Combining Relevance and Magnitude for Resource-saving DNN Pruning

Combining Relevance and Magnitude for Resource-saving DNN Pruning

C. F. Chiasserini*†‡§, F. Malandrino†‡, N. Molner¶, Z. Zhao*
* Politecnico di Torino, Italy – † CNR-IEIIT, Italy – ‡ CNIT, Italy
§ Chalmers University of Technology, Sweden – ¶ iTEAM Research Institute, Universitat Politècnica de València, Spain

***Abstract*—Pruning neural networks, i.e., removing some of their parameters whilst retaining their accuracy, is one of the main ways to reduce the latency of a machine learning pipeline, especially in resource- and/or bandwidth-constrained scenarios. In this context, the pruning *technique*, i.e., how to choose the parameters to remove, is critical to the system performance. In this paper, we propose a novel pruning approach, called FlexRel and predicated upon combining training-time and inference-time information, namely, parameter magnitude and relevance, in order to improve the resulting accuracy whilst saving both computational resources and bandwidth. Our performance evaluation shows that FlexRel is able to achieve higher pruning factors, saving over 35% bandwidth for typical accuracy targets.**

***Index Terms*—Distributed learning, Resource utilization, Machine Learning model compression.**

I. Introduction

In the last years, computing at the edge has been gaining importance as more sectors get digitalized and require processing of data closer to the end users. This includes the storage of data as well as the intelligence, e.g., machine learning (ML) to process and extract knowledge from such information. However, ML – especially when implemented through deep neural networks (DNNs), has significant requirements. In addition to computational resource consumption in the data centers, critical issues are represented by the bandwidth consumption due to the information transfer on the radio link from the (potentially large number of) data sources and the training and inference time required by large (ML) models for training and inference. Indeed, mobile applications typically demand for a swift inference outcome, which can be challenging to obtain in a resource-constrained scenario like the network edge. The magnitude of the challenge is such that the SA6 group of 3GPP is discussing model transferring approaches to alleviate it.

ML model compression [1]–[3] has recently emerged as a way to address this crucial issue and a significant amount of work has leveraged this approach to find the optimal balance between the ML model size (hence, complexity) and performance. For instance, [2] developed an algorithm to compress ML models on-demand offering flexibility to run them on different devices at the cost of increasing time complexity. [4] introduced the concept of knowledge distillation (KD) to compress the knowledge of a set of full and specialized models into a single one, improving accuracy in neural networks that are not very deep. [5] considers pruning the DNN parts impacting most on the latency, ensuring shorter epoch training time, however more epochs may be required, incurring in similar global convergence time. [6] investigates quantization based methods to obtain highly compact DNNs that can be efficiently computed at the price of decreasing the accuracy compared to full-precision methods.

Among the above techniques, pruning represents a very appealing option due to its low complexity and, as noted above, it allows for small ML models while preserving high-quality inference as far as the adopted pruning (i.e., compression) factor is not too high, and no significant drift is observed in the input data statistics. The justification behind pruning is the so-called lottery-ticket hypothesis, introduced in the seminal paper [7]: only a small portion of the DNN parameters have a significant impact on its effectiveness, hence, the rest can be safely removed. The hypothesis, however, does not say *which* are the "winning tickets", hence, several pruning *techniques* have been devised to identify – and avoid removing – them.

Fig. 1 summarizes the most relevant ways to identify the parameters to prune: filled boxes represent input information; patterned boxes correspond to operations; ellipses represent ways to combine or

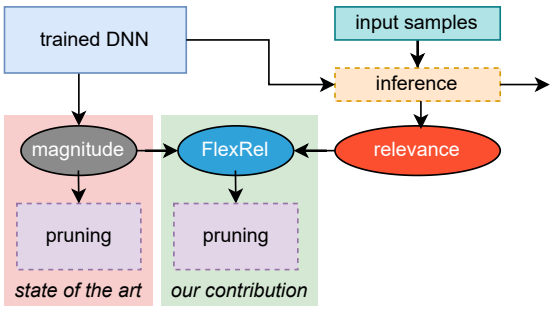

Fig. 1. Three ways to make pruning decisions: the traditional way (on the left-hand side), i.e., directly considering the *magnitude* of the DNN parameters and pruning those with the smallest magnitude; *relevance-based* pruning (on the right-hand side) – with relevance being a quantity computed during inference – using both the DNN parameters and input samples. Our FlexRel approach (in the middle), which combines *both* magnitude and relevance to make more effective pruning decisions.

