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

Marco Zullich*, Eric Medvet*, Felice Andrea Pellegrino*, and Alessio Ansuini†
* Department of Engineering and Architecture, University of Trieste, Italy
Email: marco.zullich@phd.units.it, {emedvet, fapellegrino}@units.it
† AREA Science Park, Trieste, Italy
Email: alessio.ansuini@areasciencepark.it

***Abstract*—In recent years, Artificial Neural Networks (ANNs) pruning has become the focal point of many researches, due to the extreme overparametrization of such models. This has urged the scientific world to investigate methods for the simplification of the structure of weights in ANNs, mainly in an effort to reduce time for both training and inference. Frankle and Carbin [1], and later Renda, Frankle, and Carbin [2] introduced and refined an iterative pruning method which is able to effectively prune the network of a great portion of its parameters with little to no loss in performance. On the downside, this method requires a large amount of time for its application, since, for each iteration, the network has to be trained for (almost) the same amount of epochs of the unpruned network. In this work, we show that, for a limited setting, if targeting high overall sparsity rates, this time can be effectively reduced for each iteration, save for the last one, by more than $50\,\%$, while yielding a *final product* (i.e., final pruned network) whose performance is comparable to the ANN obtained using the existing method.**

***Index Terms*—Artificial Neural Network, Convolutional Neural Network, Neural Network Pruning, Magnitude Pruning, Lottery Ticket Hypothesis.**

I. Introduction

Starting from the 2010s, ANNs have revolutionized the world of machine learning, from computer vision to computational linguistics and reinforcement learning. These models, though, are known to rely on a very large number of parameters: for instance, a rather simple Convolutional Neural Network (CNN) like VGG19 [3] for ImageNet [4] employs around 144 million parameters; BERT [5], a state-of-the-art ANN-based model for computational linguistics, employs up to 340 million parameters.

Various techniques have been thought of for reducing the number of parameters while still preserving (or improving) its performance, pruning being one of them. In 2015, Han *et al.* [6] has introduced the concept of Iterative Magnitude Pruning (IMP), i.e., the removal, at the end of training, of the smallest ANN parameters, followed by a *re-training* phase to recover the performance lost with pruning. The process may be repeated for multiple iterations, each time obtaining an increasingly pruned network with respect to the original one. More recently, Frankle and Carbin [1] and Frankle *et al.* [7] have refined the procedure, introducing a *rewinding* policy according to which, before the re-training phase, the pruned network is brought to a configuration close to the original network initialization. The network is then re-trained for a similar number of epochs the unpruned ANN was trained for. The resulting algorithm, called IMP with Weight Rewind (WR), will be described in detail in Section III-A. Even more recently, Renda, Frankle, and Carbin [2] introduced a different policy for the retraining phase of ANNs, called Learning Rate Rewind (LRR), which does not require the weights to be rewound after each iteration, but requires the training to continue with the same weights (except the pruned ones) of the previous iteration, while resetting the learning rate schedule of the optimizer to its initial state.

Those algorithms, despite being able to produce well-performing ANNs with sparsity rates hovering around $99\,\%$, require considerable time and resources, thus motivating some speed-up strategies. Our work is driven by the idea that the parameters to be pruned may be known before the network has been fully trained. We show that, within the settings of our experiments, the number of epochs for the re-training phase can be reduced by more than $50\,\%$, for all but the last iterations of IMP, with no performance loss. We call our method AIMP: Accelerated Iterative Magnitude Pruning.

II. Related Work

The literature distinguishes between two main categories of pruning techniques: (a) *Unstructured Pruning*: "unstructured pruning prunes individual weights without consideration for where they occur within each tensor" ([2]); IMP belongs to this category. (b) *Structured Pruning*: pruning operates on neurons or groups of units (e.g., full convolutional filters); an example is [8]. Note that iterative procedures akin to IMP may be designed also for structured pruning techniques, as done, for example, in [2].

With the recent introduction of many pruning techniques for ANNs, there has been a number of new works trying to characterize the pruned network: for instance, [9]–[11] debate, with somehow conflicting results, about the robustness of pruned ANNs; Ansuini *et al.* [12] analyzed the similarity between layers of pruned and unpruned CNNs, showing that pruned networks might be forced to learn high-level features differently from the unpruned ones.

More recent work dealt with: (a) the strategies for re-training a pruned network originating from a pre-trained one; and (b) the differences in effectiveness between structured and unstructured techniques. In particular, Paganini and Forde [13]

analyzed the latter, providing similarities between unstructured and structured pruning techniques. Renda, Frankle, and Carbin [2], instead, focused on both. They compared two re-training policies: full re-training with Weight Rewind (WR), introduced by Frankle *et al.* [7], and *fine-tuning*, introduced by Liu *et al.* [14]. The former is implemented by (i) *rewinding* the pruned weights to a situation close to the initial one and (ii) re-training the network for the same amount of epochs of the unpruned one, keeping the same learning rate schedule. The latter, instead, (i) does not rewind weights (i.e., the parameters surviving the pruning keep the same value they had before pruning) and (ii) re-trains the pruned network for a smaller number of epochs, keeping the learning rate constant at the last value it had before pruning. The authors of [2] showed that, in many scenarios of unstructured and structured pruning, WR seems to behave better than fine-tuning; moreover, they introduce a third strategy, Learning Rate Rewind (LRR), that empirically outperforms both in the majority of tasks. It is a compromise between WR and fine-tuning: it consists in re-training starting with the same parameters obtained from the previous iteration, for the same amount of epochs as the first training, while re-utilizing the same learning rate schedule as done in the first training. Moreover, they showed that unstructured techniques produce pruned networks achieving better test-set accuracy with respect to those obtained with structured techniques.

