APP: Anytime Progressive Pruning

Diganta Misra DIGANTA@LANDSKAPE.AI
Landskape AI, Mila, UdeM, Morgan Stanley

Bharat Runwal BHARAT@LANDSKAPE.AI
Landskape AI

Tianlong Chen TIANLONG.CHEN@UTEXAS.EDU
VITA, UT-Austin

Zhangyang Wang ATLASWANG@UTEXAS.EDU
VITA, UT-Austin

Irina Rish IRINA.RISH@MILA.QUEBEC
Mila, UdeM

Editors: Emtiyaz Khan and Mehmet Gonen

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 generalization gap by $\approx 22\%$, while being $\approx 1/3$ rd the size of the dense baseline model in few-shot restricted imagenet training.

Keywords: Progressive Pruning; Anytime Learning; Replay; Phase Transition

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. 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

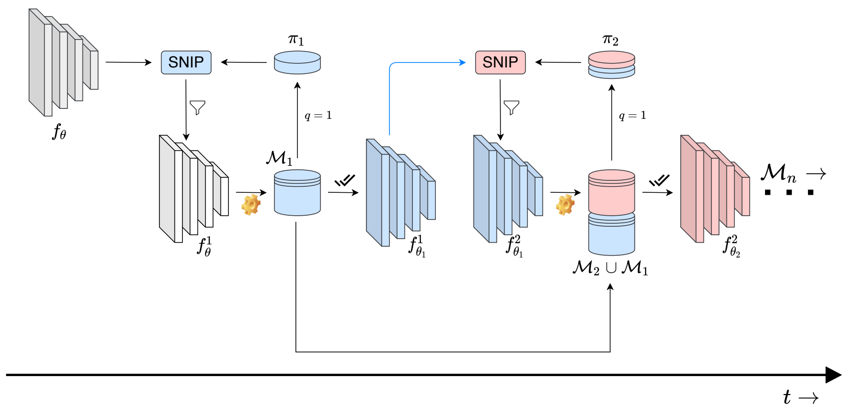

Figure 1: Overview of **Anytime Progressive Pruning** (APP) using full replay with a given randomly initialized dense model f_θ and $|S_B|$ total megabatches.

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 and 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. (2020). However, few studies Chen et al. (2020) have investigated the 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 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 that 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. Anytime Progressive Pruning

In this section, we formally introduce our proposed method *Anytime Progressive Pruning*(APP) as shown in the Fig. 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 $0.2 * |\mathcal{M}_t|$ [2] 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.

To evaluate APP, we use primarily 2 baselines:

1. C-10 and C-100 denote CIFAR-10 and 100 respectively.
2. The $|\cdot|$ denotes the size of the set/stream.

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^{τ} $\times 100\%$ 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.

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

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 generalization 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. Results

3.1. CIFAR-10/ 100 experiments ($|S_B| = 8$)

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 generalization gap. For example, with ResNet-50 (R50) in C-100, APP improved the test accuracy by 17.97% and 11.12%, reduced the CER by 9927 and 5533 and decreased the generalization 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.