use output data. The most common pruning technique is magnitude-based pruning [7], represented on the left-hand side of Fig. 1; it is fairly straightforward and simply removes the lowest-magnitude ML model parameters. In spite of its simplicity, magnitude-based pruning has been consistently shown to be fairly effective, and had been further refined through alternative approaches that trade higher complexity for better accuracy. Examples include works that use the *gradients* of parameters instead of their magnitude [8], and approaches that seek for an optimal *pruning threshold*, and remove all parameters whose magnitude falls below it. Other works instead solve an optimization problem to decide whether or not each parameter shall be pruned. Recent works extend pruning to emerging types of network, including recurrent ones and transformer architectures [9]. Finally, works like [10] collect real-world experiments and experience with model pruning, and draw conclusions on the best strategies to maximize its effectiveness.

Additionally, there exist other techniques that, although not originally designed with pruning in mind, can be successfully adapted to it. A prominent example is parameter relevance, a methodology designed for Artificial Intelligence (AI) explainability that can be extended, as better explained in Sec. III-A, to DNN pruning. Prominent among such techniques is *parameter relevance*. The relevance of each parameter expresses how much influence the value of each DNN parameter has on its final output; importantly, it also means that changing the value of a low-relevance parameter – or, crucially, removing it altogether – will not significantly change the DNN output. Thanks to this property, relevance scores can be used as an alternative to magnitude to select the parameters to prune, or both magnitude and relevance – as opposed to magnitude alone – can be considered when making pruning decisions. As depicted on the right-hand side of Fig. 1, relevance scores and magnitude are obtained at different moments in time. Specifically, magnitude can be observed *a priori*, during the training of the DNN or immediately after; relevance, on the other hand, is computed *a posteriori*, considering a trained DNN and a specific set of input data. It follows that computing relevance could be more onerous than observing magnitude; on the positive side, relevance accounts for more valuable information.

Our main contributions, also highlighted in the center of Fig. 1, can be summarized as follows:

- We identify *relevance* as a promising metric to exploit when making pruning decisions;
- We make the key observation that both high-magnitude and high-relevance parameters contribute to the learning performance, albeit through different mechanisms;
- We propose a new pruning approach, called FlexRel, exploiting both magnitude and relevance to select the parameters to prune;
- We validate our intuition through a set of experiments using publicly-available, popular DNNs and datasets, finding that FlexRel leads to a substantial performance improvement in terms of latency and bandwidth consumption at the cost of a modest amount of additional computing complexity;
- We further discuss when, given the features of the scenario at hand, it is beneficial to use FlexRel.

The rest of the paper is organized as follows. Sec. II discusses some relevant related work. Sec. III explains our concept and methodology, while Sec. IV validates our approach and demonstrates its superiority with respect to state-of-the-art alternatives. Sec. V highlights open issues for future research, and Sec. VI concludes the paper.

II. Current Pruning Approaches

Pruning is a popular solution for ML model compression [11], [12]. The ambition of model

pruning is to remove some model parameters either at the end of the training to get a smaller, less complex model, or after each epoch as the training proceeds (a.k.a. dynamic pruning) to also make each epoch duration shorter. We remark that, as pruning reduces the size of an ML model, in the case of distributed training or inference, it can also significantly decrease the amount of information that needs to be transferred over the communication links connecting the nodes that contribute to the task as well as the overall latency. Such scenarios have gained prominence recently, as ML models are deployed towards the edge of the network due to the fact that large amounts of data needed for training are collected at the edge and that data input to inference is often generated at the end devices and ML-based applications require low latency. Further, even if an ML model is trained or fine-tuned in a cloud server, it may then need to be delivered to edge nodes, in which case, transmitting the model would imply a non-negligible network load.