In the same work, the authors debated over another open issue of ANN pruning: the long time which is required for the re-training of the pruned networks. Proven the ineffectiveness of fine-tuning, which constituted *de facto* a strategy to reduce re-training time, they tried to lower the number of epochs for the re-training for the various iterations of WR and LRR, though achieving a smaller final accuracy, despite noticing that "accuracy saturates after re-training for about half of the original training time".

There exist works trying to speed-up network pruning, for instance by pruning as soon as the unpruned network is being trained, such as in [15], thus eliminating the need for a pre-trained overparametrized ANN.

In our work, instead, inspired by [2], we aim at reducing the re-training epochs for some iterations of IMP. We show that, if the final sparsity of the pruned network is sufficiently high, we can effectively reduce the time to execute IMP by *truncating* the training of such iterations when using either WR and LRR.

III. Background

Hereby we will present the two main iterative pruning techniques used in this work: IMP+WR and IMP+LRR. In the following schemes, we denote with retrain() the re-training algorithm for IMP. It takes as input an ANN, the number of epochs for retraining, the vector of initial parameters, a *pruning mask*, and a learning rate schedule. The pruning mask (which we denote with the letter $\boldsymbol{m}$) is the binary vector identifying which parameter (a) survives the pruning (corresponding entry value 1) or (b) has been pruned from the network (value 0); the learning rate schedule identifies at which epochs the learning rate gets annealed. The routine retrain() outputs the performance of the re-trained network and the vector of parameters after the training is finished. The re-training is carried out like a standard ANN training, with the only difference that the gradient is not propagated back to parameters having a corresponding 0 entry in the mask (i.e., they have been pruned from the network).

A. IMP+WR

IMP+WR was introduced in [1] and refined in [7]. At each iteration, IMP+WR prunes parameters and rewinds the surviving parameters at the value they had at an epoch t of their original training. As of [7], t is called *late resetting epoch* and is usually taken between 1 and $7\,\%$ of the total training.

The procedure is the following ($\odot$ denotes the element-by-element matrix multiplication):

Algorithm 1: IMP+WR.

input : A dense ANN $\mathcal{N}$ fully trained for T epochs with learning rate schedule Λ; a late resetting epoch t defined as before; $\boldsymbol{\theta}_T^{(0)}, \boldsymbol{\theta}_t^{(0)} \in \mathbb{R}^K$, vectors holding the parameters of $\mathcal{N}$ at epoch T and t, respectively; fixed pruning rate $p \in (0,1)$; maximum number of iterations I; threshold for minimum performance of the pruned network $\pi^\star$.

output: A pruned ANN with parameters $\boldsymbol{\theta}_T^\star$

for $i \in \{1, ..., I\}$ **do**
  $\tilde{\boldsymbol{\theta}}^{(i-1)} = \text{abs}(\boldsymbol{\theta}_T^{(i-1)})$
  $\eta_p = p$-th quantile of $\tilde{\boldsymbol{\theta}}^{(i-1)}$
  $\boldsymbol{m}^{(i)} = \mathbf{1}_K$ `// mask is vector of ones`
  for $k \in \{1, ..., K\}$ **do**
    `/* if magnitude of parameter is smaller than the quantile, set corresponding mask entry to 0 */`
    if $\tilde{\theta}_k^{(i-1)} \leq \eta_p$ **then**
      $m_k^{(i)} = 0$
    end
  end
  $\boldsymbol{\theta}_t^{(i)} = \boldsymbol{\theta}_t^{(i-1)} \odot \boldsymbol{m}^{(i)}$ `// weight rewind`
  $\pi^{(i)}, \boldsymbol{\theta}_T^{(i)} = \text{retrain}(\mathcal{N}, T - t, \boldsymbol{\theta}_t^{(i)}, \boldsymbol{m}^{(i)}, \Lambda)$
  if $\pi^{(i)} < \pi^\star$ **then**
    break
  end
end

B. IMP+LRR

IMP+LLR was proposed in [2] as a better alternative to IMP+WR in terms of test accuracy.

Its formulation is similar to IMP+WR, but it gets rid of the weight rewind. Considering the algorithm described in

Section III-A, IMP+LLR obtains a new parameters vector by multiplying the final parameters of iteration $i-1$ with the mask. Rows 10 and 11 become, respectively,

$$\boldsymbol{\theta}_0^{(i)} = \boldsymbol{\theta}_T^{(i-1)} \odot \boldsymbol{m}^{(i)},$$

and

$$\pi^{(i)}, \boldsymbol{\theta}_T^{(i)} = \text{retrain}(\text{net}, T, \boldsymbol{\theta}_0^{(i)}, \boldsymbol{m}^{(i)}, \Lambda).$$

IV. Accelerated Iterative Magnitude Pruning

With the aim of reducing the time to obtain pruned ANNs, we propose the following modification to the re-training algorithm of IMP: we define a quantity $\tau \in \{1, \ldots, T-1\}$ which we call *partial training epochs* such that, for all the *intermediate* iterations of IMP (i.e., from the first one to the second-to-last one) we re-train the network only for this number of epochs[1].

