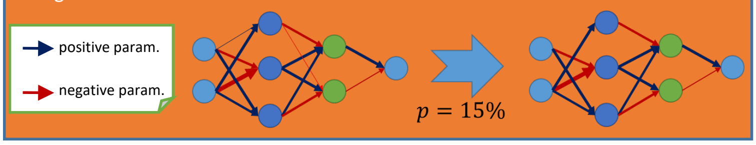
Speeding-up pruning for Artificial Neural Networks
Presenting Accelerated Iterative Magnitude Pruning

Marco Zullich, Eric Medvet, Felice Andrea Pellegrino (University of Trieste, Italy)
Alessio Ansuini (AREA Research & Technology, Trieste, Italy)

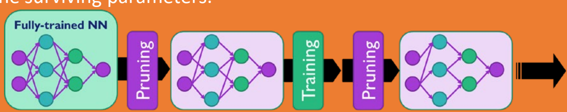
Pruning an Artificial Neural Network (ANN)

ANN pruning is the act of removing one or more connections between neurons.
Pruning can be carried out **during** or **after** training, there exist a fair amount of techniques in both cases.
E.g., pruning during training can be carried out via LASSO or L-1 norm regularization
In this work, we are going to concentrate on pruning **after** training.
Magnitude pruning removes parameters whose absolute value is low w.r.t. the other parameters.
A **pruning rate** $p \in (0,1)$ needs to be fixed which identifies the proportion of weights that will be removed from the ANN.



Re-training and iterating

Since, after pruning, it is likely that the ANN accuracy will suffer of some degradation, usually the pruning is followed by a re-training.
An iterative scheme can hence be devised, in which an iteration consists of re-training and pruning, each time removing a fixed proportion of weights p and re-training the surviving parameters.



Stoppage is performed when (a) the performance of the pruned and re-trained network falls below a threshold, or (b) a target value for *sparsity* is reached.
The iteration of a magnitude pruning scheme is called I**terative Magnitude Pruning (IMP)** [1].

Training assumptions

We suppose the networks, in all iterations, are (re-)trained using Stochastic Gradient Descent (SGD).
Moreover, we assume, concerning the **unpruned** ANN, that:

- it is trained for T epochs
- it is trained using a stepwise Learning Rate (LR) annealing schedule called Λ
- the initial value of the LR is λ_0
- the final value of the LR is λ_f
- the initial and final configuration of the parameters is called Θ_0 and Θ_f, respectively

Techniques for re-training

In the recent literature, there exist three established techniques to re-trained a pruned ANN effectively:

Fine-Tuning (FT) [2]

- ❑ Start re-training the surviving parameters from the values Θ_f
- ❑ Re-train the pruned ANN for $t \ll T$ epochs
- ❑ Use a fixed LR λ_f
- ❑ Usually not applied iteratively, in conjunction with a high p

Weigth Rewind (WR) [3]

- ❑ Before re-training, rewind all surviving weights to initial configuration Θ_0
- ❑ Re-train the pruned ANN for T epochs
- ❑ Use the same LR annealing schedule Λ

Learning Rate Rewind (LRR) [4]

- ❑ Start re-training the surviving parameters from the values Θ_f
- ❑ Re-train the ANN for T epochs
- ❑ Use the same LR annealing schedule Λ

Recent studies [4] has shown that FT, at high pruning rates, produces pruned ANNs consistently worse than WR and LRR as far as accuracy is concerned.

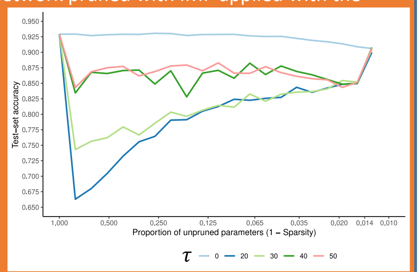
Essential bibliography
[1] S. Han *et al.*, “Learning both weights and connections for efficient neural network”.
[2] Z. Liu *et al.*, “Rethinking the value of network pruning”.
[3] J. Frankle & M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks”.
[4] A. Renda *et al.*, “ Comparing fine-tuning and rewinding in neural network pruning”.

Speeding-up IMP

IMP and WR [source] can effectively produce ANNs with comparable or higher accuracy than their unpruned counterpart while having a much smaller number of parameters.
There are multiple downsides of this, one being that the time for repeatedly re-training the ANN becomes extremely high.
Our work experiments with finding an easy way of speeding-up the execution of IMP by reducing the number of epochs for re-training.
Suppose that we train the unpruned ANN for T epochs and we apply IMP for K iterations: in iterations $1, \dots, K-1$, we re-train the pruned ANN for τ epochs, where $\tau \ll T$. In iteration K, instead, we re-train for T epochs once again.
We call this method **Accelerated Iterative Magnitude Pruning (AIMP)**. Note that it can be applied in conjunction with both WR and LRR.

Experimenting with (A)IMP + WR

In order to experiment with AIMP and compare its performance w.r.t. IMP, we decided to train VGG-19 Convolutional Neural Networks (CNNs) on the dataset CIFAR10 and to apply AIMP with different values of τ for 20 iterations and $p = 0.2$, comparing it against the same network pruned with IMP applied with the same number of iterations.



The values of τ selected were 20, 30, 40, and 50.
Re-training was applied, in all cases, with WR.
In the figure, we can see that, expectedly, while IMP (upper line) shows a high performance throughout the pruning, AIMP steadily recovers, while catching up with the final performance of IMP.

In the table we have the accuracy of the pruned network after 20 iterations of (A)IMP. We can see how AIMP with $\tau = 40, 50$ beats the accuracy of the regular IMP with WR.
AIMP with $\tau = 40$ is 3,47 times faster than the regular IMP.

Method	Accuracy
IMP + WR	90.64%
AIMP $\tau = 50$	**90.71%**
AIMP $\tau = 40$	**90.82%**
AIMP $\tau = 30$	90.39%
AIMP $\tau = 20$	90.01%

Additional experiments

In addition to the experiments presented above, which constitute the bulk of our work, we experimented with additional configurations:

Reducing the number of epochs of training of the unpruned networks (WR)

- ❑ $\tau = 50, p = 0.2$
- ❑ Also the unpruned CNN is trained for 50 epochs
- ❑ Last re-training is operated for 160 epochs
- ❑ Median accuracy = 91,1%

Higher pruning rates (WR)

- ❑ $\tau = 50$
- ❑ $p = 0.3$, 12 iterations of (A)IMP
- ❑ $p = 0.4$, 9 iterations of (A)IMP
- ❑ No large difference between IMP and AIMP, although accuracies generally lower than the (A)IMP with $p = 0.2$ and 20 iterations.

Lower sparsity rates ↔ less iterations of (A)IMP (WR)

- ❑ $\tau = 50, p = 0.2$
- ❑ Apply (A)IMP for $2, \dots, 19$ iterations
- ❑ AIMP can’t keep up to IMP unless sparsity very high

Experiments with (A)IMP + LRR

- ❑ $\tau = 50, p = 0.2$
- ❑ IMP shows slightly higher accuracy than AIMP (93.68% vs 93.62%)
- ❑ In more recent experiments, the gap is higher
- ❑ Increasing τ does not seem to work

Drawbacks

- ❑ We couldn’t find a criterion to identify a good value for τ during re-training
- ❑ AIMP seems to work only when sparsity rates are very high
- ❑ Still some work to do in the comparison with IMP+LRR