Notably, most works on pruning focus on pruning by magnitude, as it is a simple, yet effective, technique. Essentially, if the magnitude of a given parameter is small, then its influence on the DNN output is limited, hence, it is preferable to prune that parameter rather than other, higher-magnitude ones. Key to this technique is the fact that weights can be cut *a priori*, i.e., just by observing the magnitudes without the need to compute any extra value. Other works, however, propose new metrics for pruning, focusing on, e.g., the evolution of gradients [8]. Focusing on distributed scenarios, [5] accounts for communication issues when making pruning decisions, e.g., pruning away those parts of the DNN with the largest latency footprint.

Alternative model compression approaches pursue the same goal by focusing on information compression [1]–[3]. In particular, [1] proposes lossy *network compression* by representing with fewer bits the least important parameters of the DNN, i.e., those deemed less likely to be winning tickets [7]. Finally, [3] introduces semantic compression to preserve model explainability.

To the best of our knowledge, **existing pruning techniques only account for information available at *training* time**, e.g., the magnitude of DNN parameters and their evolution over training epochs. Furthermore, **there is no work combining different pruning techniques**, i.e., different ways of choosing the parameters to prune, to increase the learning performance.

III. The FlexRel Approach

All pruning techniques are predicated on keeping the most important parameters in the DNN (i.e., those deemed most likely to be "winning tickets") and removing the rest. As depicted in Fig. 1, pruning techniques essentially differ in what quantities (e.g., parameter magnitude or gradients) are used to select the parameters to prune. Further, current pruning techniques exploit and combine information generated at *training* time, such as parameter magnitude; however, there is no need for this to be the case.

This is especially important in *split learning* scenarios, where mobile nodes and edge-based servers cooperate in running the same learning task, e.g., the training of a DNN. The mobile node hosts the local data and runs the first layers of the DNN, then, intermediate results are sent to the edge where the rest of the layers are hosted – hence, the "split" name. In split learning scenarios, the total learning time (i.e., the duration of each epoch) is given by *two* components, to wit, the computation time and the network delay. High-quality pruning decisions might require more computation, but this can be compensated by the higher pruning fractions they allow, hence, the need to send less data over the network.

In this context, the key intuition behind our FlexRel scheme is to exploit additional information, generated during the *inference* phase, to make pruning decisions. The reason is twofold:

1) In general, considering both the training and inference phases allows gathering more information, hence, potentially making better decisions;
2) Especially in scenarios where training and testing datasets can be qualitatively different [1], [2], [5], the parameters that matter the most during inference (the so-called "winning tickets") may not be the same as those that evolve the most during training.

To leverage on our intuition, we need a way to quantify the importance of DNN parameters during inference; however, no ready-made metric akin to magnitude exists. Therefore, we first define a new metric based on relevance (Sec. III-A), and then discuss how to use it in Sec. III-B.

A. Inference-time importance of DNN parameters: Relevance scores

We begin by considering an aspect that is different from pruning but related to it, namely, *input relevance*. The basic goal of input relevance, which has been introduced in [13], is to assess which parts of each input sample (e.g., which pixels of an image) had the largest impact on the DNN decision (e.g., the class assigned to the image). As an example, in the case of image classification, background pixels will have lower relevance than those belonging to an object. Input relevance is computed through the *activity maximization* framework, a set of mathematical techniques seeking for the input data (e.g., the pixels) that maximize a certain part of the output (e.g., the logit associated with the selected class). These techniques are applied recursively from the output layer of the DNN to its input. The result of the operation is a *score* associated with each parameter of the DNN, expressing its impact on the output; it follows that scores also express how much the output itself would change if the parameter were removed. Importantly, such high-relevance parameters are not necessarily the highest-magnitude ones.

Input relevance is computed at inference time and accounts for inference-time input information, hence, it fits very well the goal of our FlexRel scheme to account for the inference phase as well. However, given the relevance scores defined for input information, we need to understand how to exploit such scores to identify the DNN parameters to prune, i.e., we need to extend the notion of relevance to the model parameters. The extension is straightforward for fully-connected DNN layers where, denoting with n the size of the input vector, each element b_{ij} of the output is given by $b_{ij} = \sum_{k=1}^{n} a_{ij} w_{kj}$, i.e., a summation of products between elements a_{ij} of the input and model parameters w_{kj}.