This modification is compatible with both IMP+WR and IMP+LRR: practically, it could be applied to all those iterative pruning techniques requiring a re-training phase. In our work, though, we limit our experiments to IMP+WR and IMP+LRR. One more note on the algorithm: probably, the intermediate iterations will not produce networks performing on par with the original network. As a consequence, the *if* construct at row 12 has to be dropped.

We call the corresponding algorithm Accelerated Iterative Magnitude Pruning (AIMP). In the following sections, whenever the number of partial training epochs τ is specified, we indicate it as a subscript to the acronym, i.e., AIMP$_\tau$ indicates that we ran AIMP with τ partial training epochs.

An undesirable behaviour which we noted on our models, which got more frequent as τ was decreased, is the phenomenon of exploding gradient. We tackled it by operating norm *gradient clipping* [16], [17] at 1 at each optimization step.

V. Experiments, Results, and Analysis

In order to analyze the effectiveness of our method, we trained a CNN on the dataset CIFAR10 with the VGG19 [3] architecture as described in [18], removing the two hidden fully-connected layers and applying batch normalization before the ReLU activation in each hidden layer.

CIFAR10[2] is a widely-used dataset for image classification composed of 60 000 images (of which 10 000 belonging to the test-set) of size 32×32 arranged in 10 classes.

We used the Stochastic Gradient Descent (SGD) optimizer, with hyperparameters as in [18]: initial learning rate of 0.1, momentum of 0.9, weight decay of 0.0001; for computational reasons, we reduced the batch size to 128, as in [1]. We set the epochs of initial training (T) to 160. We annealed the learning rate by a factor of 10 at epochs 80 and 120. The experiments were executed on single GPU.

[1] Actually, when using IMP+WR, we rewind the parameters to the situation at iteration t, so we re-train for only $\tau - t$ epochs.

[2] https://www.cs.toronto.edu/~kriz/cifar.html

For each training epoch, we shuffled the mini-batches for SGD and applied random transformations (cropping, flipping, affine transformations) to the original images.

Unless stated otherwise:

- we performed IMP for 20 iterations and used $p = 20\,\%$;
- we applied AIMP and IMP in conjunction with WR with late resetting epoch $t = 1$ (i.e., the pruned weights were rewound at the situation found after the first epoch of the initial training);
- we performed 5 runs for each combination of the hyperparameters and computed the median performance indexes.

We performed several experiments to evaluate AIMP in different conditions and from different points of view.

In Section V-A, we present the results of AIMP+WR applied with $\tau \in \{20, 30, 40, 50\}$ and we compare them with IMP+WR. In Section V-B, we show that AIMP$_{50}$+WR is effective also when the original training of the unpruned network is truncated at 50 epochs. In Section V-C, we show the results obtained by applying AIMP$_{50}$+WR and IMP+WR with p increased to 30 and $40\,\%$. In Section V-D, we compare the performance of AIMP$_{50}$+WR to that of IMP+WR applied for a number of iterations smaller than 20. In Section V-E, we show the results obtained by AIMP$_{50}$ and IMP+LRR.

A. AIMP+WR with 20 iterations and pruning rate $p = 20\,\%$

We applied AIMP+WR with $p = 20\,\%$ and $\tau = 50$, 40, 30, and 20.

For this round of experiments, we analyze the performance under two facets: effectiveness, as test-set accuracy, and efficiency, as search-cost. The latter is defined in [2] as the computational resources required to run the whole pruning algorithm. This quantity is approximated by total epochs of re-training for all the pruning algorithm iterations.

1) Test-set accuracy analysis: Median results are shown in Table I. It can be seen that applying AIMP+WR with $\tau = 30$, 40, and 50 produces results on par with the regular IMP+WR procedure. Lowering the epochs to 20, instead, produces slightly worse results. The effect of AIMP+WR and IMP+WR on pruning can be appreciated in Figure 1: *regular* IMP+WR produces models with comparable performance for all iterations; with AIMP+WR, instead, we observe that all the intermediate byproducts are, as expected, very underperforming. The final iterations produce the results we see in Table I, second and third column.

2) Search-cost analysis: In our setting, IMP requires the networks to be re-trained for 160 epochs for all 20 iterations, which means we train for a total of $20 \cdot 160 = 3200$ epochs (as done in [2], we do not consider the first training in this calculation). Following this scheme, if we consider the generic AIMP$_\tau$, we have $19 \cdot \tau + 160$ total training epochs; obtaining the results in Table I, fourth and fifth column. We can see that, in order to get a CNN with approximately $1\,\%$ of the weights of the original network with a comparable performance, with AIMP we can save more than $70\,\%$ of time with respect to the regular IMP, resorting to AIMP$_{40}$.

