Titre: Title:	Operations Research Techniques for Neural Network Compression
Auteur: Author:	Matteo Cacciola
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Cacciola, M. (2023). Operations Research Techniques for Neural Network Compression [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/56771/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Operations Research Techniques for Neural Network Compression

MATTEO CACCIOLA
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Novembre 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Operations Research Techniques for Neural Network Compression

présentée par **Matteo CACCIOLA**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Charles AUDET, président
Dominique ORBAN, membre et directeur de recherche
Andrea LODI, membre et codirecteur de recherche
Antonio FRANGIONI, membre et codirecteur de recherche
Daniel ALOISE, membre
Thiago SERRA, membre externe

ACKNOWLEDGEMENTS

The past four years have been an extraordinary journey filled with unexpected challenges and incredible moments. Throughout this intense expedition, numerous individuals provided unwavering support, making this final achievement possible. To all of you, I extend my heartfelt gratitude.

I am deeply grateful to my advisor Antonio, I'm sure I would not have even started a Ph.D. if it wasn't for you. Thanks for always being there for me and supporting me since the start of my academic career over 7 years ago.

I am also extremely grateful to my advisor Andrea. Thanks for making me part of your exceptional research group, for giving me the freedom I needed, and for being very patient with my mistakes. Your expertise, encouragement, and dedication have been instrumental in shaping this thesis.

Thanks, Vahid, Alireza, Massoud, and XinLin, for being my coauthors and for guiding me during my time at Huawei.

A very special thanks goes to Khalid and Jiaqi, for those months when it was only us defending the old fort during the pandemic.

Thank you, Fede, for being my brother, for pushing me to overcome my limits, and for asking me the right questions when I needed it the most. Thank you, Youssouf, for being the best office mate I could have asked for. Thanks, Didier, for having so much fun with me, no matter if outside there were -30 degrees. Thanks, Alex, for being my friend despite my jokes about France and my provocations during our chess games. Thanks, Claudia, for the *spettegolezzi*, for the parties, and for all the fun we had together. Thanks to Lea, Thibaut, Leandro, Marius, De Feng, Nico, Tommaso, Medhi, Maria, Edo, Pierre, Jingyi, and all the cool people of the 5th floor, you are truly wonderful. Thanks to all the members of the DS4DM, even if I met many of you only for 3 months, it was truly amazing being part of such an enthusiastic and united group.

Thanks, Claudio and Gabri, for the moral support, your wise advice, and the amusing time together. Thank you, Marco, for the dinners and the coffee work sessions filled with meaningful conversations. Thanks, Theo, Kriss, Fiona, Claudine, Nizard, and Alessandra for the chalets, the trips, and the aperos that lighted up my time in Montreal. Thanks to Ylenia, Martin, Carlos, Hani, Dan, Linda, Pietro, and all my friends in Montreal, each of you is a part of this journey.

Thanks, Kahlid, for being the cornerstone of this experience, for your wisdom, for our long conversations, and your invaluable friendship.

Thanks to my friends from my time in Pisa, it is always wonderful to see you. Thanks, Dario and Ale, every time we reunite it is like we never separated.

Thanks, Mamma, Papà, Maddi, Giordi, Irene, Nonna, and all my family, I will always bring you with me, no matter where I am in the World.

RÉSUMÉ

Dans de nombreux scénarios exigeant des prévisions basées sur les données, les modèles d'Apprentissage Automatique (*ML*) atteignent constamment les performances de l'état de l'art. À noter en particulier, les Réseaux des Neurones (*NNs*), qui ont affirmé leur domination ces dernières années dans des domaines divers tels que la Vision par Ordinateur et le Traitement automatique du Langage Naturel. La précision des prédictions des nouveaux modèles *NN* s'améliore chaque année, bien que cela s'accompagne d'un compromis en termes de tailles d'architectures plus importantes. Cela se traduit par une augmentation des besoins en stockage mémoire et en puissance de calcul pour la formation et l'exécution des modèles. Le nombre substantiel de paramètres dans les architectures *NN* récentes a nécessité le développement de techniques de Compression de Modèles (*MC*). Ces techniques visent à réduire les ressources nécessaires à un modèle tout en préservant son intégrité en matière de performance. Dans ce contexte, l'utilisation de techniques de Recherche Opérationnelle (*OR*) semble très appropriée. La formulation inhérente du problème de haut niveau consiste à minimiser les ressources requises par le modèle tout en préservant une précision assez proche de la precision du modèle. Cela ouvre naturellement la voie à diverses approches potentielles de *OR* pour formuler mathématiquement une solution à ce défi. En outre, une grande partie des techniques existantes dans la littérature sur la *MC* ne propose pas de méthodologies dont la base théorique est suffisamment solide, mais plutôt des méthodologies simplistes qui reposent sur un raisonnement heuristique. Par ailleurs, la compréhension présente dans la littérature des mécanismes sous-jacents responsables de l'efficacité de certaines algorithmes de *MC* et de leurs principes opérationnels demeure restreinte.

L'objectif principal de cette thèse est de formuler et de développer des méthodologies de *MC* qui exploitent les techniques de Recherche Opérationnelle, qui possèdent des bases théoriques solides, et qui contribuent à découvrir les principes directeurs de ce domaine.

Dans un premier temps, nous introduisons le *SPR*, un nouveau terme de régularisation conçu pour l'élagage des *NNs*. Nous commençons par présenter une représentation mathématique du problème d'élagage, suivie d'une reformulation plus solide obtenue grâce à la fonction de perspective. Par simplification algébrique, nous dérivons finalement l'expression conclusive du *SPR*. Ce terme, lorsqu'il est incorporé dans la fonction de perte pendant la formation, facilite l'obtention d'architectures creuses. Nous validons empiriquement l'efficacité de cette technique en l'appliquant à l'élagage de filtres, en utilisant les architectures de convolution les plus courantes et les ensembles de données CIFAR et Imagenet.

Ensuite, nous nous tournons vers le domaine de la Quantification. Cette technique a connu une augmentation significative de son adoption par les praticiens ces dernières années, bien que ses principes sous-jacents restent insaisissables. Nous menons une analyse de convergence de l'entraînement de *NN* concernant les architectures en virgule flottante à faible nombre de bits. Notre enquête conduit à l'établissement de nouvelles limites en matière de dégradation de la perte, qui dépendent directement du nombre de bits utilisés pour représenter les poids du *NN* et pour effectuer les opérations requises lors de la passe avant et arrière. Enfin, nous validons empiriquement la exactitude de nos résultats en minimisant des fonctions artificiellement perturbées ainsi qu'en entraînant des architectures de NN élémentaires sur l'ensemble de données MNIST.

Dans un dernier temps, nous mettons en lumière les avantages de l'élagage dans le domaine de l'Apprentissage par Contraintes (*CL*). Un aspect novateur de notre travail est la démonstration empirique que l'application de l'élagage structuré à un modèle destiné à être intégré dans un Programme Entier Mixte (*MIP*) conduit à une réduction exponentielle du temps de résolution nécessaire. En utilisant la technique d'élagage que nous avons établie dans la première section de cette thèse, nous procédons à la compression de *NNs* à propagation avant puis nous abordons un traitement du problème de recherche d'exemples adversaires. Les résultats de cette application sont sans équivoque, une réduction drastique des temps de résolution est évidente. D'ailleurs, les résultats indiquent clairement que l'élagage ne détériore pas la qualité de la solution dans notre contexte spécifique.

ABSTRACT

In numerous scenarios requiring data-driven predictions, Machine Learning (*ML*) models consistently achieve the State Of The Art (*SOTA*) performances. Particularly noteworthy are Neural Networks (*NNs*), which have asserted their dominance in recent years across diverse domains such as Computer Vision and Natural Language Processing. The prediction accuracy of novel *NN* models improves annually, though accompanied by the trade-off of larger architectures. This results in augmented memory storage and computational power requirements for model training and execution. The substantial parameter count in recent *NN* architectures has necessitated the development of Model Compression (*MC*) techniques. These techniques aim to reduce the resource demands of a model while maintaining its performance capabilities. In this context, leveraging Operations Research (*OR*) techniques appears highly suitable. The inherent high-level problem formulation involves minimizing the resources required by a model while preserving its accuracy within a closely defined range. This naturally paves the way for various potential *OR* approaches to articulate and address this challenge mathematically. Furthermore, a significant portion of the existing techniques within the *MC* literature relies on heuristic reasoning or simplistic concepts, often lacking robust theoretical foundations. Additionally, there exists a limited comprehension of the underlying mechanisms responsible for the efficacy of certain *MC* algorithms and their operational principles.

The primary objective of this thesis is to formulate and develop *MC* methodologies that leverage *OR* techniques, rooted in robust theoretical foundations, and that contribute towards unraveling the dynamics governing this domain.

First, we introduce the Structured Perspective Regularization (*SPR*), a novel regularization term designed for *Pruning NNs*. We commence by presenting a mathematical representation of the pruning problem, followed by a stronger reformulation achieved through the perspective function. Through algebraic simplification, we ultimately derive the conclusive expression for the *SPR*. This term, when incorporated into the loss function during training, facilitates the attainment of sparse architectures. We empirically validate the effectiveness of this technique through its application to filter pruning, using the most common Convolutional architectures and CIFAR and Imagenet datasets.

Next, we shift our focus to the realm of *Quantization*. This technique has seen a significant surge in adoption by practitioners in recent years, though its underlying principles remain elusive. We conduct a convergence analysis of *NN* training concerning low-bit floating-point

architectures. Our investigation yields the establishment of new bounds on loss degradation, which directly depends on the number of bits used to represent the *NN* weights and to perform the operations required in the forward and backward pass. Finally, we empirically validate the correctness of our results by minimizing artificially perturbed functions as well as training elementary *NN* architectures on the MNIST dataset.

Lastly, we shed light on the advantages of Pruning within the domain of Constraint Learning (*CL*). A pioneering aspect of our work is the empirical demonstration that applying *structured pruning* to a model intended for integration into a Mixed Integer Programming (*MIP*) model leads to a massive reduction in the necessary solving time. Employing the pruning technique we established in the initial chapter of this thesis, we proceed to compress feed-forward *NNs* before addressing an Adversarial Example Search problem. The outcomes of this application are unequivocal, a drastic reduction in solution times is evident. Furthermore, the results indicate that Pruning does not compromise solution quality within our specific context.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

CL	Constraints Learning
DL	Deep Learning
MC	Model Compression
MIP	Mixed Integer Programming
ML	Machine Learning
NN	Neural Network
OR	Operations Research
SOTA	State Of The Art
SPR	Structured Perspective Regularization

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Neural Networks (*NNs*) have sparked a revolutionary shift in the State Of The Art (*SOTA*) across diverse fields in recent years. This success is primarily attributed to their remarkable performance, made possible by a substantial increase in available data and the capacity to construct larger models, in turn, managed by dedicated computing architectures (GPUs). However, this progress comes at a cost: the growing model sizes escalate the demands on resources for both storage and execution.

Conversely, the prominence of *NNs* in edge computing is steadily escalating with each passing year. Consequently, the dimensions of these models have emerged as a critical limitation when it comes to storing and operating *NNs* on low-energy devices like smartphones, smartwatches, and wireless base stations [65, 96, 109].

One potential solution involves the direct design of smaller models. However, this endeavor proves challenging, particularly when the objective is to achieve competitive performance comparable to full-size networks. In numerous instances, a compact *NN* trained extensively over epochs achieves inferior performance compared to a larger counterpart trained for only a limited duration. This lends support to the claim by Li et al. [101] and Karnin [84] that the optimal approach for obtaining efficient yet high-performing models is to initially train very large *NNs* and subsequently reduce their size. In particular, Li et al. [101] perform extensive experimentation on Transformers models to support their thesis. Techniques that leverage the aforementioned principle are commonly referred to as Model Compression (*MC*) techniques. A plethora of *MC* methods have been developed, encompassing various approaches such as *Pruning* [16], *Quantization* [57], *Knowledge Distillation* [70], *Parameter Sharing* [123], *Neural Architecture Search* [69], and *Low-Rank Decomposition* [83].

A subset of these techniques, including Pruning, possess an inherent combinatorial essence, involving numerous binary decisions (e.g., retaining or discarding specific weights within the model). This makes the use of Mixed Integer Programming (*MIP*) tools particularly fitting within this domain. Nonetheless, the substantial magnitude of parameters within *NNs* introduces significant hurdles in terms of scalability for any approach that employs *MIP* to address the *MC* problem.

Furthermore, the integration of Machine Learning (*ML*) models in different contexts is gaining popularity. Once more, the size of the *ML* model being employed can significantly influence the overall scalability of the approach, as observed in cases like Constraints Learning (*CL*) [5, 15, 133]. This introduces fresh perspectives for *MC*, where the objective transcends mere

reduction of memory usage or energy consumption in the *ML* model. Instead, the focus shifts toward minimizing the influence on the computational complexity of the specific methodology into which this model is being integrated.

This thesis focuses on the development of new *MC* methods through the integration of Operational Research (*OR*) techniques. Specifically, we propose a pruning technique supported by a robust theoretical foundation. Additionally, we establish error bounds within the context of training quantized *NNs*. Furthermore, we demonstrate the effectiveness of Pruning when employed in a *CL* framework.

1.1 Research objectives

The primary objective of this thesis is to develop compression techniques that are grounded in theory while also providing theoretical support to existing methods through the utilization of *MIP* and other optimization techniques.