We can interpret the above formula as *linking* parameters w_{kj} with elements a_{ij} of the input and elements b_{ij} of the output. Recalling that we can compute the relevance of both the input and the output as per [13], we make the relevance of each parameter proportional to the relevance values of input and output elements linked to it. Specifically, indicating with rel the relevance, we have:

$$\mathsf{rel}(w_{kj}) = \sum_{i=1}^{n} \left[\mathsf{rel}(a_{ik}) + \mathsf{rel}(b_{ij})\right]. \qquad (1)$$

Intuitively, if a parameter connects relevant inputs with relevant outputs, then it must be relevant itself.

It is worth remarking that, virtually, all modern DNNs consist of types of layers other than fully-connected; most relevantly, image classification DNNs heavily feature convolutional layers. Notably, convolutional layers have been shown to admit equivalent fully-connected representations, hence, we can further extend our notion of parameter relevance to convolutional networks. For convolutional neural layers, we exploit the fact that, as shown in [14], convolution operations can be transformed into matrix products, hence, convolutional layers can be transformed into fully-connected ones (those used in multi-layer perceptrons). We can then use (1) to compute the relevance of the transformed parameters.

B. Using Relevance Scores

Given their ability to capture the inference-time behavior of the DNN and to account for inference-time inputs, one might be tempted to take relevance as the sole criterion to select which parameters to prune, thus altogether replacing magnitude with relevance. However, in doing so, one might risk losing the information obtained during training and carried by magnitude values. Indeed, as we mentioned earlier, high-relevance parameters tend to have high-relevance elements (e.g., pixels) as *both* input and output. At the same time, high-magnitude parameters might connect high- and low-relevance ones elements, precisely because of their large magnitude. Importantly, *both* actions are important parts of the way DNN operate, and both need to be preserved when performing pruning.

Indeed, relevance and magnitude seek to ask the same question, i.e., which parameters of the DNN are the most important, albeit in different ways. As reported in Tab. I, FlexRel seeks to combine both into a single score, aiming at keeping only those parametes that exhibit both high relevance and high magnitude parameters. Specifically, for each parameter, magnitude and relevance are first normalized between 0 and 1, and then summed in a weighted manner, with δ indicating the weighting factor. The weighting factor δ also allows for managing conflicts or inconsistencies between relevance and magnitude. Specifically:

- Parameters with high magnitude and high relevance will have very high score, close to 1 (the maximum possible);
- Parameters with low magnitude and low relevance will have very low score, close to 0 (the minimum possible);
- Parameters with low magnitude and high relevance will have a score close to δ;
- Parameters with high magnitude and low relevance will have a score close to $1-\delta$.

It follows that, by selecting δ, we can also decide how high (or low) the score of parameters where relevance and magnitude do not match will be.

The main features of the pruning techniques we consider are summarized in Tab. I. It is worth mentioning that traditional, magnitude-based pruning already works reasonably well in many cases, thanks to the extensive research devoted to it, as reviewed in Sec. II. Accordingly, our goal with FlexRel is not to replace existing methodologies, but rather to complement and perfect them.

TABLE I
COMPARISON BETWEEN PRUNING TECHNIQUES

Strategy feature	SoTA: magnitude	Benchmark: relevance	Ours: FlexRel
inputs	magnitude M	relevance R	both M and R
score s	$s \leftarrow M$	$s \leftarrow R$	$s \leftarrow \delta M + (1-\delta)R$
requires relevance	no	yes	yes
effectiveness	good	medium	best
best suited for	CPU-constrained scenarios		balanced scenarios

In the next section, we explore the impact of the weighting factor δ – expressing the relative importance we assign to relevance and magnitude.