TABLE I

RESULTS FOR AIMP+WR WITH DIFFERENT VALUES OF τ COMPARED TO THE REGULAR IMP+WR ($\tau = 0$), BOTH WITH $p = 20\,\%$. Δ_{IMP} REFERS TO THE DIFFERENCE IN ACCURACY OF AIMP$_\tau$ W.R.T. IMP. SPEED-UP IS CALCULATED AS $\frac{\text{RE-TRAINING EPOCHS IMP}}{\text{RE-TRAINING EPOCHS AIMP}_\tau}$.

τ	Acc.	Δ_{IMP}	Total retraining epochs	Speed-up
0	0.9064		3200	
50	0.9071	+0.0007	1110	2.88
40	0.9082	+0.0018	920	3.47
30	0.9039	−0.0025	730	4.39
20	0.9001	−0.0063	540	5.92

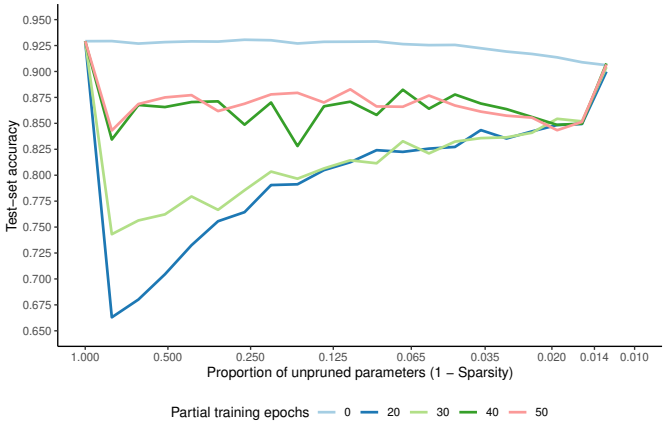

Fig. 1. Median test-set accuracy vs. proportion of unpruned weights during the execution of the pruning method for models pruned with AIMP$_\tau$+WR (with $\tau \in \{50, 40, 30, 20\}$), compared to the model pruned with IMP+WR ($\tau = 0$), both with $p = 20\,\%$. For the 20th and last iteration, all models are trained for 160 epochs.

TABLE II

RESULTS FOR AIMP$_{50}$+WR AND IMP+WR FOR DIFFERENT VALUES OF p. SPARSITY IS CALCULATED AS $\frac{\text{NUMBER OF PRUNED PARAMETERS}}{\text{NUMBER OF PARAMETERS OF UNPRUNED MODEL}}$.

			Median test-set accuracy	
p [%]	Iterations	Sparsity	IMP+WR	AIMP$_{50}$+WR
20	20	0.9873	0.9064	0.9071
30	12	0.9850	0.9081	0.9053
40	9	0.9887	0.8999	0.8998

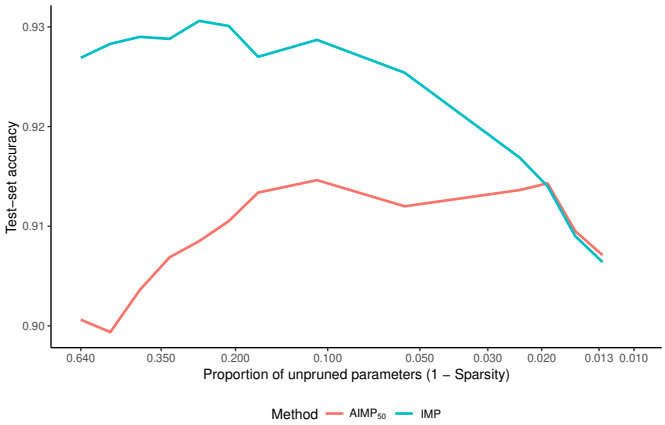

Fig. 2. Median test-set accuracy vs. proportion of unpruned weights for AIMP$_{50}$+WR and IMP+WR applied to lower values of p. Given a number N iterations, leading to a desired sparsity rate in the final product, re-training is applied for 50 epochs up to the $(N-1)$th iteration, and eventually for 160 epochs.

B. Reducing the number of epochs of the original training

One question which may arise is the following: is it possible that the pruned network performs similarly as the ones presented in the previous section if also the unpruned network is trained for the same number of epochs τ? The answer is: yes.

We explored this scenario with the same setting as those presented before: VGG-19 for CIFAR10, same optimizer hyperparameters, first training operated for 50 epochs, AIMP$_{50}$+WR applied for 20 iterations; in the last iteration, the network trained for 160 epochs.

The five networks obtained with this method achieved a median test-set accuracy of 91.1 %, which is higher than both IMP+WR (90.6 %) and AIMP$_{50}$+WR (90.7 %).

C. AIMP with higher pruning rates

Another question which might arise is: is it possible that AIMP works only because the pruning rate is too small? The answer is: no.

We decided to increase p to 30 % and 40 %, producing a pruned model with a similar sparsity and performance as the one obtained with $p = 20\,\%$.

With $p = 30\,\%$, we applied IMP+WR and AIMP$_{50}$+WR for 12 iterations; with $p = 40\,\%$, for 9 iterations.

The results are shown in Table II. As we can see, aside from some small differences for the networks pruned with $p = 30\,\%$, the test-set accuracy is very similar independently of the chosen method. Moreover, with $p = 40\,\%$, both IMP+WR and AIMP$_{50}$+WR are slightly outperformed by their counterparts obtained with $p = 20\,\%$ and $p = 30\,\%$.

