APP: Anytime Progressive Pruning

Diganta Misra
Landskape AI, Mila, UdeM, Morgan Stanley
diganta@landskape.ai

Bharat Runwal
Landskape AI
bharat@landskape.ai

Tianlong Chen
VITA, UT-Austin
tianlong.chen@utexas.edu

Zhangyang Wang
VITA, UT-Austin
atlaswang@utexas.edu

Irina Rish
Mila, UdeM
irina.rish@mila.quebec

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 ≈ 7% and a reduction in the
generalization gap by ≈ 22%, while being ≈ 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 $D_{x,y}$ 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 $D_{x,y}$, 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 $M_t \in S_B$, we construct the replay inclusive megabatch $M_t$ by taking the union of all previous megabatches along with the current megabatch and then create a small sample set $\pi_t$ of size $0.2 * |M_t|$\(^2\) to be used to prune the model to $0.8^δ_t \times 100\%$ 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 $M_t$ 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 $0.8^\tau \times 100\%$ at the initialization of $f_\theta$ 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 $\pi_1$ of the first megabatch $M_1$ of size $0.2 * |M_1|$.

* **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.

\[
CER = \frac{s_B |T_{x,y}|}{\sum_{t=1}^{s_B} \sum_{j=1}^{|T_{x,y}|} 1(F_t(x_j) \neq y_j)}
\]  

(1)

Here, $T_{x,y}$ represents the held-out test set used for evaluation, $F_t$ represents the trained model at the $t$ -th megabatch, and $F_t(x_j)$ 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 ($M_1$) 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.
3.2. Few shot experiments on Restricted ImageNet

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

| Method       | $|M_t|$ | $|S_B|$ | $\alpha$ | Test Accuracy($\uparrow$) | CER($\downarrow$) | Generalisation Gap($\downarrow$) |
|--------------|--------|--------|----------|---------------------------|-------------------|----------------------------------|
| Baseline     | 756    | 10     | 540      | 43.36%                    | 25328             | 17.39%                           |
| Anytime OSP  | -      | -      | -        | 47.25% (+3.89 %)          | 24978 (-450)      | 21.53% (+4.13 %)                 |
| APP          | -      | -      | -        | 40.40% (-2.96%)           | 24712 (-616)      | 6.96% (-10.43 %)                 |
| Baseline     | 126    | 30     | 270      | 48.03%                    | 66832             | 48.73%                           |
| Anytime OSP  | -      | -      | -        | 50.23% (-2.2%)            | 73206 (-1922)     | 34.42% (-21.08 %)                |
| APP          | -      | -      | -        | 55.04% (+7.01 %)          | 66239 (-2593)     | 26.39% (-22.34 %)                |
| Baseline     | 252    | 30     | 540      | 47.88%                    | 159204 (-917)     | 45.36% (-1.48 %)                 |
| Anytime OSP  | -      | -      | -        | 51.45% (-3.37 %)          | 158608 (-306)     | 39.36% (-14.29 %)                |
| 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. $\alpha$ represent the number of samples per class in the complete dataset. 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 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