Our research endeavors to address the following key questions:

1. Can we formulate problems arising from *MC* techniques into *MIP* frameworks?
2. Can we effectively solve these models, or at least, generate solutions that can compete with current *SOTA* approaches?
3. What underlies the remarkable accuracy achieved by highly aggressive techniques such as Quantization?
4. To what extent can we compress a network before its performance starts to degrade?
5. Is it possible to apply *MC* methods within the realm of *CL*?
6. Does the implementation of *MC* techniques within *CL* contexts lead to a deterioration in the quality of the optimal solutions?

1.2 Contributions

Deep Neural Networks Pruning via the Structured Perspective Regularization (Chapter 5, [23]). Our initial contribution revolves around the development of a novel pruning methodology. We selected this particular *MC* technique due to its evident combinatorial nature, focusing on identifying parameters within the *NN* that can be eliminated without compromising the overall performance. We initiate the process by establishing a *MIP* model

to frame the problem. Subsequently, we strengthen this formulation using the Perspective function [39]. This process ultimately leads to the removal of certain model variables, resulting in the creation of a sophisticated regularization term named the Structured Perspective Regularization (*SPR*). Remarkably, the *SPR* term is versatile, capable of pruning any structures within the *NN*. Through an extensive series of experiments, we demonstrate that our method not only relies on a strong theoretical foundation but also emerges as a competitive contender against *SOTA* pruning techniques documented in the literature.

On the Convergence of Stochastic Gradient Descent in Low-precision Number Formats (Chapter 6, [24]). In our second contribution, we delve into the intricacies of quantization techniques. Quantization has gained attention in recent years due to its effectiveness in training *NNs* using low-bit Floating Points. Despite the cost reduction for model inference, training, and storage, the theoretical support for the progress of Quantization is not well understood. In this context, we make significant progress by establishing several bounds on convergence errors during the training of quantized *NN*, considering both deterministic and stochastic scenarios. We directly relate these bounds to the granularity of Quantization, i.e., the number of bits used. Additionally, we provide a practical tool for selecting the optimal learning rate to minimize theoretical error bounds. Our computational experiments validate the correctness of these bounds and show their accuracy in different cases.

Structured Pruning of Neural Networks for Constraints Learning (Chapter 7, [25]). In our final contribution, we showcase the application of *MC* techniques to enhance the scalability of *CL* approaches. Specifically, we employ the pruning technique we developed in Chapter 5 to achieve an impressive reduction in solution times for *MIP* models containing embedded *NNs*. To assess the effectiveness of this strategy, we address Adversarial Example Search Problems on the MNIST dataset [92] using feed-forward *NN* architectures. The outcome is two-fold: a significant reduction in solution times and a validation that Pruning maintains the quality of final solutions unharmed. This combination of efficiency improvement and solution quality preservation supports the viability of leveraging *MC* techniques to enhance the scalability of *CL* approaches.

CHAPTER 2 BACKGROUND

This chapter, presents the concepts, notations, and language that are necessary to understand the main body of the thesis. Besides the very fundamental concepts of Deep Learning (DL), we also give a brief introduction to the more specific concepts of Pruning, Quantization, and Constraint Learning.

2.1 Deep Learning

A Neural Network is "a graph of parameterizable layers (or "operators") that together implement a complex nonlinear function f" [71]. Often, the parameters that define the function $f : \mathcal{X} \times \mathbb{R}^n \to \mathbb{R}^m$ implemented by the *NN* are referred to as weights, we will use the notation $f(x; w)$ to indicate the output of the *NN* when the input is $x \in \mathcal{X}$ and the weights are $w \in \mathbb{R}^n,\ n \in \mathbb{N}$. The set $\mathcal{X}$ represents the input space, to not lose generality we do not make any assumptions about it. In this work, we focus on the scenario of supervised learning, where a labeled set of data (X, Y) is given and the goal is to find the best values for the parameters of the *NN* such that $f(x; w)$ is *close* to $y\ \forall (x, y) \in (X, Y)$. Here, the term *label* is used to denote the correct output or category assigned to input data. The structure of the *NN* graph, called architecture, is usually manually chosen and is unchanged throughout the whole process. Many different (learning) architectures have been proposed in the literature, but usually, all of them have an *input layer* (the one processing the input), an *output layer* (the one giving the final output of the model), and at least one *hidden layer* (a layer positioned between the first two). Finally, an operation known as the *activation* function is applied to the output of each layer before forwarding it as input to the subsequent layer. Different layers in the same *NN* can have different *activation* functions, with one of the most common being the ReLu function [58].

The *NN* architectures examined in this thesis encompass both Fully Connected Neural Networks, also referred to as Feed-Forward Neural Networks, and Convolutional Neural Networks. While a comprehensive overview of these architectures can be found in Wu [151] and Rosenblatt [130], we provide a brief notation recap here. In Fully Connected *NNs*, the fundamental components within each layer are denoted as *neurons*. In contrast, Convolutional *NNs* exhibit a distinct arrangement, where the weights in a single layer are organized into three-dimensional structures known as *filters*. Furthermore, each filter can be subdivided into two-dimensional units called *kernels*, each focusing on specific dimensions of the input. The collective set of kernel tensors spanning across filters that operate on the same pair of input

dimensions form what is referred to as a *channel*. The core operation in a Convolutional *NNs* is the convolution operation. This process involves iteratively computing the dot product between a kernel and a local region of the input image as the kernel slides across the same channel of the input image. The results are then summed for kernels belonging to the same filter, resulting in a 2-dimensional output. This cumulative process typically leads to the creation of a final 3-dimensional output, as multiple filters are often used.

To tune the weight values, the *NN* goes through a phase called *training*. To train an *NN* is necessary to choose a loss function $l : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, tipically differentiable, that penalizes the *difference* between $f(x; w)$ and the corresponding label y. In the training process, the goal is to solve a minimization problem in which the loss function serves as the objective function, and the *NN* parameters act as the variables to be optimized. That is

$$\min_{w \in \mathbb{R}^n} \sum_{(x,y) \in (X,Y)} l(f(x; w), y),$$

where n is the number of *NN* parameters. The optimization is carried out using a gradient-like algorithm ([79]), which we will discuss in the following. Different losses have been used in the literature with the most popular being the l_2 norm when the labels are continuous and the *Cross Entropy* loss ([31]) when the labels are discrete.

During training, *inference* is first performed, which consists in computing the output of the *NN* for one (or usually more) sample x and storing the output of each *NN* node. Then, the current loss $l(f(x; w), y)$ is computed. Finally, during the *back-propagation* phase, the gradient of the loss with respect to the weights is retrieved. Starting from the output layer, the gradient is computed by back-tracking the loss through the *NN* and applying the chain rule, so the value of the weights can be updated accordingly to minimize the loss. This whole process is named *training iteration* and it is repeated many times with different samples x until convergence. It is also common practice to not use just one sample for each training iteration but to instead sum the gradients of a *batch* of samples before updating the weight values. The set of iterations necessary to go through the whole dataset (X, Y) one time is referred to as an *epoch*.

A prevalent training algorithm widely used in machine learning is Stochastic Gradient Descent (*SGD*), which will also be pertinent in one of our contributions. *SGD* operates by iteratively updating model parameters in a direction that minimizes the loss function, aiming to achieve the optimal set of parameters for the given problem. In contrast to traditional Gradient Descent, which computes gradients over the entire dataset, SGD computes gradients using small random subsets (mini-batches) or individual samples of the data [20, 79]. This approach

injects randomness into the optimization process, enabling faster convergence and scalability to large datasets [18]. The ability of *SGD* to handle large-scale problems efficiently and to find optimal solutions has rendered it a cornerstone in training neural networks. A more detailed description of this procedure is provided in Algorithm 1.

Algorithm 1 Stochastic Gradient Descent algorithm

Initialize model parameters w
Set learning rate η
Set number of epochs N
Set mini-batch size h
for $epoch = 1, 2, \ldots, N$ **do**
  for each mini-batch $\{(x_i, y_i)\}_{i=1,..,h}$ of size h **do**
    Compute average loss: $\mathcal{L}(w) = \frac{1}{h} \sum_{i=1}^{h} l(f(x_i; w), y_i)$
    Compute gradient: $\nabla_w \mathcal{L}(w) = \frac{1}{h} \sum_{i=1}^{h} \nabla_w l(f(x_i; w), y_i)$
    Update parameters: $w \leftarrow w - \eta \nabla_w \mathcal{L}(w)$
  end for
end for

Usually, the labeled set (X, Y) is partitioned into three subsets. One, substantially bigger than the other two, is used in the above-mentioned training phase, and so it is called *training set*. The second one, referred to as *validation set*, is used if some parameters different from the weights, called hyper-parameters, need to be tuned. For instance, these hyper-parameters can be the step size (in this context called learning rate), the number of training epochs, or the batch size. The last subset, called *test set*, is used for the final performance evaluation of the model. The main goal of splitting the data in this way is to prevent *overfitting*, which happens when the model is extremely accurate on the training set but it fails to make good predictions on other samples. The ability of an *NN* to perform well on samples that have not been used during training is called model *generalization*.

2.1.1 Pruning

Loosely speaking, Pruning "requires finding the best compromise between removing some of the elements of the *NN* (weights, channels, filters, layers, blocks, ...) and the decrease in accuracy that this could bring" [62, 66, 84]. We refer to "removing elements" to indicate that the operations that were involving those parameters will no longer be performed and their results will be replaced by the most natural choice (e.g., by zero if the operation was a sum). Figure 2.1 shows an example of a dense and of a pruned *NN*, left and right, respectively.

Figure 2.1 Example of a dense network (left) and of a sparsified version of it (right). In the sparsified architecture, we can see that one neuron has been removed from the second hidden layer. Also, some single connections among unpruned neurons are missing between the input layer and the second one.

Prunable entities A relevant aspect of the process is the choice of the elements to be pruned. Given the prevailing reliance on GPUs for both *NN* training and inference, it is important to recognize that pruning individual weights might not significantly enhance the time required for these operations if other weights within the same *computational block* are preserved. This stems from the fact that GPUs operate through vector processing, and unstructured forms of sparsity might not align well with their capabilities, thus yielding limited efficiency gains ([65]). Therefore, in order to be effective, pruning has to be achieved simultaneously on all the weights of a given element, like a channel or a filter, so that the element can be deleted entirely. This kind of Pruning has been named in the literature as *structured pruning* as opposed to *unstructured pruning*, where single connections are removed. In Figure 2.2, we show some examples of structures that can be pruned in convolutional layers

Pruning after or before training Pruning can be performed while training or after training. The advantage of the latter is the ability to use standard training techniques unmodified, which may lead to better performances since they have already been extensively fine-tuned. On the other hand, pruning while training automatically adapts the values of the weights to the new architecture, dispensing with the need to re-train the pruned *NN*, or at least significantly reducing this requirement. In principle, it is also possible to prune before training, but, as discussed in Chapter 1, this is not a common practice. Figure 2.3 helps visualizing these different pruning schemes.

Re-training and regrowth Because pruning after and during training are the most common choices, this usually leads to models that are under-trained. This means that the *NN* attains significantly poorer performance than it could potentially achieve with different parameter values. To address this drawback, the *NN* is retrained for a few epochs and with

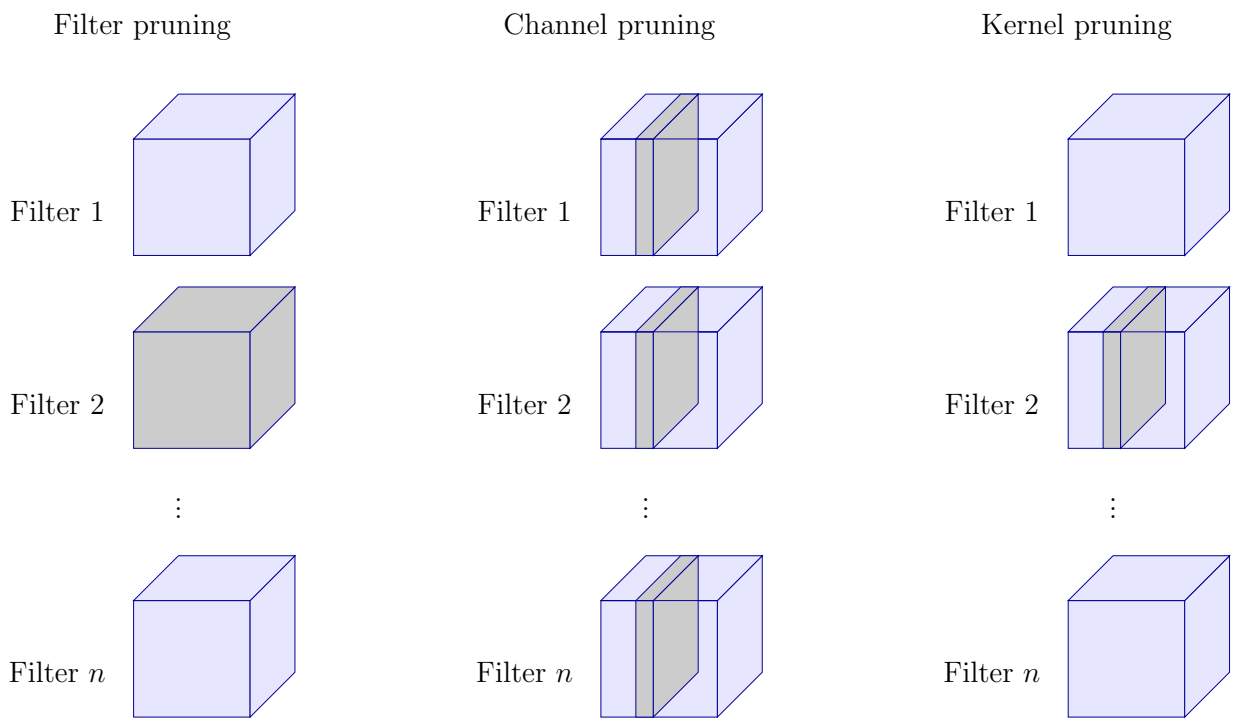


Figure 2.2 Different structures that can be pruned in a Convolutional *NN*. The areas in gray are the pruned ones.

a small learning rate. This practice, also called *fine-tuning*, has been shown to have significant benefits in terms of accuracy, thus leading to the development of pruning methods that alternate multiple pruning phases and training phases.