D. AIMP with lower sparsity rates

We then decided to compare the accuracy of the models pruned via AIMP+WR to that of models pruned via IMP, at various sparsity rates. To allow for this comparison, we repeatedly applied AIMP$_{50}$+WR for a smaller number of iterations, from 3 to 19: the corresponding pruned network obtained with AIMP was always trained for 160 epochs for the last iteration. The performances of IMP+WR are the same as in Figure 1.

The results of the comparison are shown in Figure 2. It can be seen that, for the longest part of the pruning process, AIMP+WR could not keep up with the performance of IMP+WR; the gap, though, seemed to progressively reduce, until, at around 2 % of parameters remaining, the performances of IMP+WR and AIMP+WR were almost identical. This indicates that, most likely, when pruning the network aggressively, AIMP becomes a viable alternative to IMP+WR.

E. AIMP+LRR

Given the recent introduction of LRR as an alternative to WR, we decided to try if AIMP_{50} could produce quality results also in conjunction with this new technique.

The setting was the same Section V-A, but instead of applying WR, in each iteration we applied LRR: we trained the network starting from the final pruned weights of the previous iteration, rewinding the learning rate to 0.1. Since τ was 50, the learning rate stayed constant at 0.1 for all iterations, except for the 20th and last one.

The results are promising: the 5 repetitions for IMP+LRR achieved a median test accuracy of $93.68\,\%$, while AIMP_{50}+LRR scored a median test accuracy of $93.62\,\%$.

F. Analysis

In this section we want to address the question: is there some indicator of AIMP effectiveness during the pruning process? Ideally, we would like to find an indicator that, during a re-training phase of AIMP, tells that a *steady state* has been reached, so we may stop training, proceed with the pruning, and start the next iteration. We will not answer the question definitely, but we provide some tools which may be investigated further in future works in order to find said indicators.

The candidate indicators we consider are:

- pruning masks;
- parameters;
- validation-set accuracy[3].

We decided not to analyze the gradient because of the effects of gradient clipping (see Section IV) which might introduce distortions in the distribution.

1) Pruning masks: Let $\tilde{\boldsymbol{m}}_t^{(i)}$ be the *potential* pruning mask obtained at iteration of IMP+WR i and epoch t, and $\tilde{m}_{t,j}^{(i)}$ its jth entry. We use the term *potential* since pruning is not operated at that moment; rather, we're just calculating the mask that we would have obtained, had we pruned the network at epoch t of iteration i. Let $P^{(i)}$ be the total number of parameters in the model at iteration of AIMP i. We define two indicators:

$$D^{(i)} = \frac{\sum_{j=1}^{P^{(i)}} \mathbb{1}\left(\tilde{m}_{t,j}^{(i)} \neq \tilde{m}_{t-1,j}^{(i)}\right)}{P^{(i)}}$$

$$\tilde{D}^{(i)} = \frac{\sum_{j=1}^{P^{(0)}} \mathbb{1}\left(\tilde{m}_{t,j}^{(i)} \neq \tilde{m}_{t-1,j}^{(i)}\right)}{P^{(0)}},$$

where $\mathbb{1}$ is the indicator function:

$$\mathbb{1}(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$

Practically speaking, both quantities identify the fraction of elements within the mask that have changed with respect to the previous epoch, i.e., parameters that would have been pruned at the previous epoch and would not have been pruned at the current one, or vice-versa; in $D^{(i)}$ the proportion is relative to the current number of parameters in the model, while in $\tilde{D}^{(i)}$ is relative to the total number of parameters in the initial model. Note that $D^{(0)} = \tilde{D}^{(0)}$.

[3]The validation set was obtained by holding out $20\,\%$ of images from the test set.

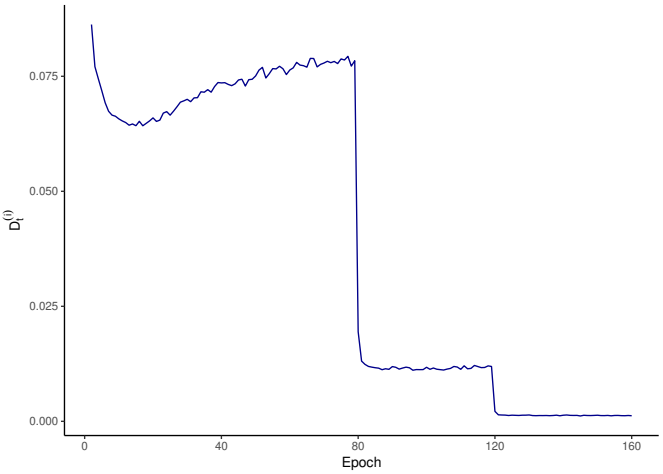

Fig. 3. $D^{(0)}$, dynamics of the pruning mask as training is performed. The jumps at iteration 80 and 120 are due to the drop in learning rate. Results are extracted from a single run.

First, in Figure 3 we see the trend of $D^{(0)}$, the mask dynamics for the first training, as training is performed. We see that, after each change of learning rate, the mask becomes more and more stable, and, after epoch 120 a very small fraction of parameters changes. This might be indicative of the fact that, in order to determine a pruning mask, the epochs 120 to 160 are hardly needed for the first training, due to the very small variations in the mask from epoch to epoch.