IV. NUMERICAL RESULTS

In the interest of reproducibility, we design our reference scenario using free, popular, and publicly-available models, datasets, and information. Specifically, our experiments use the VGG16 DNN and the ImageNet dataset for image classification. VGG16 is a convolutional DNNs widely used for computer vision applications, with 16 layers and about 138 million parameters. ImageNet is a popular dataset including 1,000 classes and over 1.2 million images, and is one of the most challenging – and most used – benchmarks when performing image classifications. We use the Python programming language, the PyTorch framework, and the Adam optimizer, with a starting learning rate of 0.05.

We perform a total of 50 training epochs: 10 with the full DNN (i.e., before pruning), and 40 with the pruned DNN (i.e., after pruning). All information used for pruning (magnitude and/or relevance) is computed after the first 10 epochs. Pruning is always structured, e.g., pruning decisions concern entire channels ("filters") of convolutional layers and not individual parameters. We compare the following three pruning *techniques*, i.e., ways to choose the filters to remove, exploiting different kinds of information:

(i) *Magnitude*, i.e., the SotA approach of removing the filters with the lowest average magnitude;
(ii) *Relevance*, i.e., removing the filters with the lowest relevance;
(iii) Our *FlexRel* approach, combining both relevance and magnitude as previously explained.

The three techniques correspond, respectively: *(i)* to the state of the art approach [5], [7], [12], *(ii)* to an approach based exclusively on the notion that relevance can be a good guidance in pruning decisions, and *(iii)* an approach integrating such a notion within existing, well-established, well-performing pruning methodologies.

The connectivity between the mobile device and edge node is modeled accounting for typical 5G data rates[1] of 150 Mbit/s in uplink and 20 Mbit/s in downlink. As in all split learning scenarios, the mobile device and the edge node have to exchange the input/output of the so-called cut-layer at each epoch.

Roadmap. We study the impact of three main quantities: *(i)* the pruning factor, *(ii)* the accuracy target, and *(iii)* the weight factor δ. Specifically, through our performance evaluation, we characterize the relationship between each of these quantities and the overall behavior of the system, and how pruning strategies influence such relationships.

Fig. 2 reports the accuracy achieved as a function of the pruning factor, for different pruning techniques. Recall that pruning a certain fraction of the weights reduces, by virtually the same fraction, both the computations to perform and the quantity

[1]Source: OpenSignal.com, "Benchmarking the Global 5G Experience – June 2023".

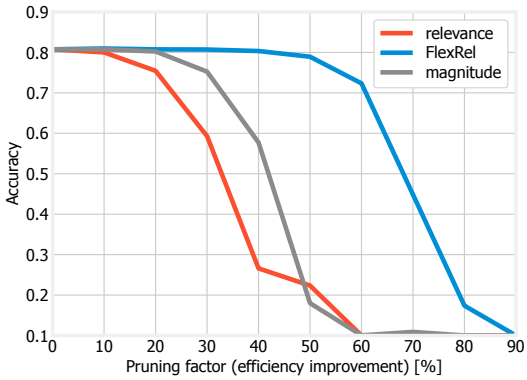

Fig. 2. Accuracy reached by the VGG16 DNN when trained over the ImageNet dataset as a function of the pruning factor, for different pruning techniques.

of data to transfer across nodes. Pruning fractions thus map, one-to-one, to efficiency improvements. We can distinguish three main regions in each of the curve:

- At first, for small pruning factors, the accuracy remains almost constant;
- As pruning factors grow, the accuracy drops in a roughly linear manner;
- Further increasing pruning, the accuracy stays constant to a very small value.

The third region shall always be avoided; in most cases, we want to operate around the border between the first and second region.

We can observe that using magnitude alone provides very good performance, which is consistent with the popularity of this approach. Using relevance alone results, instead, in a substantially lower accuracy. Intuitively, only considering that metric results in pruning high-magnitude parameters, which adversely affects learning performance. Most importantly, *combining* relevance and magnitude as per our FlexRel approach results in the best performance, even better than magnitude. This key result validates our intuition: magnitude and relevance express two different – albeit related – quantities, *both* of which have a bearing on the final learning quality. Consequently, considering both when making pruning decisions results in the best performance. As an example, considering an accuracy target of 70%, we are able to prune around 25% of parameters if we consider relevance, around 35% of parameters if we consider magnitude and over 60% with FlexRel, for an overhead reduction of over 30%.