Another critical feature that can be implemented in pruning techniques is the ability to recover from early mistakes, i.e., from removing too many parameters or parameters that were very important to retain the model accuracy. This step, called *regrowth*, identifies removed parameters that may contribute to improving the model performances and restore them in the architecture.

2.1.2 Quantization

Until this moment, we considered the weights of a *NN* to be continuous parameters. To reduce the cost of storing and running these models, we can instead consider networks with weights that live in a discrete space. The *MC* techniques derived from this concept are called quantization methods and they are among the most effective strategies to compress *NNs*. In fact, Quantization is possible even before training, reducing drastically the cost of this extremely expensive phase.

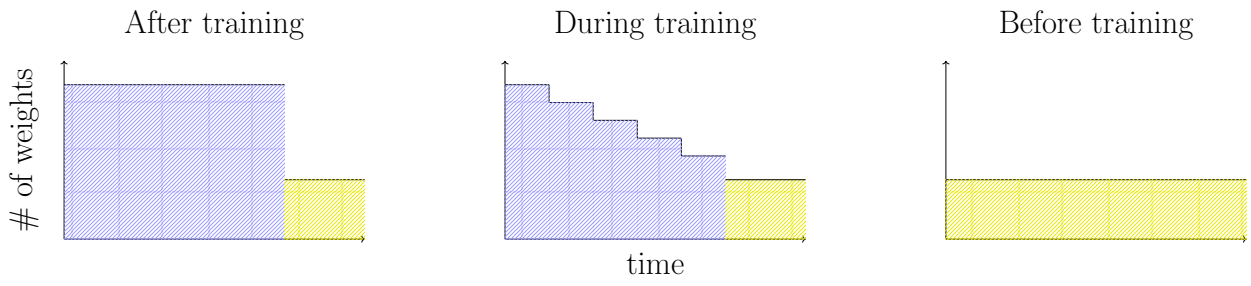


Figure 2.3 Respectively, from the left to the right, pruning after training, pruning during training, and pruning before training. The yellow regions correspond to time after training.

However, after a *NN* is quantized, the training can no longer be performed with the standard algorithm we presented in Section 2.1 since this algorithm relies on the continuity of the parameters to update them. So different training techniques that are specific to the particular quantization methodology need to be developed [76, 99].

An extremely promising method is based on the fact that weight values, layer outputs, and gradients are already stored as 32-bit floating-point numbers in the hardware, thus they already live in a discrete space. The set of the 32-bit floating-point values is very large, which is also the reason why the theory and algorithms developed for $\mathbb{R}$ usually generalize well in this space. The idea is to replace 32-bit floating points with lower precision numbers format (16-bit floating points, 8-bit integers, etc.), allowing the use of standard training algorithms and removing part of the implementation problematic [82, 150].

2.2 Constraint Learning

An important class of problems we will tackle in the final part of this work is the class of (Mixed-Integer) Nonlinear Programs with "learned constraints". These problems involve constraints containing functions, denoted as $g_i(x)$, where $i = 1, \ldots, k$. These functions are typically not directly computable, implying that their algebraic formulation is unknown. However, we have access to an extensive dataset comprising pairs $(\bar{y}, \bar{x})$ where $\bar{y} = g_i(\bar{x})$. Hence, we can construct a *ML* model approximating $\bar{g}_i(x)$ with an algebraic formulation

applicable to an optimization model.

$$\min_{x,y} cx + by \tag{2.1}$$

$$\text{s.t. } y_i = \bar{g}_i(x) \qquad i = 1, \ldots, k \tag{2.2}$$

$$Ax + By \leq d \tag{2.3}$$

$$x \in X \tag{2.4}$$

The primary challenge in (2.1)–(2.4) lies in the nature of $\bar{g}_i(x)$. The specific algebraic representation of $\bar{g}_i(x)$ significantly influences the fundamental properties of the problem. Moreover, selecting an *ML* model that leads to an algebraic formulation for $\bar{g}_i(x)$ requiring numerous variables and constraints can notably impact the computational time for solving such problems.

CHAPTER 3 CRITICAL LITERATURE REVIEW

This chapter provides a literature review encompassing the most pertinent works related to this thesis. Further references of interest will be provided within the main body of the work.

Pruning Among the initial strategies developed for parameter pruning, *magnitude-based pruning* emerged as a pioneering approach. This technique involves the removal of weights, post-training, based on their absolute values. Despite its apparent simplicity, this strategy has consistently demonstrated remarkable effectiveness. A notable instance can be found in the seminal work [64], which played a pivotal role in revitalizing interest in Pruning in recent times. In that work, an iterative scheme of magnitude-based pruning and subsequent retraining is notably successful.

This magnitude-based approach was further extended in Frankle and Carbin [51], where a retraining phase is introduced, with the peculiarity that the non-pruned weights are re-initialized to their original, pre-training, values.

In the context of structured pruning, similar methods can be tailored by utilizing the sum of absolute values of parameters associated with the same prunable entity. An illustrative instance is presented in Li et al. [97], where a magnitude-based approach is employed to prune filters within Convolutional Neural Networks.

Another straightforward approach is *pruning through regularization*, which entails augmenting the standard objective function with a term that penalizes less sparse structures. Once the regularization term is selected, the Neural Network can be trained using conventional techniques, rendering these methods highly accessible for implementation. A diverse array of regularization terms has been proposed in the literature, with the simplest being the ℓ_1 norm. This particular term has found application in various domains, such as feature selection ([19]).

Another prevalent technique involves polarization, where weights are coerced to assume values either in close proximity to zero or significantly distant from it [166].

In the realm of structured pruning, an alternative approach involves the inclusion of a scaling factor for each prunable entity within the network. This entails multiplying all pertinent parameters by their respective scaling factor. Sparsity is then enforced by penalizing the ℓ_1 norm of the scaling factors vector. An illustrative instance is provided in Ramakrishnan et al. [129], where a pruning mask is defined. This mask serves as a differentiable approximation of a *thresholding function*, effectively nudging the scaling factors towards 0 when they fall

below a predetermined threshold. This technique serves to mitigate potential numerical complications while maintaining the integrity of the pruning process.

When performing during or post-training pruning, the integration of first and second-order derivative information becomes feasible for determining the elements to be pruned. This approach is rooted in the concept that leveraging comprehensive information should yield improved performance [59, 121]. Moreover, since back-propagation is conducted during training, gradient information is inherently accessible without incurring additional computational.

The employment of a second-order expansion of the *NN* has been examined since the early stages of *MC* methodologies [66, 93]. This method is still applied in contemporary research, as demonstrated by recent works like [38, 140, 144, 158].

Quantization In theory, it is feasible to employ quantization methods that map *NN* weights to binary or ternary sets. Pioneering efforts, exemplified by works like Hubara et al. [76] and Li et al. [99], have achieved success in this endeavor by upholding the continuity of all computations during training. Notably, in Nia and Belbahri [122], binary quantization is enforced through a differentiable regularization term, while in Darabi et al. [34] a smooth proximal operator designed to approximate the sign function is introduced.

Other studies have successfully pursued the quantization of *NN* before training, employing sparse estimations for this purpose [54, 112, 129]. The adoption of lower-bit number formats directly within the *NN* implementation has also gained widespread popularity [82, 150]. In Bengio et al. [12], the introduction of a *straight-through estimator* aims to restore differentiability to quantized activation functions. Conversely, in Agustsson et al. [3] is presented a differentiable technique that progressively transitions from soft to hard quantizations. Lastly, the work [64] employs clustering methods for Quantization, methods that are subsequently merged with Pruning and Huffman encoding to yield impressive outcomes.

Operations Research in Model Compression Within the realm of *NNs*, *MIP* has demonstrated its efficacy in assessing model robustness against adversarial examples. In Fischetti and Jo [46], the authors present a *MIP* formulation specifically tailored for the output of a Neural Network featuring ReLU activation functions. This model's versatility extends to applications like feature visualization and the creation of adversarial examples. The work in Anderson et al. [5] introduces an alternative approach to model ReLU and max-pooling functions. In certain instances, this novel perspective culminates in optimal formulations for specific *NN* architectures. These contributions collectively showcase the potential of *MIP* techniques in enhancing our understanding of model behavior and vulnerabilities.

The integration of *MIP* within the realm of *MC* remains relatively unexplored. An exception is represented by ElAraby et al. [41], wherein a score function is introduced to evaluate the significance of neurons. Subsequently, *MIP* is utilized to minimize the count of crucial neurons requiring retention at each layer while preserving a substantial portion of the model's accuracy. This example once again showcases the potential of *MIP* techniques to optimize and streamline *MC* strategies.

In Serra et al. [134], the identification of stable neurons through *MIP* is used for structured pruning following [145], including operations for removing and merging neurons, as well as for folding layers of the neural network. Later on, the same authors improved their methodology by drastically reducing the time needed to identify stable neurons [135].

Likewise, in Aghasi et al. [1], a pruning methodology is devised where a Convex Linear Program is solved for each layer. This process enables the identification of prunable entities while maintaining the congruence of inputs and outputs with the original layer configuration.

CHAPTER 4 ORGANIZATION OF THE THESIS

The remainder of the document is organized as follows: The three contributions highlighted earlier are sequentially presented in Chapters 5 to 7. Chapter 8 is dedicated to a comprehensive analysis of our contributions as a collective entity. The emphasis is placed on identifying and exploring the interconnections and similarities that bind these contributions together. Finally, we discuss our overall conclusions and ideas for future directions in Chapter 9.

CHAPTER 5 ARTICLE 1: DEEP NEURAL NETWORKS PRUNING VIA THE STRUCTURED PERSPECTIVE REGULARIZATION

Authors: Matteo Cacciola, Antonio Frangioni, Xinlin Li, and Andrea Lodi.
To appear in SIAM Journal on Mathematics of Data Science (22/09/2023)[1].

Abstract In Machine Learning, Artificial Neural Networks (ANNs) are a very powerful tool, broadly used in many applications. Often, the selected (deep) architectures include many layers, and therefore a large amount of parameters, which makes training, storage and inference expensive. This motivated a stream of research about compressing the original networks into smaller ones without excessively sacrificing performances. Among the many proposed compression approaches, one of the most popular is *pruning*, whereby entire elements of the ANN (links, nodes, channels, ...) and the corresponding weights are deleted. Since the nature of the problem is inherently combinatorial (what elements to prune and what not), we propose a new pruning method based on Operational Research tools. We start from a natural Mixed-Integer-Programming model for the problem, and we use the Perspective Reformulation technique to strengthen its continuous relaxation. Projecting away the indicator variables from this reformulation yields a new regularization term, which we call the Structured Perspective Regularization, that leads to structured pruning of the initial architecture. We test our method on some ResNet architectures applied to CIFAR-10, CIFAR-100 and ImageNet datasets, obtaining competitive performances w.r.t. the state of the art for structured pruning.

5.1 Introduction

The striking practical success of Artificial Neural Networks (ANN) has been initially driven by the ability of adding more and more parameters to the models, which has led to vastly increased accuracy. This brute-force approach, however, has numerous drawbacks: besides the ever-present risk of overfitting, massive models are costly to store and run. This clashes with the ever increasing push towards edge computing of ANN, whereby neural models have to be run on low power devices such as smart phones, smart watches, and wireless base stations [65, 96, 109]. While one may just resort to smaller models, the fact that a large model trained even for a few epochs performs better than smaller ones trained for much longer lends credence to the claim [84] that the best strategy is to initially train large and

[1] A pre-print is available in [23].

over-parameterized models and then shrink them through techniques such as pruning and low-bit quantization.

Loosely speaking, pruning requires finding the best compromise between removing some of the elements of the ANN (weights, channels, filters, layers, blocks, ...) and the decrease in accuracy that this could bring [62, 66, 84]. Pruning can be performed while training or after training. The advantage of the latter is the ability of using standard training techniques unmodified, which may lead to better peformances. On the other hand, pruning while training automatically adapts the values of the weights to the new architecture, dispensing with the need to re-train the pruned ANN.

A relevant aspect of the process is the choice of the elements to be pruned. Owing to the fact that both ANN training and inference is nowadays mostly GPU-based, pruning an individual weight may yield little to no benefit in case other weights in the same "computational block" are retained, as the vector processing nature of GPUs may not be able to exploit un-structured forms of sparsity. Therefore, in order to be effective pruning has to be achieved simultaneously on all the weights of a given element, like a channel or a filter, so that the element can be deleted entirely. The choice of the elements to be pruned therefore depends on the target ANN architecture, an issue that has not been very clearly discussed in the literature so far. This motivates a specific feature of our development whereby we allow to arbitrarily partition the weight vector and measure the sparsity in terms of the number of partitions that are eliminated, as opposed to just the number of weights.