Then, in Figure 4, we see the trend of $D^{(i)}$ and $\tilde{D}^{(i)}$ for a selected number of iterations of AIMP+WR. We notice that, as the iteration number increases, the absolute number of changes gets smaller and smaller (right chart), which is to be expected, since there are less parameters in the model left to be pruned (given that the p is constant at each iteration); instead, if related to the number of parameters present in the model at each iteration (left chart), the proportion of changes within the mask increases as the number of iterations (thus, the sparsity) increases.

As far as the behaviour across epochs is concerned, it would seem that the value for $\tilde{D}^{(i)}$ stabilizes after 20 to 40 iterations (even if at various rates throughout the different iterations), while $D^{(i)}$ seems to be stable earlier in the re-training as the iteration increases.

However, we do not see, in those indicators, any immediate element in support of a choice of a particular level of τs.

As a side note, it can be seen that between subsequent epochs there may be a large portion of parameters changing between the pruning masks. As already noted in [13], there are multiple *lucky* subsets of parameters within a dense ANN which, when isolated (e.g., via pruning methods) and re-trained, can lead to a performance comparable to that of the initial network. Originally, the lottery ticket hypothesis [1]

postulated the existence of one of these subnetworks.

2) Parameters: We considered two indicators, namely the L_∞ norm of the parameter vector, and the $L_1^{\text{TOP}(k)}$ measure, i.e. the sum of the magnitude of the greatest (in magnitude) k entries. Let $\boldsymbol{\theta}_t^{(i)}$ be the vector of parameters at epoch t of training for AIMP iteration i. We explored the application of $L_1^{\text{TOP}(k)}$ on $\boldsymbol{\theta}_t^{(i)}$ with various values of k, both dependent and independent from the number of parameters in the model; eventually, we resorted to a fixed k because, by doing so, we always operate on a subspace of the parameter space having the same dimension, thus yielding comparable values no matter the iteration number. The final choice of 1000 is rather arbitrary, as other values (like 500 and $10\,000$) yielded similar trends.

Figure 5 shows the trends for $L_1^{\text{TOP}(1000)}$ and L_∞ (which is equivalent to $L_1^{\text{TOP}(1)}$): we can see how the latter seems, in many iterations, to settle after 20 to 40 epochs, while the former does not stabilize, and keeps increasing as the iteration increases.

The stabilization of L_∞ may also be a signal of stabilization of the training process, and could play a role in finding a criterion to determine τ in a dynamical way, iteration by iteration.

3) Validation-set performance: Figure 6 illustrates the validation-set accuracy for iterations 1 to 19 of AIMP, for all the three choices of τ (30, 40, and 50). The final test-set accuracies of the pruned networks were, respectively, $90.5\,\%$, $90.6\,\%$, and $90.5\,\%$. The results are obtained from a single run, they are not averages of the 5 multiple trials we ran.

We observe that:

- Especially for the first iterations of AIMP, there is a significant amount of performance gained by stopping training at 50 rather than 40 or 30 epochs: for the first iteration, accuracy is around $78\,\%$ at the 50th epoch vs. around $73\,\%$ at the 30th epoch; the same trend is much lighter for the latest iterations, where the gain stands at around $2.5\,\%$ of accuracy.
- For all the choices of τ, the higher the iteration of AIMP, the higher the performance throughout the training.
- Not one of the networks comes close, in terms of performance, to the final test-set accuracy of $90.5\,\%$ (for the models obtained with AIMP_{30} and AIMP_{50}) and $90.6\,\%$ (for the model obtained with AIMP_{40}). Anyway, we can notice that, stopping training when validation-set accuracies were above $75\,\%$ produced, after the 20th AIMP iteration, a good final product, competitive with respect to the original network pruned with IMP+WR.

All to all it is hard, based on the data at our disposal, to formulate a criterion to decide, in a dynamical fashion, when to stop training during the intermediate AIMP iterations. There are still many questions to be answered, and many experiments to conduct before using validation-set performance in support of the selection of the level of τ.

4) Takeaways: In the previous sections we have presented an analysis of parameters, pruning masks, and validation-set accuracy data for each epoch of re-training and iteration of AIMP in hope to find regularities or trend(s) supporting a given choice of τ. Except for maybe the analysis of L_∞ norm, we have not noticed anything immediate from our investigations, but, with more experiments and analyses, and maybe employing different metrics or quantities, it might be possible to find some criteria to support the selection of a given τ.

Since then, AIMP remains a method based solely on heuristics, and the hyperparameter τ needs to be found for each network architecture and dataset.

VI. Conclusion, limitations, and future work

In this work, we presented AIMP, a pruning technique for ANNs based off of IMP. Our method consists into reducing the number of epochs of re-training for the intermediate iterations of IMP.

Our experiments, although performed on a limited array of settings, indicate that AIMP can produce pruned ANNs with same or better test-set accuracy than the networks with same sparsity rate obtained with the regular IMP (both with WR or with LRR) when the target sparsity rate is sufficiently high (starting from about $98\,\%$ in our case), while speeding up the execution of the algorithm by a factor of almost 3.