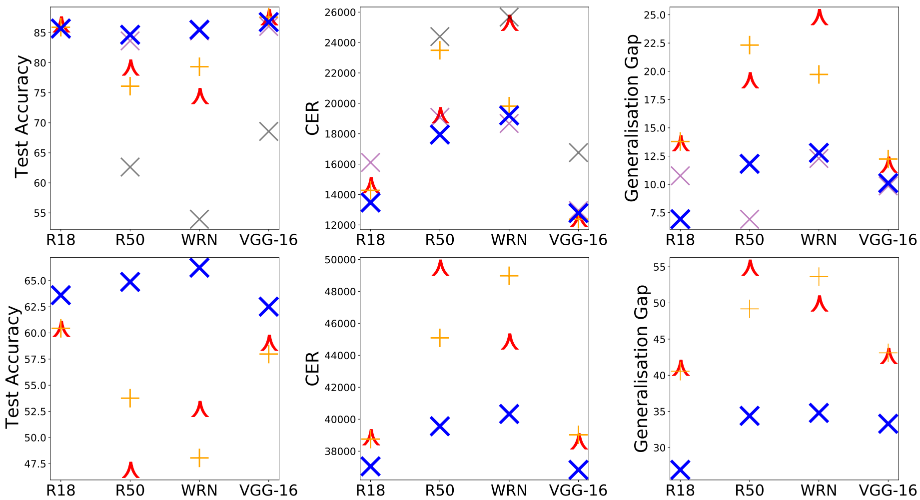

Figure 2: Top and Bottom Rows: L→R Test Accuracy(↑), CER(↓) and Generalization Gap(↓) results for C-10 and C-100, respectively. ⅄, +, ×, × and × represent the baseline, Anytime OSP, APP (Snip), APP (Magnitude), and APP (Random), respectively.

3.2. Few shot experiments on Restricted ImageNet

Table 1: Results on Few-shot Restricted ImageNet ALMA.

Method	$\lvert\mathcal{M}_t\rvert$	$\lvert S_B\rvert$	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.

As reported in Table 1, we observe that APP significantly reduces the generalization gap for each model variant compared to the Anytime OSP and the baseline counterparts. α represent the number of samples per class in the complete dataset. Excluding the experiment of $\alpha = 270$, $\lvert S_B\rvert = 70$, we observed a decrease in CER compared to the baseline model. For

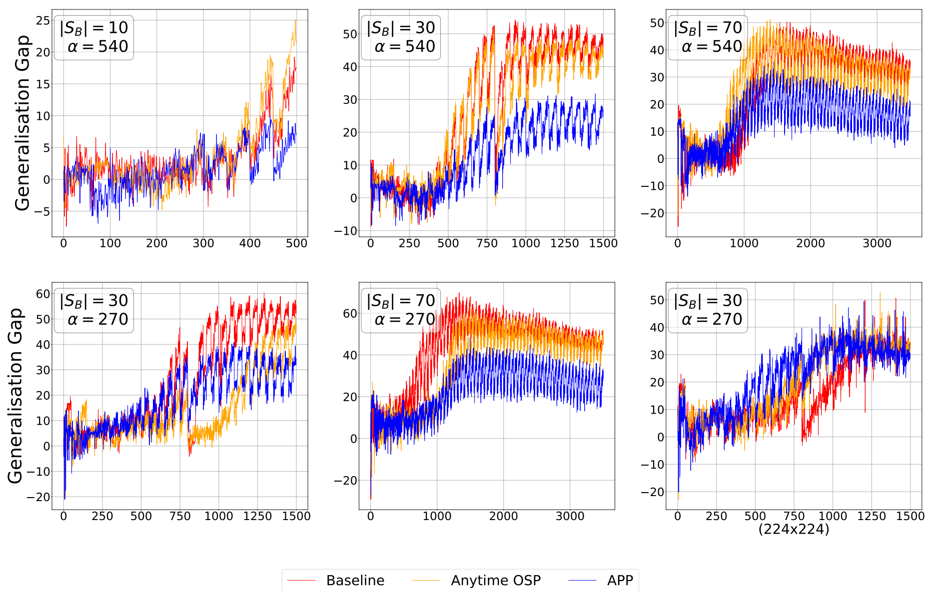

Figure 3: Generalization gap curves during training of APP, Anytime OSP and Baseline for the results in Table 1.

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

3.3. Transitions in generalization gap

We visualize the generalization gap as a function of training iterations across the megabatches in the stream S_B in Fig.3 for the experiments reported in Table 1. We observe 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 generalization gap steadily decreases.

In all subplots, it can be seen that APP consistently maintains a lower generalization 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 generalization gap compared to one-shot pruning methods and the dense baseline model.

References

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

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.

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

Lucas Caccia, Jing Xu, Myle Ott, Marc'Aurelio Ranzato, and Ludovic Denoyer. On anytime learning at macroscale. *arXiv preprint arXiv:2106.09563*, 2021.

Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2020.

Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

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

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019b.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Chaoqi Wang, ChaoQi Wang, Guodong Zhang, and Roger B. Grosse. Picking winning tickets before training by preserving gradient flow. *ArXiv*, abs/2002.07376, 2020.

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Emerging paradigms of neural network pruning. *arXiv preprint arXiv:2103.06460*, 2021.