In this work, we develop a novel method to perform structured pruning during training through the introduction of a Structured Perspective Regularization (SPR) term. More specifically, we start from a natural exact Mixed-Integer Programming (MIP) model of the sparsity-aware training problem where we consider, in addition to the loss and ℓ_2 regularization, also the ℓ_0 norm of the structured set of weights. A novel application of the Perspective Reformulation technique leads to a tighter continuous relaxation of the original MIP model and ultimately to the definition of the SPR term. Our approach is therefore principled, being grounded on an exact model rather than based on heuristic score functions to decide what entities to prune as prevalent in the literature so far. It is also flexible as it can be adapted to any kind of structured pruning, provided that the prunable entities are known before the training starts, and the final expected amount of pruning is controlled by the hyper-parameter providing the weight of the ℓ_0 term in the original MIP model. While our approach currently only solves a relaxation of the integer problem, it would clearly be possible to exploit established Operations Research techniques to improve on the quality of the solution, and therefore of the pruning. Yet, the experimental results show that our approach

is already competitive with, and often significantly better than, the state of the art. Furthermore, since we perform pruning during training by just changing the regularization term, our approach can use standard training techniques and its cost is not significantly higher than the usual training without sparsification.

5.2 Related works

The field of pruning is experiencing a growing interest in the Machine Learning (ML) community, starting from the seminal work [64] that obtained unexpectedly good results from a trivial magnitude-based approach. The same magnitude-based approach was extended in [51] with a re-training phase where the non-pruned weights are re-initialized to their starting values. Moreover, in [108] the authors claim that, for most pruning methods, the most important result is the final structure of the pruned ANN, while the final values of the weights or their original initialization are not crucial.

A multitude of pruning approaches has been developed over the years, including but not limited to Bayesian methods [11, 110, 118, 165], regularization methods [104, 153], and combinations of pruning with other compression techniques [4, 52, 113]. Part of the literature [10, 28, 63, 158] focuses on pruning without modifying the model outputs or at least trying to minimize the output change. This approach can be effective when the model is highly over-parameterized or when very few parameters need to be pruned, but it is sub-optimal otherwise.

Another possibility is adding to the network parameters a scaling factor for each prunable entity, multiplying all the corresponding parameters; then, sparsity is enforced by adding the ℓ_1 norm of the scaling factors vector, as done for example in [107]. In [129] a pruning mask is defined, i.e., a differentiable approximation of a thresholding function that pushes the scaling factors to 0 when they are lower than a fixed threshold, avoiding numerical issues. Other methods that use a similar approach are [105, 111, 152].

Most recently-published state-of-the-art pruning methods either use a magnitude-based approach to identify prunable parameters [29, 30, 55, 91, 100, 113, 149, 163], or try to estimate the impact of a parameter removal [35, 37, 53, 94, 95, 119, 120, 126, 136, 156, 157]. In both cases, they rely on heuristic rules to compute the *importance* of an element of the ANN, mostly based just on its l_2 norm. This is arguably sub-optimal in general, and we aim at improving on this by using a principled approach. The need for a more theoretically grounded approach has been clearly been felt already, as proven by the proposals [26, 111, 118, 161] that, like ours, start from an exact theoretical model of the pruning problem formulated

through the l_0 norm. A significant difference, that has a profound impact on the developed technique, is that all these previous proposals do not focus on structured pruning.

Elsewhere, MIP techniques have been successfully used in the ANN context, but mostly in applications unrelated to pruning, such as the construction of adversarial examples (with fixed weights) [46]. In [5], the approach is extended to a larger class of activation functions and stronger formulations are defined. An exception is [41], where a score function is defined to assess the importance of a neuron and then a MIP is used to minimize the number of neurons that need to be kept at each layer to avoid large accuracy drops. In [134] a MIP is used first to derive bounds on the output of each neuron, which is then used in another MIP model of the entire network to find equivalent networks, local approximations, and global linear approximations with fewer neurons of the original network. Since MIPs are $\mathcal{NP}$-hard, these techniques may have difficulties scaling to large ANNs. Indeed, the pruning method developed in [1, 2] rather solves a simpler convex program for each layer to identify prunable entities in such a way that the inputs and outputs of the layer are still consistent with the original one. This layer-wise approach does not take into account the whole network at once as our own does.

The link between Perspective Reformulation techniques and sparsification has been previously recognized [6, 36], but typically in the context of regression problems that are much simpler than ANNs. In particular, all the above papers count (the equivalent of) each weight individually, and therefore they do not consider structured pruning of sets of related weights as it is required for ANNs. Furthermore, the sparsification approach is applied to input variables selection in settings that typically have orders of magnitude fewer elements to be sparsified than the present one.

5.3 Mathematical model

We are given a dataset X, an ANN model architecture whose set of parameters $W = \{ w_j \mid j \in I \}$ includes *prunable entities*, that is, disjoint subsets $\{ W_i = [w_j]_{j \in E_i} \}_{i \in N}$ for disjoint subsets of indices $\{ E_i \}_{i \in N}$ s.t. $I \supseteq \cup_{i \in N} E_i$, and a loss function $L(\cdot)$. If the value of a parameter w_j is zero it could be eliminated from the model (pruned) but, for the reasons discussed above, we are only interested in pruning the entities E_i, which corresponds to $w_j = 0$ for all $j \in E_i$. We therefore face a three-objective optimization problem which aims at: i) minimize the loss, ii) minimize some standard regularization term aiming a improving the model's generalization capabilities, and iii) maximize the number of pruned entities E_i. As customary in this setting, we approach this by scaling the three objective functions by means of hyperparameters whose optimal values are found by standard grid-search techniques. Employing

the usual ℓ_2 regularization, the problem can be cast as the MIP

$$\min_{W,y} L(X,W) + \lambda[\,\alpha\|\,W\,\|_2^2 + (1-\alpha)\textstyle\sum_{i\in N} y_i\,] \tag{5.1}$$

$$-My_i \le w_j \le My_i \qquad w_j \in E_i \quad i \in N \tag{5.2}$$

$$y_i \in \{0,1\} \qquad i \in N \tag{5.3}$$

where $\alpha \in [\,0\,,\,1\,]$ and $\lambda > 0$ are scalar hyper-parameters while M is an upper bound on the absolute value of the parameters. The binary variable y_i is 0 if the corresponding prunable entity is pruned, 1 if it is not. The standard "big-M" constraints (5.2) ensure that if $y_i = 0$ then $0 \le w_j \le 0$ for all parameters in the entity E_i, while if $y_i = 1$ the parameters can take any possible useful value (since M is an upper bound). Hence, the term "$\sum_{i\in N} y_i$" in the objective (5.1) represents the ℓ_0 norm of the structured set of weights. In the unstructured case, i.e., when each E_i is a singleton, the standard sparsification approach is to substitute the ℓ_0 norm with the ℓ_1 one; this allows to do away with the y_i variables entirely, replacing the corresponding term in the objective with $\|\,W\,\|_1$. This *elastic net regularization* [170] combines the properties of the ridge/Tikhonov (ℓ_2) and Lasso (ℓ_1) regularizations; it has also been extended to different forms, like the *Huber regularization* [77, 124] where the ℓ_2 and ℓ_1 norms, rather than being summed, are applied to different subsets of the space. The choice of the ℓ_1 norm is motivated by it being the best possible convex approximation of the nonconvex (and not even continuous) ℓ_0 one. However, these arguments do not readily carry over to the structured case.

5.3.1 The Perspective Reformulation

Basically all known strategies to solve MIPs like (5.1)–(5.3), be them exact or heuristic, start from considering its *continuous relaxation* whereby (5.3) is relaxed to $y_i \in [\,0\,,\,1\,]$. Such a problem is significantly easier than the original MIP, in the sense that a locally optimal solution $(\bar{w}, \bar{y})$ is efficiently obtainable using standard techniques for ANN training. However, it is well-known that such a solution can be rather different from the optimal solution (w^*, y^*) of (5.1)–(5.3), in both the y and w variables, due to the rather crude approximation of the nonconvex constraints (5.3) by means of their convex counterpart $y_i \in [\,0\,,\,1\,]$. This would hold even if the $(\bar{w}, \bar{y})$ were globally optimal, which happens, e.g., if $L(X,\cdot)$ is convex (not typical in the ANN context), save in the fortunate case where $\bar{w}$ happens to satisfy (5.3). Since $\bar{w}$ is typically what one could use to decide what entities to remove, this could lead to inefficient prunings. We therefore we seek a different relaxation that can provide us with higher quality solutions. In principle, an "exact" convex relaxation exists, which is obtained

by constructing the *convex envelope* of the objective function (5.1) on the set of integer solutions, i.e., its best possible convex approximation (technically, the convex function with smallest epigraph containing that of the original function). However, constructing the convex envelope of a function is in general $\mathcal{NP}$-hard, even in much less demanding settings than (5.1)–(5.3). A strategy that has proved successful is to devise convex envelope formulæ of fragments of the problems with specific structure; while the combination of these is typically not equivalent to the true convex envelope, it is often a much better approximation, leading to much better continuous relaxation solutions and therefore more efficient computational approaches. We can rewrite (5.1)-(5.3) as the following unconstrained optimization problem,

$$\min_{W,y} \Big\{ L(X,W) + \lambda [\, \textstyle\sum_{i \in N} h_i(W_i, y_i) \,] \Big\},$$

where

$$h_i(W_i, y_i) = \begin{cases} 0 & \text{if } y_i = 0 \text{ and } w_j = 0 \ \ \forall j \in E_i \\ \alpha \sum_{j \in E_i} w_j^2 + (1-\alpha) & \text{if } y_i = 1 \text{ and } |w_j| \leq M \ \forall \ j \in E_i \\ +\infty & \text{otherwise.} \end{cases}$$

The (clearly, nonconvex) function $h_i(\cdot, \cdot)$ belongs to a class of functions whose convex envelope can be explicitly computed: following [50], the convex envelope of h_i can be proven to be

$$\hat{h}_i(W_i, y_i) = \begin{cases} 0 & \text{if } y_i = 0 \text{ and } w_j = 0 \ \ \forall j \in E_i \\ \alpha \sum_{j \in E_i} \frac{w_j^2}{y_i} + (1-\alpha) y_i & \text{if } |w_j| \leq y_i M \ \forall j \in E_i \text{ and } y_i \in (0,1] \\ +\infty & \text{otherwise.} \end{cases}$$

This leads to the new formulation of problem (5.1)–(5.3)

$$\min_{W,y} \Big\{ \, L(X,W) + \lambda \textstyle\sum_{i \in N} \Big[\, \alpha \sum_{j \in E_i} \frac{w_j^2}{y_i} + (1-\alpha) y_i \, \Big] \ : \ (5.2) \ , \ (5.3) \Big\} \tag{5.4}$$

known in the literature as *Perspective Reformulation* (PR), that is easily seen to have the same integer optimal solution (w^*, y^*) as the original problem but a continuous relaxation (the *Perspective Relaxation*) that is "better" in a well-defined mathematical sense: its optimal objective value is (much) closer to the true optimal value of (5.1)–(5.3), which typically implies that its optimal solution $(\bar{w}, \bar{y})$ is more similar to the true optimal solution (w^*, y^*). Indeed, $\hat{h}_i(W_i, y_i)$ can be seen to have larger value than $h_i(W_i, y_i)$, the more so the more y_i is close to 0.5, i.e., "farther from being integer" [50], thereby discouraging highly fractional values in y^*. This has been already shown to leading to much better performances of both exact and heuristic approaches, w.r.t. using the standard continuous relaxation, for other

MIPs with similar structure.

5.3.2 Eliminating the y variables

While one can expect that the solution $(\bar{w}, \bar{y})$ of the Perspective Relaxation can provide a better guide to the pruning procedure, the presence of the explicit variables y makes it more difficult to apply standard training techniques to obtain it. Following the lead of [47, 48], we proceed at simplifying the PR model by projecting away the y variables. This amounts to computing a closed formula $\tilde{y}(w)$ for the optimal value of the y variables in the continuous relaxation of (5.4) assuming that w are fixed: the problem then decomposes over the E_i subsets, and therefore we only need to consider each fragment

$$f_i(W_i, y_i) = \lambda\Big[\, \alpha \textstyle\sum_{j \in E_i} w_j^2 / y_i + (1-\alpha) y_i \,\Big]$$

separately. Since f_i is convex in y_i if $y_i > 0$, we just need to find the root of the derivative

$$\frac{\partial f_i(W_i, y_i)}{\partial y_i} = \lambda \left[-\alpha \sum_{w_j \in E_i} \frac{w_j^2}{y_i^2} + (1-\alpha) \right] = 0 \ \ ,$$

that is

$$y_i = \sqrt{\frac{\alpha \sum_{w_j \in E_i} w_j^2}{1-\alpha}}$$

(we are only interested in positive y), and then project it on the domain. Note that, technically, $f_i(W_i, y_i)$ is nondifferentiable for $y_i = 0$ but that value is only achieved when $W_j = 0$, in which case the choice is obviously optimal. The constraints that defines the domain of y_i can be rewritten as $y_i \geq |\, w_j \,|/M$ for all $j \in E_i$, together with $y_i \in [0, 1]$; putting everything together, we obtain

$$\tilde{y}_i(w) = \min \left\{ \ \max \left\{ \{\, |w_j|/M \ : \ j \in E_i \}\, , \sqrt{\alpha \textstyle\sum_{j \in E_i} w_j^2/(1-\alpha)} \right\}, \, 1 \right\} \tag{5.5}$$

where we note that we do not need to enforce positivity since all the quantities are positive. Replacing y_i with $\tilde{y}_i(W_i)$ in the objective function of (5.4) we can rewrite the continuous relaxation of (5.4) as

$$\min_W \Big\{ \, L(X, W) + \lambda \textstyle\sum_{i=1}^N z_i(W_i; \alpha, M) \, \Big\}, \tag{5.6}$$

