|$ values of 25, 50 and 100. We observe the change in CER in Fig. 3, which shows that APP maintains a lower CER throughout training compared to its counterparts while varying the sequence length $|S_B|$.

3.1.3. Few shot experiments on Restricted ImageNet

Table 2: Results on Few-shot Restricted ImageNet ALMA.

Method	$\|\mathcal{M}_t\|$	$\|S_B\|$	α	Test Accuracy(↑)	CER(↓)	Generalisation Gap(↓)
Baseline	756	10	540	43.36%	25328	17.39%
Anytime OSP	-	-	-	**47.25%**(+3.89 %)	24978 (-350)	21.53%(+4.13%)
APP	-	-	-	40.40%(-2.96%)	**24712** (-616)	**6.96%**(-10.43 %)
Baseline	126	30	270	40.81%	75128	55.50%
Anytime OSP	-	-	-	**44.55%**(+3.74 %)	76871 (+1743)	48.53%(-6.97%)
APP	-	-	-	44.11%(+3.23%)	**73206** (-1922)	**34.42%**(-21.08 %)
Baseline	252	30	540	48.03%	68832	48.73%
Anytime OSP	-	-	-	50.23%(+2.2 %)	68765 (-67)	45.29%(-3.44%)
APP	-	-	-	**55.04%**(+7.01 %)	**66239** (-2593)	**26.39%**(-22.34 %)
Baseline	54	70	270	47.88%	159204	45.03%
Anytime OSP	-	-	-	**51.45%**(+3.57 %)	**158608** (-596)	45.36%(+0.33 %)
APP	-	-	-	48.90%(+1.02 %)	162360 (+3156)	**30.74%**(-14.29 %)
Baseline	108	70	540	61.39%	140069	34.46%
Anytime OSP	-	-	-	61.39%(0%)	**139152** (-917)	32.98%(-1.48 %)
APP	-	-	-	**62.49%**(+1.10 %)	139963 (-106)	**17.59%**(-16.87%)

In this section, we investigate the performance of APP compared to Anytime OSP and the baseline models on Restricted Balanced ImageNet (Engstrom et al., 2019; Tsipras et al., 2018) using various few-shot learning settings. We primarily conduct experiments using the following two few-shot settings.

1. $\alpha = 270$: For this, we only keep 270 samples per class, which totals to 3780 samples for the entire data set. We tested this for $|S_B| = 30, 70$ where $|\mathcal{M}_t| = 126, 54$ samples, respectively.

2. $\alpha = 540$: For this, we keep only 540 samples per class, which totals 7560 samples for the complete dataset. We tested this for $|S_B| = 10, 30, 70$ where $|\mathcal{M}_t| = 756, 252, 108$ samples, respectively.

As reported in Table 2, we observe that APP significantly reduces the generalisation gap for each model variant compared to the Anytime OSP and the baseline counterparts. Excluding the experiment of $\alpha = 270$, $|S_B| = 70$, we observed a decrease in CER compared to the baseline model.

For all models, APP significantly reduces the generalisation gap and also improves test accuracy, except in the case of the experiment $|S_B| = 10$. Similar to Sect. 3.1.2, the target sparsity was kept fixed at $\tau = 4.5$ and the backbone used throughout was R-50.

3.2. Transitions in generalisation gap

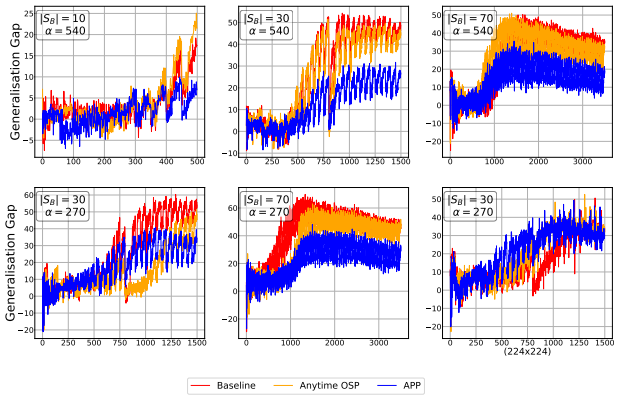

Figure 4: Generalisation gap curves during training of APP, Anytime OSP and Baseline for the results in Table 2.