Fig. 2 also shows that FlexRel pushes to the right the border between the first and the second of the three regions outlined earlier. In other words, it possible to attain higher pruning factors (hence, higher efficiency), exceeding 30% and up to 50%, without significantly degrading the learning performance (i.e., accuracy). On the negative side, computing the relevance requires additional time, which might in principle negate the accuracy gains highlighted in Fig. 2. To verify whether or not this is the case, we vary the *accuracy target* between 10% and 80%, and quantify, for each technique, (i) how long it takes to reach that accuracy level, and (ii) how such time is spent. The results for the magnitude-based and FlexRel approach are summarized in Fig. 3 (the relevance-based technique is not represented due to its lower performance).

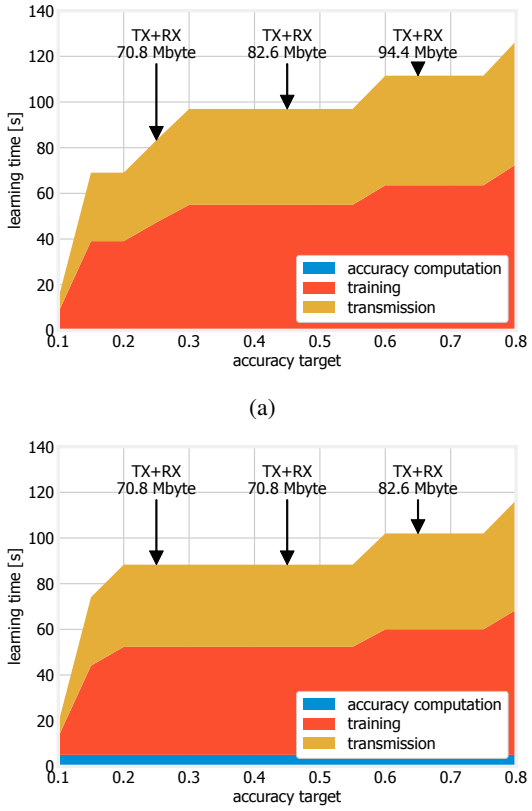

Fig. 3. Elapsed learning time as a function of the accuracy target, for the magnitude-based (a) and FlexRel (b) techniques. Numbers in the plot represent the quantity of transmitted data.

Next, we look at training times and investigate to which extent the additional computations needed to obtain relevance values impact the total elapsed time. Fig. 3 shows how the learning time changes as a function of the accuracy target, for the

magnitude-based and FlexRel techniques. We can indeed observe an additional contribution coming from the need to compute the relevance (blue area at the bottom). However, such a contribution is almost negligible compared to the time it takes to perform the actual training, i.e., the forward- and backward-passes (red areas) and the network delays (yellow areas). Comparing the total height of the colored areas between the two plots, we can see that FlexRel (Fig. 3(b)) results in substantially shorter learning times than the magnitude-based technique (Fig. 3(a)), in spite of the additional complexity.

The reason can be again inferred from Fig. 2: given a target accuracy level, e.g., 0.5, FlexRel reaches that level with a much higher pruning factor (in the example, 70%) than magnitude-based pruning (in the example, 45%). More aggressive pruning results in both less computation being performed (hence, shorter computation times) and less data being transferred between learning nodes and learning server (hence, shorter network delays). These two factors abundantly offset the extra time required to compute the relevance.

Focusing on the quantity of transmitted data, reported in Fig. 3, we can observe that FlexRel results in a smaller quantity of transmitted data, which is consistent with the shorter time spent for network transmissions. Importantly, transmitting less data over the air also results in further benefits, including lower energy consumption, less congestion, and smaller likelihood of communication issues.

Last, in Fig. 4 we study the effect of the weighting factor δ over the system performance. We can observe that the best performance is obtained for intermediate values of δ. In other words, both considering magnitude alone (i.e., δ=0) or relevance alone (i.e., δ=1) results in suboptimal performance; in the latter case, the accuracy might become particularly low. Also notice how the best value of δ, identified by a star in the plots, depends upon the pruning factor, hence, some fine-tuning might be required.