where

$$
\begin{aligned}
z_i(W_i;\alpha,M) &= \begin{cases}
\alpha \sum_{j\in E_i} \frac{\sqrt{(1-\alpha)}w_j^2}{\sqrt{\alpha \sum_{j\in E_i} w_j^2}} + (1-\alpha)\sqrt{\frac{\alpha \sum_{j\in E_i} w_j^2}{(1-\alpha)}} & \text{if } \frac{\|W_i\|_\infty}{M} \leq \sqrt{\frac{\alpha \sum_{j\in E_i} w_j^2}{1-\alpha}} \leq 1 \\
\alpha \sum_{j\in E_i} \frac{w_j^2 M}{\|W_i\|_\infty} + (1-\alpha)\frac{\|W_i\|_\infty}{M} & \text{if } \sqrt{\frac{\alpha \sum_{j\in E_i} w_j^2}{1-\alpha}} \leq \frac{\|W_i\|_\infty}{M} \leq 1 \\
\alpha \sum_{j\in E_i} w_j^2 + (1-\alpha) & \text{otherwise,}
\end{cases} \\
&= \begin{cases}
\sqrt{(1-\alpha)\alpha}||W_i||_2 + \sqrt{(1-\alpha)\alpha}||W_i||_2 & \text{if } \frac{||W_i||_\infty}{M} \leq \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq 1 \\
\frac{\alpha M}{||W_i||_\infty}||W_i||_2^2 + (1-\alpha)\frac{||W_i||_\infty}{M} & \text{if } \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq \frac{||W_i||_\infty}{M} \leq 1 \\
\alpha||W_i||_2^2 + (1-\alpha) & \text{otherwise,}
\end{cases} \\
&= \begin{cases}
2\sqrt{(1-\alpha)\alpha}||W_i||_2 & \text{if } \frac{||W_i||_\infty}{M} \leq \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq 1 \\
\frac{\alpha M}{||W_i||_\infty}||W_i||_2^2 + (1-\alpha)\frac{||W_i||_\infty}{M} & \text{if } \sqrt{\frac{\alpha}{1-\alpha}}||W_i||_2 \leq \frac{||W_i||_\infty}{M} \leq 1 \\
\alpha||W_i||_2^2 + (1-\alpha) & \text{otherwise.}
\end{cases}
\end{aligned} \tag{5.7}
$$

We call $z_i(W_i;\alpha,M)$ the *Structured Perspective Regularization* (SPR) w.r.t. the structure specified by the sets E_i. It is easily seen that the SPR behaves like the ordinary ℓ_2 regularization in parts of the space but it is significantly different in others. Due to being derived from (5.4), we can expect, all other things being equal, the SPR to promote sparsity—in terms of the sets E_j—better than the ℓ_2 norm. Indeed, SPR for $i \in I$ depends on the ℓ_∞ norm of W_i. This means that it penalizes entities on the ground of their maximum non-zero component, regardless to how many w_j have the maximum value. This arguably better promotes structured sparsity, as required by our application, w.r.t., say, using the ordinary ℓ_1 norm that rather promotes sparsity on each weight individually. This intuition is substantiated in the next S 5.3.3 where a more detailed discussion about the properties of the SPR regularizer can be found. Yet, all of the usual algorithms for training ANNs (SGD, Adam, etc.) can be employed for the solution of (5.6), which therefore should not, in principle, be more costly than non-sparsity-inducing training or unstructured sparsity-inducing terms like the ℓ_1 norm.

It is perhaps useful to remark that the Lasso/elastic net regularization can be seen as the application of an analogous process in the non-structured case. Indeed, assume W is fixed in (5.1)–(5.3): the optimal value of the y variables in the continuous relaxation of the problem solves (independently for each i)

$$\min\{\, (1-\alpha)y_i \,:\, (5.2)\,,\, y_i \in [\,0\,,\,1\,]\,\}$$

where the constraints are of course equivalent to $y_i \geq |w_i|/M$: hence, $y_i^* = |w_i|/M$, which

leads to the replacing of the ℓ_0 norm with the ℓ_1 one. Thus, our approach can be seen as a generalization of the standard one, but with two meaningful differences: i) it takes into account the effect of the quadratic regularization term, and ii) it applies the PR to the problem before doing the projection. Note that the first point is crucial to the second, because the PR of a linear function is easily seen to be the original function itself: in other words, the PR has no effect on linear problems. It is interesting to remark what happens to the SPR term in the context of unstructured pruning. In this case, the vector W_i in (5.7) is just a scalar, so $\|W_i\|_\infty = \|W_i\|_2 = |W_i|$ and the formula becomes much simpler. First, we only get two possible cases: if $1/M \leq \sqrt{\alpha/(1-\alpha)}$, then the second case is never possible; otherwise, it is the first case that never verifies. Moreover, both the first and the second cases of (5.7) become equal to the l_1 norm times a constant. This yield the known Berhu (reverse Huber) penalty [90], which has already been shown to be effective. However, doing this in the structured case is novel, and yields the SPR term that is significantly more complex than what was previously known, as better illustrated next.

5.3.3 Intuition on our new regularization term

We now provide a discussion on the shape of the SPR term, focussing on the features that could be linked to its better structured sparsification properties. We remark that, unlike what was done with the heuristic approaches in the literature, we did not develop the SPR in order to obtain such properties: instead, they were the natural results of constructing a better continuous approximation of the inherently combinatorial (and, therefore, hard) exact training-with-structured-sparsification problem (5.1)–(5.3).

First, we notice that SPR is not differentiable in zero. Since the gradient does not vanish in points close to the origin, this is known to increase the amount of parameters that are effectively zero after training is completed; indeed, this is the effect underlying the Lasso (l_1) regularizer for unstructured sparsity. This property is likely crucial, and in fact it is common to basically all other regularization-based approaches to structured sparsification, many of which use the non-squared ℓ_2 norm (also known as l_2/l_1 norm [29, 104]). Again, this feature was not planned, but it emerged as a result of our principled approach.

Out of 0, the behaviour of the SPR is different in different zones of the space. In particular, when the norm of a prunable entity is "large" (more precisely, when at least one among $\|W_i\|_\infty \geq M$ and $\|W_i\|_2 \geq \sqrt{(1-\alpha)/\alpha}$ holds, the white region of Figure 5.1), then SPR is equivalent to the standard ridge/Tikhonov (ℓ_2) regularization. Intuitively, the SPR identifies the entities that are "not likely" to be pruned, and, since no structured regularization needs to be applied there, the usual regularization is used which is still needed for generalization

purposes. This is similar to the (much simpler) Berhu regularization [90] (for the unstructured case) that coincides with the l_2 norm "far from 0", while rather being the (nondifferentiable) l_1 norm "close to 0". Again, we did not explicitly plan for this to happen, and such a behavior is not foreseen in the popular regularizers employed in the sparsification literature.

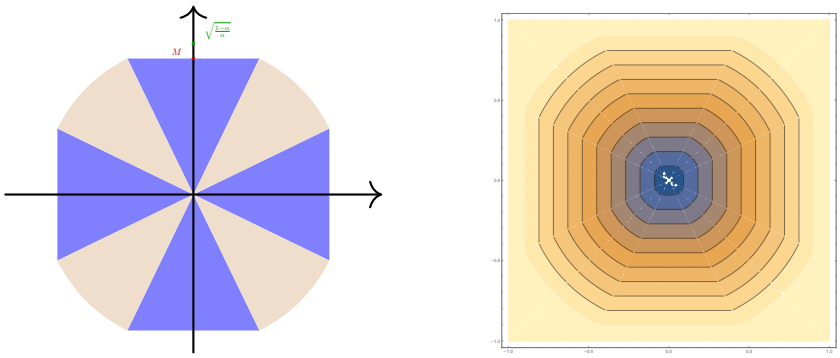
If an entity is "still within pruning range", SPR has a complex behavior organized around two different kinds of regions of the space. The first is the one in which a few parameters of an entity have disproportional larger absolute value compared to the others in the same entity (more precisely when $\|W_i\|_\infty \geq \sqrt{\alpha/(1-\alpha)}M\|W_i\|_2$, blue region of Figure 5.1). There the SPR is close to the infinity norm, and therefore the learning process focuses on reducing precisely the largest entries, since the infinity norm gradient is non-zero only in the entries corresponding to the coordinates in which the norm is reached (the ones with maximum absolute value). From a structured pruning point of view, entities with *unbalanced* parameters are not ideal since they may have many "small" (even possibly 0) weights, that therefore likely provide small (or null) benefit in terms of loss reduction, and yet they can not be removed due to a "few" large weights. The SPR identifies such entities and promotes the reduction of the disproportion among the weight magnitudes, possibly leading to the final removal.

In fact, when instead an entity has parameters with similar magnitudes (more precisely when $\|W_i\|_\infty \leq \sqrt{\alpha/(1-\alpha)}M\|W_i\|_2$, grey region of Figure 5.1), a sparse gradient could cause convergence speed problems. In this case, the SPR is equal to the (non-squared) l_2 norm whose gradient is not sparse; thus, the SPR promotes the simultaneous reduction of all the parameters, hopefully finally leading to the pruning of the entity.

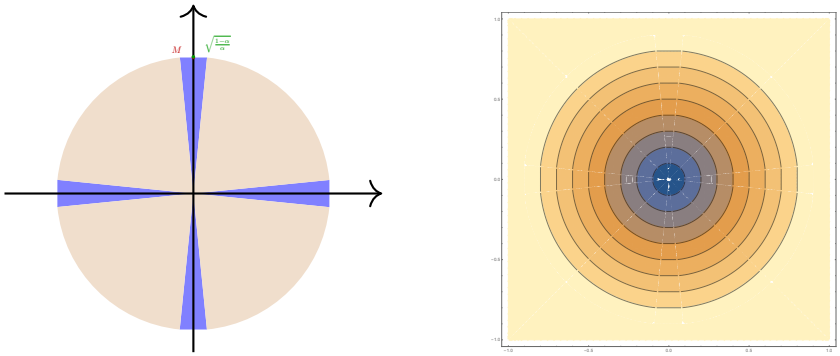
A pictorial representation of the previous discussion is provided in Figure 5.1 for a two-dimensional entity, with the left panels highlighting the regions where each case of the SPR occurs, while the right panels show the level sets of the SPR term that induces structured sparsity (that is, the term that multiplies $(1-\alpha)$ in (5.7)). Different plots corresponding to different choices of α (for fixed and M) are given to illustrate the complexity of the term as a function of its hyperparameters, and therefore its flexibility. A three-dimensional plot of the SPR term that induces structured sparsity is reported in Figure 5.2, illustrating how it transitions between different regions. Arguably, such a complex behaviour would have been rather complex to engineer; yet, it naturally emerged from our use of sophisticated mathematical optimization techniques.

5.3.4 Minor improvements

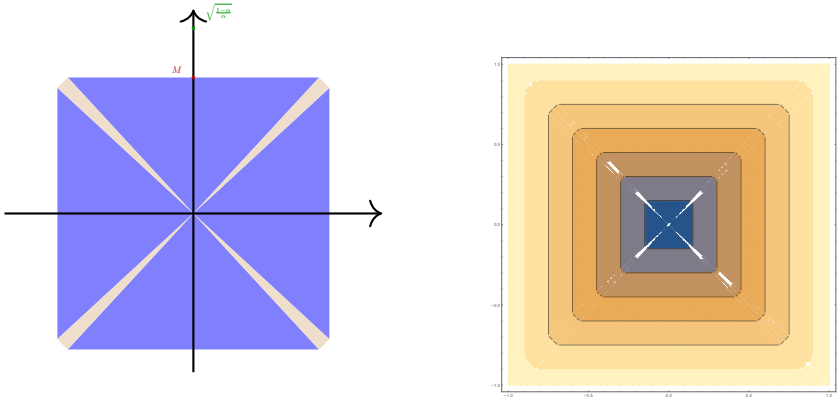
Remarkably, the SPR depends on the choice of M, which is, in principle, nontrivial. Indeed, all previous attempts of using PR techniques for promoting (non-structured, i.e., $E_i = \{\, i \,\}$)



(a) $M = 0.9$, $\alpha = 0.5$



(b) $M = 0.9$, $\alpha = 0.55$



(c) $M = 0.9$, $\alpha = 0.4$

Figure 5.1 Left, regions in which the SPR changes definition, right level sets of the structured sparsity term of the SPR

sparsity [6, 36] have been using the "abstract" nonlinear form $(1 - y_i)w_i = 0$ of (5.2). This still yields the same Perspective Reformulation, but it is not conducive to projecting away the y variables as required by our approach. While M could in principle be treated as another hyperparameter, in a (deep) ANN, different layers can have rather different optimal

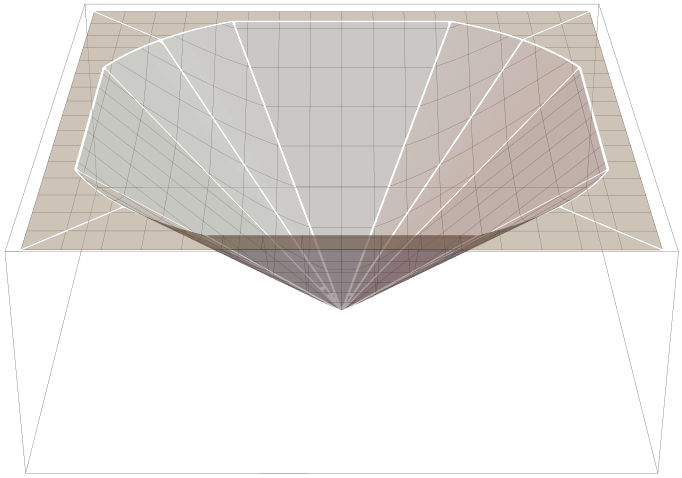


Figure 5.2 3-dimensional plot of the structured sparsity term of the SPR. When the norm of the entity is big enough, the term is constant. Otherwise, it is more similar to the l_2 or l_∞ norm, based on how are distributed the weights in the entity.

upper bounds on the weights; hence, using a single constant M for all the prunable entities is sub-optimal. The ideal choice would be to compute one constant M_i for each entity E_i; however, entities in the same layer are often similar to each other, so we only computed a different constant for each layer of the network, as detailed in S5.4.1, and used it for all entities belonging to that layer.