Evidence suggests that, when the target sparsity rate is not large enough, AIMP fails to meet the performance achieved by the homologous networks produced via IMP.

One downside of our algorithm is that the number of iterations of AIMP (or, equivalently, the target sparsity to achieve) have to be set beforehand: probably, all the iterations before the last one will have a performance uncomparable to the one of the unpruned network; consequentially, a stopping criterion for the algorithm cannot be based on the performance of the *intermediate* pruned networks.

Moreover, there are some points, the analysis of whom we leave for future work:

- We experimented on a limited setting: VGG19 trained on CIFAR10, with 5 repetitions of the experiment for each set of hyperparameters considered. It should be interesting to observe if AIMP works with different, more complex networks (e.g., Resnet[19]) and datasets (e.g., ImageNet).
- We applied AIMP+LRR only with $\tau = 50$, $p = 0.2$, and 20 iterations. Does the same trend of *underperformance* applies for AIMP+LRR when the number of iterations is smaller? Does AIMP+LRR perform well with different values of τ and p?
- The objective of our work was to introduce AIMP and present its results in comparison to IMP. We didn't focus extensively on analyzing methods to determine dynamically, during training, good values of τ, rather presenting tools which might help future works in addressing this issue.

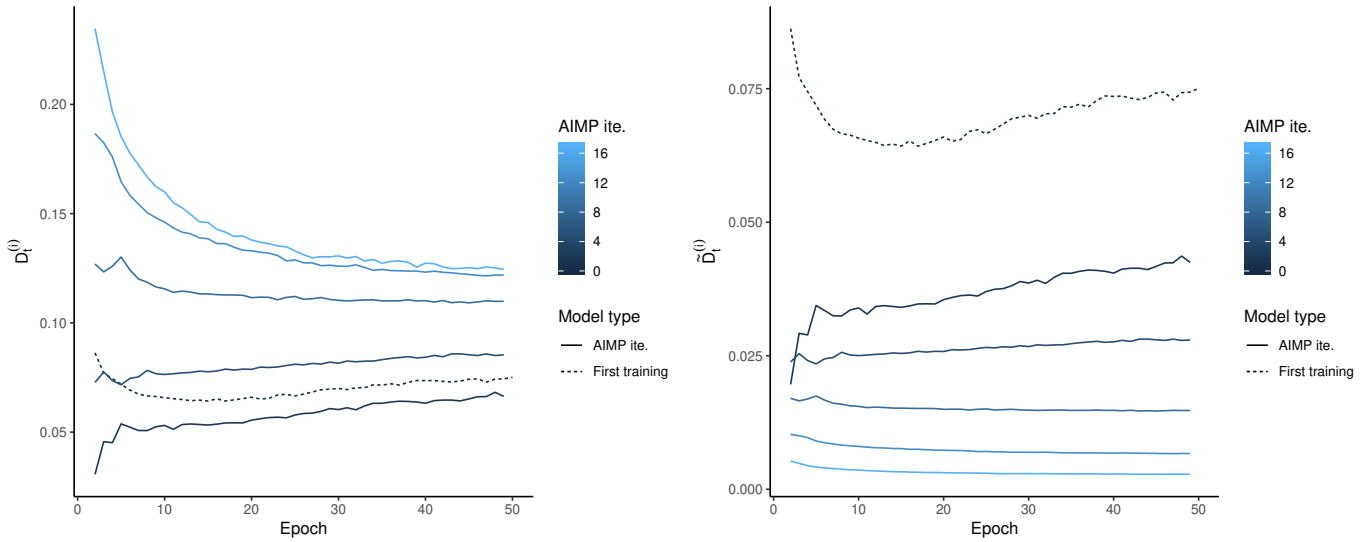

Fig. 4. $D^{(i)}$ and $\tilde{D}^{(i)}$ respectively, for a selected number of iterations (2, 5, 9, 13, and 17) of AIMP_{50}. The dashed line shows the quantity for the first training. Results are extracted from a single run.