We visualize the generalisation gap as a function of training iterations across the megabatches in the stream S_B in Fig.4 for the experiments reported in Table 2. As demonstrated in Fig. 1, we observe the same non-monotonic transition in the high number of megabatch $|S_B| = 30, 70$ settings where the model initially oscillates within the under-fitting phase and then continues into a critical over-fitting regime before undergoing a smooth continuous transition where the generalisation gap steadily decreases.

In all subplots, it can be seen that APP consistently maintains a lower generalisation gap compared to its Anytime OSP and baseline counterparts.

4. Conclusion

In this work, we introduced Anytime Progressive Pruning (APP), a novel way to progressively prune deep networks while training in an ALMA regime. We improvise on existing pruning at initialization strategy to design APP and perform an extensive empirical evaluation to validate performance improvement in various architectures and datasets. We found that progressively pruning deep networks with APP while training in an ALMA setting causes a significant drop in the generalisation gap compared to one-shot pruning methods and the dense baseline model.

Future work includes analyzing the non-monotonic transitions observed in the generalisation gap and extending the progressive pruning framework to other downstream tasks.

References

Baram, Y., Yaniv, R. E., and Luz, K. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291, 2004.

Ben-David, S., Kushilevitz, E., and Mansour, Y. Online learning versus offline learning. *Machine Learning*, 29 (1):45–63, 1997.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Guttag, J. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Bottou, L. et al. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Caccia, L., Xu, J., Ott, M., Ranzato, M., and Denoyer, L. On anytime learning at macroscale. *arXiv preprint arXiv:2106.09563*, 2021.

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2020.

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LXMSvPmsm0g.

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

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

Frankle, J., Dziugaite, G. K., and Daniel, M. Roy, and michael carbin. the lottery ticket hypothesis at scale. *arXiv preprint arXiv:1903.01611*, 2(3), 2019a.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019b.

Golkar, S., Kagan, M., and Cho, K. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019.

Grefenstette, J. J. and Ramsey, C. L. An approach to anytime learning. In *Machine Learning Proceedings 1992*, pp. 189–195. Elsevier, 1992.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015b.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

Lee, N., Ajanthan, T., and Torr, P. H. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Lee, N., Ajanthan, T., and Torr, P. H. S. Snip: Single-shot network pruning based on connection sensitivity, 2019.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Ramsey, C. L. and Grefenstette, J. J. Case-based anytime learning. In *Case Based Reasoning: Papers from the 1994 Workshop*, pp. 91–95. AAAI Press Menlo Park, California, 1994.

Ring, M. B. Child: A first step towards continual learning. In *Learning to learn*, pp. 261–292. Springer, 1998.

Sahoo, D., Pham, Q., Lu, J., and Hoi, S. C. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017.

Settles, B. Active learning literature survey. 2009.

Sokar, G., Mocanu, D. C., and Pechenizkiy, M. Spacenet: Make free space for continual learning. *arXiv preprint arXiv:2007.07617*, 2020.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Thrun, S. A lifelong learning perspective for mobile robot control. In *Intelligent robots and systems*, pp. 201–214. Elsevier, 1995.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Van de Ven, G. M. and Tolias, A. S. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Wang, C., Wang, C., Zhang, G., and Grosse, R. B. Picking winning tickets before training by preserving gradient flow. *ArXiv*, abs/2002.07376, 2020a.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020b.

Wang, H., Qin, C., Zhang, Y., and Fu, Y. Emerging paradigms of neural network pruning. *arXiv preprint arXiv:2103.06460*, 2021.

Zhang, T., Zhang, K., Ye, S., Tang, J., Wen, W., Lin, X., Fardad, M., and Wang, Y. Adam-admm: A unified, systematic framework of structured weight pruning for dnns. *arXiv preprint arXiv:1807.11091*, 2018.

Zhou, H., Alvarez, J. M., and Porikli, F. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pp. 662–677. Springer, 2016.