Furthermore, all the development so far has assumed that all prunable entities E_i are equally important. However, this may not be true, since different entities can have different number of parameters and therefore impact differently on the overall memory and computational cost. To take this feature into account, we modify our regularization terms as

$$\lambda \sum_{i \in N} \frac{u_i}{\sum_{i \in N} u_i} z_i(W_i; \alpha, M),$$

where u_i is the number of parameters belonging to entity E_i.

Finally, we perform a fine-tuning phase. After the ANN has been trained with the SPR, we prune all the entities W_i where 99.5% of the weights are smaller than the tolerance which is found using Algorithm 2. The threshold value 99.5% has been obtained through simple preliminary experiments. Though it could be treated as an hyperparameter and tuned accordingly, we did not deem this necessary since the experiments have shown that it plays a limited role in the final performances. We re-train the compressed network with the standard ℓ_2 regularization, starting from the value of the weights (for the non-pruned entities) obtained

at the end of the previous phase rather than re-initializing them.

Algorithm 2 performs a binary search in a given interval to find the highest possible pruning threshold that does not heavily affect the accuracy of the model. At each iteration, the candidate threshold is set to the medium point of the current interval, the ANN is pruned with such threshold and the new training accuracy is computed. If there was a drop in the accuracy larger than a given tolerance, the threshold is discarded and the first half of the interval becomes the interval for the next iteration. Otherwise, the threshold is accepted and the new interval is the second half of the current one. In our experiments we used $N = 10$, $a = 0$, $b =$ 1e-1 and $\delta =$ 5e-2.

Algorithm 2 Given a trained ANN with ρ^* training accuracy, the algorithm searches for the highest threshold in the interval $[a, b]$ such that the ANN compressed with such threshold does not lose more than δ accuracy.

Require: N, ρ^*, δ and $[a, b]$
  $\epsilon^* \leftarrow a$
  for $i = 1, \ldots, N$ **do**
    $\epsilon \leftarrow (a + b)/2$
    compress the network with the threshold ϵ and compute the current training accuracy ρ
    if $\rho \geq \rho^* - \delta$ **then** $a \leftarrow \epsilon^* \leftarrow \epsilon$
    else $b \leftarrow \epsilon$ **end if**
  end for
  return ϵ^*

5.4 Experiments

We tested our method on the task of filter pruning in Deep Convolutional Neural Networks; that is, the prunable entities are the filters of the convolutional layers. More specifically, the weights in a convolutional layer with n_{inp} input channels, n_{out} output channels and $k \times k$ kernels is a tensor with four dimensions (n_{inp}, n_{out}, k, k): our prunable entities correspond to the sub-tensors with the second coordinate fixed, and therefore have $n_{inp} \times k \times k$ parameters. Following [29], we include in the each prunable entity the corresponding bias and weight parameter belonging to the following batch normalization layer.

The code used to run the experiments was written starting from the public repository `https://github.com/akamaster/pytorch_resnet_cifar10` and `https://github.com/pytorch/examples/tree/master/imagenet`.

5.4.1 Datasets, architectures and general setup

For our experiments, we used 3 very popular datasets: CIFAR-10, CIFAR-100 [87] and ImageNet [88]. As architectures, we focused on ResNet [68] and Vgg [139]; in particular, we used ResNet-18, ResNet-20, Resnet50, ResNet-56 and Vgg-16 for the CIFAR 10 dataset, ResNet-20 for the Cifar-100 dataset and ResNet-18 for the ImageNet dataset. We chose these dataset-architecture pairs since they were among the most common in the literature.

For all the experiments, we used Pytorch (1.12.1) with Cuda, the CrossEntropyLoss and the SGD optimizer with 0.9 momentum. The M_i values were set as the maximum absolute values of the weights for each layer of a network with the same architecture but trained without our regularization term (for ResNet-20 and ResNet-56 we trained it, for ResNet-18 we used the pretrained version available from torchvision).

Additional details are provided in the appendix.

5.4.2 Results on CIFAR-10 and CIFAR-100

These experiments were performed on a single GPU, either a TESLA V100 32GB or NVIDIA Ampere A100 40GB. The model was trained for 300 epochs and then fine tuned for 200 ones. The dataset was normalized, then we performed data augmentation through random crop and horizontal flip. Mini batches of size 128 (64 for CIFAR-100) were used for training. The learning rate was initialized to either 0.1 or 0.01 and then it was divided by 10 at epochs 100 (200 for CIFAR-100), 250, 350, 400 and 450. We performed grid search on the crucial hyperparameters λ and α as detailed in S5.6.3.

Since the learning-with-structured-pruning problem is a multi-objective one, there is no overall best solution: rather, we report a representative selection of the non-dominated solutions on the efficient frontier (the best pruning corresponding to any achieved level of accuracy), together with the hyperparameters achieving it. An example of the pareto curve obtained through our experiments is reported in S5.6.4. We also report the number of floating point operations (FLOPs) necessary to perform inference for each model.

Table 5.1 shows the results of training ResNet-20 on CIFAR-10: we were able to prune more than 42% of the parameters by still increasing the accuracy of the original model, while we could prune more than 75% of the model by still preserving more than 90% accuracy. With the same architecture on the more challenging CIFAR-100 dataset (Table 5.2) we could prune more than 15% of parameters while improving the accuracy of the original model, but pruning many parameters resulted in a significant accuracy loss: we could still achieve more than 67% accuracy by pruning a few less than 40% of the parameters, but accuracy dropped

Table 5.1 Results of our algorithm on CIFAR-10 using ResNet-20

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.9	0.5	85.63	242424	(89.88)	12.70M	(31.31)
0.1	1.6	0.5	86.81	232409	(86.17)	13.77M	(33.96)
0.1	1.3	0.5	88.00	228094	(84.57)	16.62M	(40.99)
0.1	1.3	0.1	89.46	213958	(79.33)	15.01M	(37.02)
0.1	1.3	1e-4	90.03	203154	(75.32)	18.09M	(44.62)
0.1	0.8	0.1	91.22	172658	(64.01)	24.86M	(61.30)
0.1	0.5	1e-3	92.23	115620	(42.87)	29.69M	(73.23)
Original model			92.03	0	(0.00)	40.56M	(100.00)

Table 5.2 Results of our algorithm on CIFAR-100 using ResNet-20

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.10	0.50	0.50	65.64	160394	(58.20)	23.89M	(58.90)
0.01	1.30	0.50	67.53	102944	(37.36)	33.98M	(83.78)
0.10	0.30	0.60	68.22	79720	(28.93)	29.94M	(73.83)
0.10	0.30	0.15	68.57	61515	(22.32)	29.60M	(72.98)
0.01	1.25	0.15	69.13	42009	(15.24)	37.88M	(93.39)
Original model			68.55	0	(0.00)	40.56M	(100.00)

Table 5.3 Results of our algorithm on CIFAR-10 using ResNet-56

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.9	0.01	90.62	762869	(89.43)	30.10M	(23.99)
0.1	1.0	5e-3	91.85	726717	(85.19)	38.94M	(31.03)
0.1	0.7	0.01	92.42	677433	(79.42)	42.65M	(33.98)
0.1	0.4	0.10	92.76	612038	(71.75)	44.08M	(35.13)
0.1	0.4	0.50	93.48	553821	(64.92)	50.90M	(40.57)
0.1	0.2	0.50	93.96	395478	(46.36)	83.58M	(66.60)
Original model			93.35	0	(0.00)	125.48M	(100.00)

to less than 66% if pruning more.

Table 5.3 reports results on training the ResNet-56 architecture on CIFAR-10: once again pruning about 65% of the parameters improved accuracy and we could keep more than 92% accuracy while pruning almost 80% of the network.

Finally, Tables 5.4, 5.5 and 5.6 report results on the CIFAR-10 dataset of models Resnet-18, ResNet-50, and Vgg-16 (respectively), which have a much larger number of parameters than the previous ones: in these cases we were able to prune the vast majority of the parameters (from 89% to more than 90%) without really affecting the accuracy of the ANN, sometimes even increasing it.

5.4.3 Results on ImageNet

These experiments were performed on single TITAN V 8GB GPU. The model was trained for 150 epochs and fine tuned for 50 ones. The preprocessing was the same as for the CIFAR datasets. We used mini batches of 256 and 0.1 learning rate that was divided by 10 every 35 epochs, and the grid search detailed in S5.6.3. As usual for datasets with so many classes, we report also the top5 accuracy, i.e., the percentage of samples where the correct label was on the 5 higher scored classes by the model.

Table 5.4 Results of our algorithm on CIFAR-10 using ResNet-18

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	2.8	0.5	94.38	10691286	(95.29)	101.53M	(18.28)
0.1	2.5	0.5	94.50	10555810	(94.08)	118.51M	(21.33)
0.1	1.9	0.5	94.81	10451461	(93.15)	143.45M	(25.82)
0.1	1.3	0.5	95.34	10059742	(89.66)	192.39M	(34.64)
Original model			95.15	0	(0.00)	555.47M	(100.00)

Table 5.5 Results of our algorithm on CIFAR-10 using ResNet-50

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.6	1e-4	93.49	23197453	(97.86)	62.43M	(4.81)
0.1	1.6	1e-3	93.80	22977664	(96.93)	103.97M	(8.01)
0.1	1.3	1e-4	94.36	22931931	(96.74)	166.30M	(12.81)
0.1	1.0	1e-4	94.51	22745124	(95.95)	175.85M	(13.55)
0.1	1.0	0.5	94.96	21800173	(91.96)	258.10M	(19.88)
Original model			94.83	0	(0.00)	1.30B	(100.00)

Table 5.6 Results of our algorithm on CIFAR-10 using Vgg-16

L-rate	λ	α	Acc.	Pruned pars (%)		FLOPs (%)	
0.1	1.6	1e-4	93.44	14266694	(96.87)	82.00M	(26.18)
0.1	1.6	0.1	93.56	14179500	(96.27)	89.61M	(28.61)
0.1	1.0	0.5	93.93	13647661	(92.66)	126.74M	(40.47)
0.1	0.1	0.5	94.31	12044579	(81.78)	186.67M	(59.60)
Original model			94.12	0	(0.00)	313.20M	(100.00)

Table 5.7 Results of our algorithm on ImageNet using ResNet-18

L-rate	λ	α	top1	top5	Pruned pars (%)		FLOPs (%)	
0.1	0.75	0.1	70.26	89.66	1992131	(17.04)	2.20B	(92.84)
0.1	1.0	0.1	69.27	89.06	3811382	(32.61)	2.07B	(87.58)
0.1	1.1	0.1	68.87	88.72	4481715	(38.34)	2.03B	(85.57)
0.1	1.0	0.3	66.20	87.15	7406308	(63.36)	1.83B	(77.31)
Original model			69.76	89.08	0	(0.00)	2.37B	(100.00)

Results using ResNet-18 are reported in Table 5.7, and show that even in a very large and difficult dataset our method was able to improve the original model results while pruning more than 17% of the parameters, and basically tie with it while pruning 30% of the parameters.

Pruning almost 40% of the network caused a drop of only 0.5% in the accuracy, while a more consistent decrease resulted when we pruned about 60% of the parameters.

5.4.4 Comparison with state-of-the-art methods

In this section, we compare our results (denoted as SPR) with some of the state-of-the-art algorithms for structured pruning. We report results from [75] (denoted by SSS), [137] (denoted by EPFS), [147] (denoted by L2PF), [98] (denoted by PFFEC), [154] (denoted as RSNI), [103] (denoted as HRANK), [148] (denoted as PFC), [141] (denoted by CHIP), [89] (denoted as DNR), [29] (denoted as OTO), [100] (denoted as DHP), [156] (denoted as NISP), [168] (denoted as DCP), [142] (denoted as SCOP) , [102] (denoted as PFPE) and [42] (denoted by HFP).

Since not all the above papers reported the results for all our metrics (for example, some works only reported the percentage of parameters pruned), in some cases we had to do some conversions that naturally came with some mild approximation. Moreover, in [75], only plots were presented, so we had to approximately deduce the data from some points of the figures (Figure 2(a) and Figure 2(c) of [75], we denote the points as P1, P2, etc.). For ImageNet the top5 accuracy is not reported in [42], so we marked the corresponding field in our table with a “N/A". Finally, we report results for different settings of each method as they were given in the original papers; however, it should be remarked that not all of them are structured pruning methods as our own (in particular, pruning at the filter level), hence the results may not be completely equivalent, although in general they should be comparable.

Regarding ResNet-20 on CIFAR-10, our approach (shown in Table 5.8) outperforms all the other methods, meaning that we could reach equal or better accuracy while pruning a larger amount of parameters. For instance, L2PF achieved 89.9% accuracy with 73.96% sparsity, while we achieved higher sparsity (79.33%) and a little more accuracy (90.03%)

On CIFAR-100 using ResNet-20, the results in Table 5.9 clearly show that we outperform SSS, as we could achieve more than 68.5% accuracy while pruning more than 22% of parameters while SSS could prune only 14.81% to obtain a little bit more than 67% accuracy. In Table 5.10, we can observe a similar situation to ResNet-20 on CIFAR-10 for ResNet-56 on the same dataset. One of the few results we did not outperform was the CHIP 94.16 accuracy with 42.8% sparsity but we could obtain a little bit more sparsity (46.36%) with a comparable accuracy (93.96%).