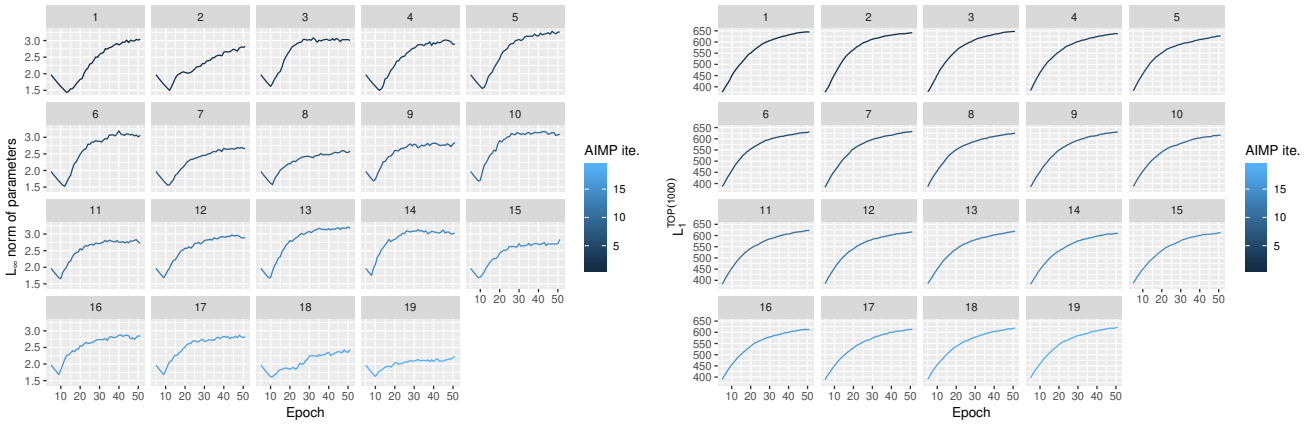

Fig. 5. $L_1^{\text{TOP}(1000)}$ (left) and L_∞ (right) norms of the parameters vs. training epoch for each iteration of AIMP_{50} with WR where pruning is applied. Results are extracted from a single run.

REFERENCES

[1] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7.

[2] A. Renda, J. Frankle, and M. Carbin, "Comparing fine-tuning and rewinding in neural network pruning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1gSj0NKvB.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://www.aclweb.org/anthology/N19-1423.

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

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

[8] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE transactions on neural networks and learning systems*, 2019.

[9] J. Cosentino, F. Zaiter, D. Pei, and J. Zhu, "The search for sparse, robust neural networks," *arXiv preprint arXiv:1912.02386*, 2019.

[10] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Towards compact and robust deep neural networks," *arXiv preprint arXiv:1906.06110*, 2019.

[11] S. Ye, K. Xu, S. Liu, H. Cheng, J.-H. Lambrechts, H. Zhang, A. Zhou, K. Ma, Y. Wang, and X. Lin, "Adversarial robustness vs. model compression, or both," in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 2019.

[12] A. Ansuini, E. Medvet, F. A. Pellegrino, and M. Zullich, "On the Similarity between Hidden Layers of Pruned and Unpruned Convolutional Neural Networks," in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2020)*, M. De Marsico, G. Sanniti di Baja, and A. Fred, Eds., La Valletta: Scitepress, Feb. 2020, pp. 52–59, ISBN: 9789897583971.

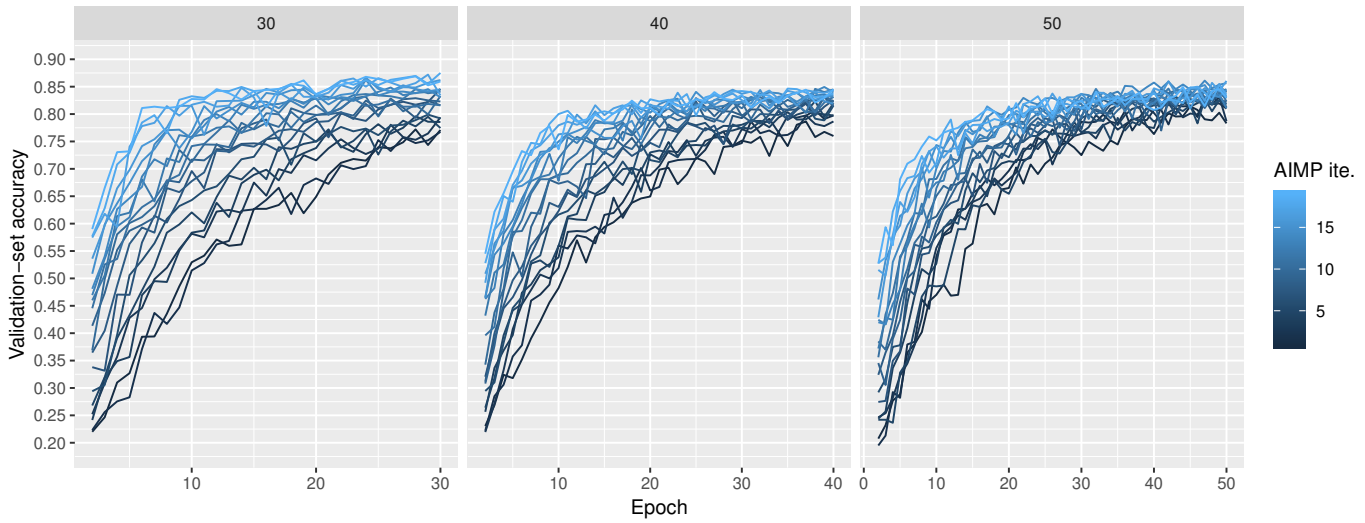

Fig. 6. Validation-set accuracy for the 19 intermediate iterations of AIMP_{30}, AIMP_{40}, and AIMP_{50} for $p = 20\,\%$ and full initial training.

[13] M. Paganini and J. Forde, "On iterative neural network pruning, reinitialization, and the similarity of masks," *arXiv preprint arXiv:2001.05050*, 2020.

[14] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJlnB3C5Ym.

[15] M. Aamir Raihan and T. M. Aamodt, "Sparse weight activation training," *arXiv*, arXiv–2001, 2020.

[16] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[17] J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: A theoretical justification for adaptivity," in *International Conference on Learning Representations*, 2019.

[18] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," *arXiv preprint arXiv:2002.10365*, 2020.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.