In summary, we can observe that our FlexRel approach allows for (i) better learning quality given a pruning factor, and (ii) shorter learning times – accounting for all contributions thereof – given a learning quality target. It thus minimizes the potential drawbacks of pruning (i.e., learning quality degradation) while amplifying its benefits (i.e., shorter learning times).

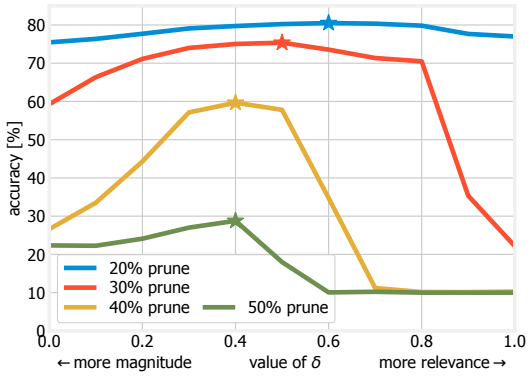

Fig. 4. Effect of the weighting factor δ over the achieved accuracy, for different pruning factors.

This is due essentially to FlexRel's ability to account for multiple factors, i.e., relevance and magnitude, when making pruning decisions. Indeed, relevance and magnitude contribute to the training quality in different ways. Specifically:

- High-relevance parameters tend to have high-relevance elements (e.g., pixels) as *both* input and output;
- High-magnitude parameters can *change* high-relevance elements into low-relevance ones, and vice versa.

By accounting for both metrics, FlexRel can avoid removing parameters that do either thing, hence, minimize the impact of pruning itself on the global learning quality.

V. Discussion and Open Issues

Our experiments have shown the benefits of combining different methodologies – exploiting both training- and relevance-time information – to quantify the importance of parameters to improve pruning performance, to wit, magnitude and relevance. By doing so, our FlexRel scheme can improve both learning quality and training time. In addition to being a viable, effective technique on its own right, FlexRel and our analysis thereof point out several avenues for further investigation, as discussed next.

An important research direction consists in considering additional metrics, beyond magnitude, to choose the parameters to prune. Indeed, as discussed in Sec. II, some existing works have sought to replace magnitude by gradient evolution [8] or even *ad hoc* scores. Taken in isolation, such alternative metrics have been found to perform similar (or slightly better) than magnitude. Therefore, it would

be interesting to assess whether combining those metrics with relevance brings the same benefit we have observed with FlexRel.

A second area worthy of investigation is the dynamic aspect of pruning. As mentioned above, computing parameter relevance is a computationally inexpensive task, hence, it could be done at multiple training epochs. This would make it possible to leverage the *evolution* of relevance to make pruning decisions, in a manner similar to what works like [8] do with gradients. In addition to the potential performance benefit, such a study would shed further light on the evolution of parameter relevance across epochs, hitherto a poorly explored topic.

In scenarios where learning is performed in a distributed manner (e.g., through FL), the relevance of both parameters and inputs can be exploited as a way to assess the value of local datasets – hence, the nodes owning them – to the overall training. Intuitively, if the local parameters of a certain node and/or its local dataset have low relevance, then that node might not give a substantial contribution to the distributed training. This can be of great help in scenarios when *node selection* [15] is a crucial task, owing to either cost or learning time considerations.

Finally, we remark that the advantages of computing relevance may be limited in two main cases. The first is represented by nodes with very constrained computational capabilities, which simply cannot afford to compute relevance (e.g., their battery would deplete if they tried). The second case includes those scenarios where relevance *could* be computed, but that would not help because very little or no pruning could be performed anyway, e.g., because the number of parameters in the DNN model is already very small. With reference to Fig. 2, in those scenarios accuracy would drop very quickly as the pruning factor grows, hence, choosing more carefully the parameters to prune provides no additional gain.

In a general sense, all the research directions sketched above contribute towards a *holistic* view of ML, where the way each learning task is approached accounts for such elements as the available data, the node(s) performing the training, and the model to use.