The results reported in Tables 5.11 and 5.12 show that our approach is very competitive with respect to the very recent state-of-the-art methods such as OTO and DNR, sometimes

Table 5.8 Results of state of the art method on CIFAR-10 using ResNet-20

Method	Setting	Acc.	Pruned pars (%)	
SSS	P1	90.80	120000	(44.44)
	P2	91.60	40000	(14.81)
	P3	92.00	10000	(3.70)
	P4	92.50	0	(0.00)
EPFS	B-0.6	91.91	70000	(24.60)
	B-0.8	91.50	100000	(36.90)
	F-0,05	90.83	130000	(51.10)
	C-0.6-0.05	90.98	150000	(56.00)
L2PF	LW	89.90	199687	(73.96)
PFC	P1	90.55	135000	(50.00)
DHP	50	91.54	118327	(43.87)
SCOP	P1	90.75	151853	(56.30)
PFPE	P1	90.91	169035	(62.67)
RSNI	model A	90.9	104708	(38.82)
	model B	88.8	190800	(70.74)
SPR	λ 1.3 - α 0.1	90.03	213958	(79.33)
	λ 0.8 - α 0.1	91.22	172658	(64.01)
	λ 0.5 - α 1e-3	92.23	115620	(42.87)

Table 5.9 Results of state of the art method on CIFAR-100 using ResNet-20

Method	Setting	Acc.	Pruned pars (%)	
SSS	P1	65.50	120000	(44.44)
	P2	67.10	40000	(14.81)
	P3	68.10	10000	(3.70)
	P4	69.20	0	(0.00)
SPR	λ 0.5 - α 0.5	65.64	160394	(58.20)
	λ 0.3 - α 0.15	68.57	61515	(22.32)
	λ 1.25 - α 0.15	69.13	42009	(15.24)

Table 5.10 Results of state of the art method on CIFAR-10 using ResNet-56

Method	Setting	Acc.	Pruned pars (%)	
PFFEC	A	93.10	80000	(9.40)
	B	93.06	120000	(13.70)
EPFS	B-0.6	92.89	240000	(27.70)
	B-0.8	92.34	500000	(58.60)
	F-0.01	92.96	170000	(20.00)
	F-0.05	92.09	510000	(60.10)
	C-0.6-0.05	92.53	570000	(67.10)
HFP	0.5	93.30	425000	(50.00)
	0.7	92.31	608430	(71.58)
HRank	P1	90.72	580000	(68.10)
	P2	93.17	360000	(42.40)
	P3	93.52	140000	(16.80)
PFC	P1	93.05	425000	(50.00)
DHP	50	93.58	354685	(41.58)
	38	92.94	510958	(59.90)
SCOP	P1	93.64	480249	(56.30)
PFPE	P1	92.67	759015	(88.98)
CHIP	P1	92.05	600000	(71.80)
	P2	94.16	360000	(42.80)
NISP	P1	93.32	363386	(42.6)
DCP	P1	93.49	420014	(49.24)
	Adapt	93.81	599897	(70.33)
SPR	λ 0.7 - α 0.01	92.42	677433	(79.42)
	λ 0.4 - α 0.1	92.76	612038	(71.75)
	λ 0.4 - α 0.5	93.48	553821	(64.92)
	λ 0.2 - α 0.5	93.96	395478	(46.36)

being able to improve them significantly. For example, DNR can only prune less than 82% of ResNet-18 achieving 94.64% accuracy, while our method reach more than 95% accuracy pruning more than 89% of the network. The only result that is somehow stronger than SPR is that obtained by the Adaptive version of DCP, see the corresponding entries in Tables 5.10 and 5.13. However, the difference in performance is not large in all cases, which confirms that SPR is at least competitive with all the alternative approaches we could compare it to.

Similarly, when training Vgg-16 on Cifar-10, our method beats all the state-of-the-art ones but the Adaptive DCP. For example, CHIP can never prune more than 88% of the ANN but our algorithm prunes consistently more than 92% achieving similar or better accuracy (Table 5.13).

Table 5.11 Results of state of the art method on CIFAR-10 using ResNet-18

Method	Setting	Acc.	Pruned pars (%)	
DNR	P1	94.64	9233284	(82.36)
SPR	λ 1.3 - α 0.5	95.34	10059742	(89.66)
	λ 1.9 - α 0.5	94.81	10451461	(93.15)

Table 5.12 Results of state of the art method on CIFAR-10 using ResNet-50

Method	Setting	Acc.	Pruned pars (%)	
OTO	P1	94.40	21570653	(91.20)
SPR	λ 1.0 -α 0.5	94.96	21800173	(91.96)
	λ 1.3 -α 1e-4	94.36	22931931	(96.74)

On ImageNet using ResNet-18, the results in Table 5.14 show that even if our method does not outperform all the other ones, we were able to achieve very competitive results. Likely some additional parameter tuning could lead us to even more competitive results.

5.5 Conclusions and future directions

Based on an exact MIP model for the problem of training-with-structured-pruning of ANNs, we proposed a new regularization term, based on the projected Perspective Reformulation, designed to promote structured sparsity. The proposed method is able to prune any kind of structures, and the amount of pruning can be tuned by appropriate hyper-parameters. We tested our method on some classical datasets and architectures and we compared the results with some of the state-of-the-art structured pruning methods, proving that our method is competitive, and often outperforms existing ones.

These results are even more promising in view of the fact that further improvements should be possible. Indeed, we are currently solving the continuous relaxation of our proposed exact model, albeit a “tight” one due to the use of the Perspective Reformulation technique. By a tighter integration with other well-established MIP techniques, further improvements are foreseeable.

Table 5.13 Results of state of the art method on CIFAR-10 using Vgg-16

Method	Setting	Acc.	Pruned pars (%)	
PFC	P1	93.63	7357792	(50.00)
EPSF	F-0.005	94.67	10305584	(69.10)
	F-0.001	93.61	8225584	(56.70)
PFEEC	P1	93.40	9315584	(64.00)
HRANK	P1	93.43	12205584	(82.90)
	P2	92.34	12075584	(82.10)
	P3	91.23	12935584	(92.00)
CHIP	P1	93.86	11955584	(81.60)
	P2	93.72	12215584	(83.30)
	P3	93.18	12815584	(87.30)
DNR	P1	92.00	13560314	(92.07)
PFPE	P1	92.39	13891701	(94.32)
OTO	P1	93.30	13918211	(94.50)
DCP	P1	94.16	7057294	(47.92)
	Adapt	94.57	13782934	(93.58)
SPR	λ 1.6 - α 1e-4	93.44	14266694	(96.87)
	λ 1.6 - α 0.1	93.56	14179500	(96.27)
	λ 1.0 - α 0.5	93.93	13647661	(92.66)
	λ 0.1 - α 0.5	94.31	12044579	(81.78)

Table 5.14 Results of state-of-the-art method on ImageNet using ResNet-18

Method	Setting	top1	top5	Pruned pars (%)	
EPFS	F-0.05	67.81	88.37	3690000	(34.60)
HFP	0.20	69.15	N/A	2354869	(22.07)
	0.35	68.53	N/A	3976709	(37.27)
SCOP	A	69.18	88.89	4593978	(39.30)
	B	68.62	88.45	5084938	(43.50)
SPR	λ 0.75 - α 0.1	70.26	89.66	1992131	(17.04)
	λ 1.0 - α 0.1	69.27	89.06	3811382	(32.61)
	λ 1.1 - α 0.1	68.87	88.72	4481715	(38.34)

5.6 Appendix

5.6.1 SPR regularity

In the following, we prove that the SPR term defined in (5.7) is continuous, differentiable almost everywhere, and non-convex but quasi-convex. Continuity of the SPR could be established by proving equality of the limits of the distinct segments defined within (5.7) at the points where the function undergoes a change in its definition, but a more concise argument uses the fact that the definition (5.7) is equivalent to the composition of (5.4) with the optimal solution formula for the optimal w variables (5.5), which is easily seen to be a continuous function of w. Furthermore, while (5.4) would seem not to be continuous in zero, it is easy to see that

$$\lim_{y_i \to 0} \sum_{j \in E_i} \frac{w_j^2}{y_i} \leq \lim_{y_i \to 0} \sum_{j \in E_i} \frac{y_i^2 M^2}{y_i} = 0,$$

on feasible solutions (y_i, w_i), i.e., when (5.2) are satified. Thus, (5.4) can be continuously extended at zero, and therefore (5.7) is a composition of continuous functions and hence continuous itself.

The fact that the SPR term is differentiable almost everywhere comes from the differentiability (almost everywhere) of the functions that define (5.7) and from the fact that the set where the SPR changes definition has zero mass. However, the previous pictures clearly show that the function can indeed be nondifferentiable there. In particular, since both the l_1 norm and the l_∞ norm are not differentiable in zero, the SPR is not differentiable in zero, as expected from a sparsity-inducing regularization term. It is easy to see by drawing a few examples that the SPR is in general not convex. For an algebraic proof consider $\alpha = 0.65$, $M = 0.4$, $W_1 = (0.3, 0, .., 0)$ and $W_2 = (0.5, 0, .., 0)$; then $z(W_1) = 0.3405$, $z(W_2) = 0.5125$, $z(\frac{1}{2}W_1 + \frac{1}{2}W_2) = 0.4540$, $\frac{1}{2}z(W_1) + \frac{1}{2}z(W_2) = 0.4265$, where $z(\cdot)$ is defined in (5.7). We have just shown that $z(\frac{1}{2}W_1 + \frac{1}{2}W_2) > \frac{1}{2}z(W_1) + \frac{1}{2}z(W_2)$, i.e., that the SPR is not convex. Yet, the function defined in (5.5) is clearly quasi-convex and (5.4) is non-decreasing in the y variable, so the SPR is quasi-convex.

5.6.2 Time complexity study

During the first step of our method, in which the SPR term and its (sub)gradient have to be computed, an extra computational cost is incurred w.r.t. the standard "simple" regularizations; note that this does not happen during the fine-tuning phase, where the standard ridge/Tikhonov (ℓ_2) regularization is used instead. The impact of the SPR term is shown Table 5.15, which compares the cost per epoch with and without the SPR regularization.

For easier data sets (small input size), our regularization term roughly doubles the cost per epoch, while for the hardest data set (more relevant to real applications) the two costs are almost the same, which proves that our approach is, generally speaking, computationally viable.

Table 5.15 Average computation times (seconds) for one epoch with and without the SPR term

Architecture and data set	time SPR	time without SPR
ResNet-20 on CIFAR-10	13.05	6.51
ResNet-56 on CIFAR-10	36.58	16.99
ResNet-20 on CIFAR-100	22.99	11.26
ResNet-18 on ImageNet	2,433.14	2,401.05

5.6.3 Detail on grid search

As we stated in the first paragraph of Section 3, α and λ hyperparameters are found through adaptive grid search. We tested 36 pairs with $\lambda \in [0.1, 3.0]$ and $\alpha \in$ [1e-4, 0.6] for all the experiments with the Cifar-10 dataset. For the Cifar-100 experiments, the intervals for λ and α were kept the same and 70 pairs were tested. Finally, we used 12 pairs with $\lambda \in [0.5, 1.2]$ and $\alpha \in$ [1e-1, 0.6] for the experiments with the Imagenet dataset.

Table 5.16 Accuracy before and after the fine-tuning phase (ResNet-18 on CIFAR.10)

λ	α	Accuracy before	Accuracy after
1.1	0.01	82.40	85.56
1.7	0.30	85.28	87.33
1.1	0.30	88.22	89.47
0.5	0.30	90.62	91.23
0.2	0.30	92.46	92.69
Original model		92.03	-

Finally, we report an observation on the importance of the fine-tuning phase. From Table 5.16, we can see that this step is crucial when the pruning caused a significant accuracy drop, while is less relevant (as one could expect) when the accuracy remains high despite the pruning.

5.6.4 Pareto curve

In Figure 5.3 we plot all the accuracy-sparsity pairs obtained with our experiments using the ResNet-20 model on the Cifar-10 dataset. Although the curve is not fully complete, it gives a good insight on how pruning affect the accuracy of the model.

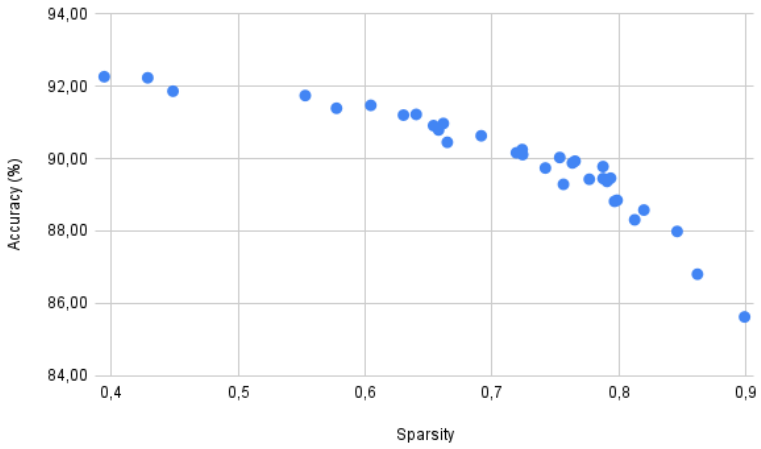


Figure 5.3 Pareto curve approximation for ResNet-20 on Cifar-10. Different points correspond to different values of α and λ.