VI. Conclusion

This paper proposes a new pruning approach, FlexRel, combining the most widely used metric for pruning DNNs, i.e., *magnitude*, with a non-traditional pruning metric, *relevance*; in doing so, it is able to leverage both training- and inference-time information. Such a combination offers better performance than each of the metrics individually used, as magnitude and relevance express two different – albeit related – properties of DNN parameters, both of which have an impact on the final learning quality.

As shown by our experimental results, both actions are important parts of the way DNNs operate, and FlexRel keeps both, allowing for (i) better learning quality given a pruning factor, and (ii) shorter learning times – accounting all contributions thereof –, given a learning quality target with (iii) reduced latency and bandwidth consumption. It thus minimizes the potential drawbacks of pruning (i.e., affecting learning quality) while amplifying its benefits (i.e., shorter learning times).

Acknowledgments

The work was partially supported by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under the MULTIX project Grant Agreement No. 101192521.

References

[1] P. Hegde, G. de Veciana, and A. Mokhtari, "Network adaptive federated learning: Congestion and lossy compression," in *IEEE INFOCOM*, 2023.

[2] F. Malandrino, G. Di Giacomo, A. Karamzade, M. Levorato, and C. F. Chiasserini, "Matching DNN compression and cooperative training with resources and data availability," in *IEEE INFOCOM*, 2023.

[3] H. Xie and Z. Qin, "A lite distributed semantic communication system for internet of things," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 142–153, 2020.

[4] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[5] T. Jian, D. Roy, B. Salehi, N. Soltani, K. Chowdhury, and S. Ioannidis, "Communication-aware dnn pruning," in *IEEE INFOCOM*, 2023.

[6] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 365–382.

[7] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *ICLR*, 2018.

[8] M. Shen, P. Molchanov, H. Yin, and J. M. Alvarez, "When to prune? a policy towards early structural pruning," in *IEEE/CVF CVPR*, 2022.

[9] W. Kwon, S. Kim, M. W. Mahoney, J. Hassoun, K. Keutzer, and A. Gholami, "A fast post-training pruning framework for transformers," *NeurIPS*, 2022.

[10] F. Hohman, M. B. Kery, D. Ren, and D. Moritz, "Model compression in practice: Lessons learned from practitioners creating on-device machine learning experiences," in *ACM CHI*, 2024.
[11] C. Zhong, Y. He, Y. An, W. W. Ng, and T. Wang, "A sensitivity-based pruning method for convolutional neural networks," in *IEEE SMC*, 2022.
[12] T. Zhang, X. Ma, Z. Zhan, S. Zhou, C. Ding, M. Fardad, and Y. Wang, "A unified DNN weight pruning framework using reweighted optimization methods," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 493–498.
[13] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital signal processing*, 2018.
[14] X. Ding, C. Xia, X. Zhang, X. Chu, J. Han, and G. Ding, "Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition," in *IEEE CVPR*, 2022.
[15] F. Malandrino and C. F. Chiasserini, "Toward node liability in federated learning: Computational cost and network overhead," *IEEE Communications Magazine*, 2021.

Carla Fabiana Chiasserini (M'98, SM'09, F'18) is currently a Full Professor with the Department of Electronic Engineering and Telecommunications at Politecnico di Torino, a WASP Guest Professor at Chalmers University of Technology, and a CNIT and CNR Research Associate. Her research interests include edge computing, network support to machine learning, and design and performance analysis of mobile networks and services.

Francesco Malandrino (M'09, SM'19) earned his Ph.D. degree from Politecnico di Torino in 2012 and is now a researcher at the National Research Council of Italy (CNR-IEIIT). His research interests include the architecture and management of wireless, cellular, and vehicular networks.

Nuria Molner obtained her Ph.D. from Universidad Carlos III de Madrid in 2021. Currently, she is a researcher at iTEAM Research Institute of Universitat Politècnica de València (iTEAM-UPV).

Zhao Zhiqiang earned his master degree from Politecnico di Torino in 2024. He is currently a research fellow at Politecnico di Torino.