5.6.5 Observation on the structure of the pruned network

From the experiments, we noticed that our algorithm heavily prunes the last layers of the network. This is due to the fact that the gain in sparsity is larger for these last layers, since their filters contain way more parameters than those belonging to the earliest layers. When the hyperparameters favor heavy pruning even at the cost of a consistent accuracy drop, or when the model is so over-parametrized that even pruning many of parameters only slightly affects the accuracy, basically all final layers are fully pruned. When, instead, less parameters are pruned then the final layers that are not fully pruned tend to be always the same for different configurations of the hyperparameters: for example, for ResNet-18 on ImageNet, the layer with the last residual connection is almost never pruned. This indicates that our pruning approach is successful in identifying the essential structures of the model that need be retained.

5.6.6 Results in the unstructured setting

As mentioned in the main body of this work, to effectively reduce the computational endeavor of GPU computations through pruning, it is necessary to remove entire structures of the network. However, we acknowledge that unstructured pruning retains its relevance in certain contexts and enables cleaner comparisons with other methods. Consequently, we have chosen to include results within the unstructured pruning setting to provide a comprehensive perspective, although it is important to note that the primary emphasis of this study lies in the structured pruning scenario.

When the prunable entities E_i described in (5.7) consist of singletons, the SPR term exhibits a strong resemblance to the Berhu regularization. While the Berhu regularization has found successful application in robust regression [90], its performance in the context of pruning remains unexplored. In the following, we present numerical results pertaining to unstructured pruning scenarios involving ResNet-32 and ResNet-56, on the Cifar10 dataset.

We compare our results with two baseline methods that use regularization to prune Neural Networks and with one relevant literature method. The first baseline method is the simple ℓ_1 regularization, known to produce sparser networks compared to the conventional ℓ_2 squared regularization. The second one is the well-known Elastic Net [170], which uses a linear combination of ℓ_1 and ℓ_2 squared regularizations. Formally, the utilization of ℓ_1 regularization yields the following optimization problem:

$$\min_W L(X, W) + \lambda \|W\|_1.$$

While the Elastic Net problem is defined by

$$\min_W L(X, W) + \lambda[\alpha \|W\|_2^2 + (1 - \alpha)\|W\|_1].$$

As the ℓ_1 regularization can be regarded as a limit case of the Elastic Net with the specific parameter α set to 0, we have aggregated their outcomes in the next section for the sake of conciseness and clarity.

Moreover, we performed a comparative evaluation alongside a more complex state-of-the-art technique developed in [26]. This method, although originating from an optimization problem akin to (5.1)-(5.3), subsequently integrates alternating learning and compression phases to systematically achieve pruning in the Neural Network.

We directly report the results from [26], while for all the other methods under comparison, we conducted a systematic grid search, following a similar configuration as detailed in Sec-

tion 5.4.1. In Tables 5.17 and 5.18, we report only the most relevant non-dominated results of the grid search.

Tables 5.17 and 5.18 present clear evidence of SPR's superiority over the baseline methods. Our approach achieves a reduction of over 90% in the number of parameters for ResNet-32 and nearly 94% for ResNet-56, while maintaining an accuracy of over 92% for both architectures. Notably, ℓ_1 regularization competes closely with Elastic Net when applied to ResNet-32 pruning, producing results that are non-dominated and reported in Table 5.17. Conversely, when pruning ResNet-56, Elastic Net consistently outperforms ℓ_1 regularization, occasionally achieving results that are competitive with SPR. Regarding the comparison with [26], the outcomes presented in Tables 5.17 and 5.18 highlight that, despite its relative simplicity compared to the competition, our approach remains competitive within the existing literature. Notably, when pruning ResNet-32, we successfully remove more than 90% of the parameters while achieving nearly identical accuracy compared to the state-of-the-art method that prunes exactly 90% of the network. However, our results are less favorable when pruning ResNet-56. This suggests that employing a more complex optimization algorithm may be crucial for larger architectures or that further hyper-parameter tuning is needed in such scenarios.

5.7 Discussion on the M hyper-parameter

In this section, we discuss the importance of the M parameter appearing in the SPR definition and some considerations surrounding its selection.

The value of M is used when projecting away the y variables in (5.4), and it conveys important information for the SPR. As partially explained in Section 5.3.3, the M parameter is used to assess if a weight is "large" or not: indeed, the SPR term changes its form based on the quantity $\|w\|/M$.

Ideally, the value of M could be chosen such that the weights will *naturally* stay below such value. In practice, this ideal M is not computable and we had to choose M empirically as explained in Section 5.4.1. It is crucial to grasp that opting for an excessively large M is detrimental. Intuitively, this is due to the previously mentioned SPR mechanism that dynamically adapts the definition of the SPR term based on the value of M. Theoretically, it is well documented in the MIP literature that, in formulations that contain constraints such as (5.2), an excessively large M value has a rather negative effect on the quality of the continuous relaxation of the MIP formulation [17]. This continuous relaxation forms the foundation of our approach and it is what we aim to strengthen when using the Perspective function in

Table 5.17 Result on CIFAR-10 using ResNet-32 in the unstructured setting.

Method	Setting	Acc.	Pruned pars (%)	
Elastic Net	λ 15 - α 0.2	92.73	334791	(72.48)
	λ 20 - α 0	92.05	369338	(79.96)
	λ 25 - α 0	91.31	384277	(83.20)
	λ 35 - α 1e-2	90.35	413674	(89.56)
[26]	P-15	92.68	392601	(85.00)
	P-10	92.12	415694	(90.00)
	P-5	90.74	438788	(95.00)
	P-3	89.26	448025	(97.00)
SPR	λ 10 - α 5e-2	93.14	349601	(75.69)
	λ 25 - α 0.2	92.46	405541	(87.80)
	λ 10 - α 0.8	92.11	416428	(90.16)
	λ 35 - α 0.2	90.85	436795	(94.57)
	λ 35 - α 0.6	89.95	441673	(95.62)

Table 5.18 Results on CIFAR-10 using ResNet-56 in the unstructured setting.

Method	Setting	Acc.	Pruned pars (%)	
Elastic Net	λ 20 - α 0.6	93.41	604445	(70.86)
	λ 20 - α 5e-2	93.22	693854	(81.34)
	λ 25 - α 5e-2	92.77	727884	(85.33)
	λ 35 - α 5e-2	91.75	751298	(88.08)
[26]	P-15	93.08	725676	(85.00)
	P-10	93.33	768123	(90.00)
	P-5	92.49	810570	(95.00)
	P-3	91.79	827549	(97.00)
SPR	λ 30 - α 1e-2	93.90	631012	(73.97)
	λ 20 - α 5e-2	92.94	736483	(86.34)
	λ 25 - α 0.2	92.14	799059	(93.67)
	λ 25 - α 0.6	91.34	806005	(94.49)

Section 5.3.1. The practical irrelevance of an excessively large value for M becomes evident when considering the limit where M approaches infinity. In fact, in this limit, the SPR term essentially converges to being almost identical to the ℓ_2 norm.

Acknowledgments

This work has been supported by the NSERC Alliance grant 544900- 19 in collaboration with Huawei-Canada. The authors are repeatedly indebted to two anonymous referees for their careful reading and useful remarks.

CHAPTER 6 ARTICLE 2: ON THE CONVERGENCE OF STOCHASTIC GRADIENT DESCENT IN LOW-PRECISION NUMBER FORMATS

Authors: Matteo Cacciola, Antonio Frangioni, Masoud Asgharian, Alireza Ghaffari, and Vahid Partovi Nia.
Published at 12th International Conference on Pattern Recognition Applications and Methods (22/02/2023)[1].

Abstract Deep learning models are dominating almost all artificial intelligence tasks such as vision, text, and speech processing. Stochastic Gradient Descent (SGD) is the main tool for training such models, where the computations are usually performed in single-precision floating-point number format. The convergence of single-precision SGD is normally aligned with the theoretical results of real numbers since they exhibit negligible error. However, the numerical error increases when the computations are performed in low-precision number formats. This provides compelling reasons to study the SGD convergence adapted for low-precision computations. We present both deterministic and stochastic analysis of the SGD algorithm, obtaining bounds that show the effect of number format. Such bounds can provide guidelines as to how SGD convergence is affected when constraints render the possibility of performing high-precision computations remote.

6.1 Introduction

The success of deep learning models in different machine learning tasks have made these models de facto for almost all vision, text, and speech processing tasks. Figure 6.1 depicts the size of deep learning models, indicating an exponential increase in the size of the models, and hence an urge for efficient computations. A common technique used in training deep learning models is SGD but the theoretical behaviour of SGD in rarely studied in low-precision number formats. Although there is a surge of articles on real numbers (for example see [125], [132], [128]), the performance of SGD in low-precision number formats started recently. Depending on the precision, the loss landscape can change considerably. Figure 6.2, for instance, depicts this situation for ResNet-18 loss landscape in both single-precision and low-precision number formats. Motivated by Figure 6.2, we present a formal study of SGD for quasi-convex functions when computations are performed in low-precision number formats.

[1]The complete reference is [24].

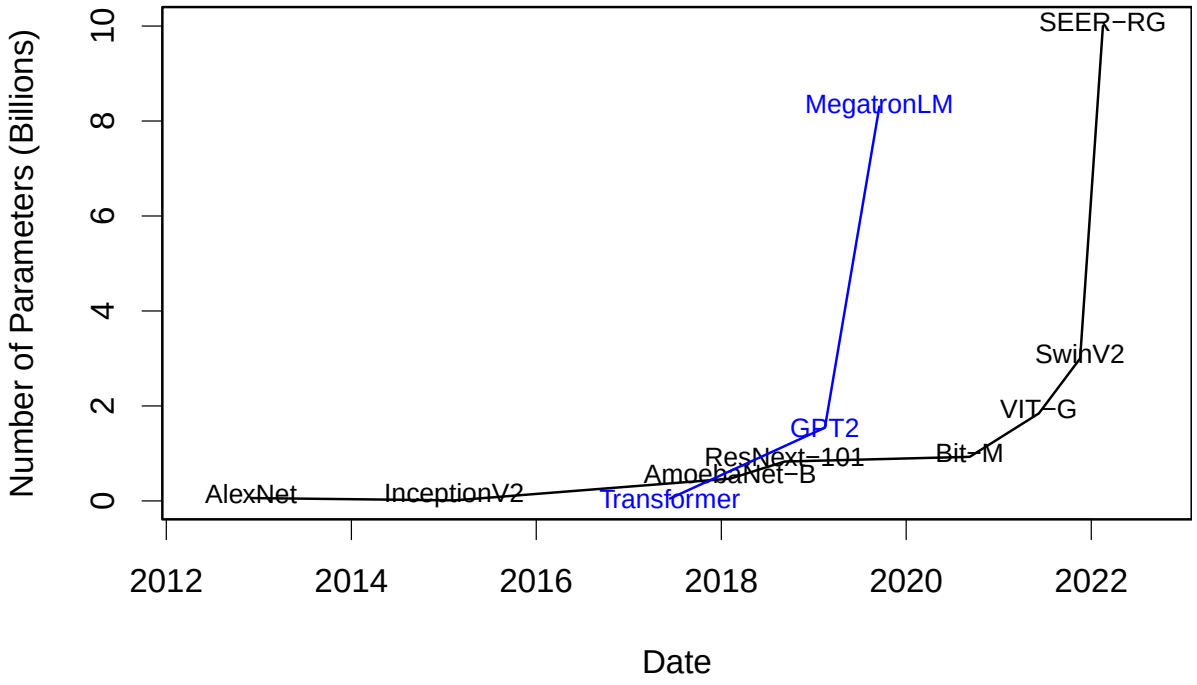


Figure 6.1 Exploding trend of deep models for image classification (black) and language models (blue) in time.

We note that numerical errors, both in forward and back propagation, can possibly affect the convergence behaviour of the algorithm. It is conceivable that the numerical errors should increase as the precision decreases. To understand the effect of number format on the convergence of SGD, a careful analysis of the SGD algorithm for a predetermined precision is needed. We present both deterministic and stochastic analysis of the normalized SGD algorithm, obtaining bounds that show, explicitly, the effect of precision, i.e. number format. Such bounds can provide guidelines as to how SGD convergence is affected when constraints render performing high-precision computations impractical, and to what extent the precision can be reduced without compromising SGD convergence.

Our experiments are performed for logistic regression on MNIST dataset. They confirm that the trajectory of the loss in low-precision SGD setup has at least a limit point whose loss value is in the proximity of the minimum when the numerical errors are relatively small, see Theorem 6.4.1 and Theorem 6.4.3.

This paper is organized as follows. Section 6.2 presents a literature review on the low-precision training of deep learning models and also provides some common background for theoretical analysis of SGD. Section 6.3 discusses some preliminary notations and definitions for analysis of quasi-convex loss function and also the floating point number formats. Section

6.4 contains the main theoretical results. Section 6.5 provides some experimental results that support our theoretical results. We conclude in Section 6.6.

6.2 Related works

Recently, deep learning models provide state-of-the-art performance in various machine learning tasks such as computer vision, speech, and natural language processing (NLP). The size of ImageNet classification models after the introduction of AlexNet size is exploded to 200×, and the size of language models are getting 10× bigger every year. The recent trend of deep learning models shows that larger models such as transformers [146] and their variants such as GPT2 [127], MegatronLM [138], and [21] are easier to generalize on different downstream tasks. Moreover, examples of large language models are included in Figure 6.1 (blue line) and they show an increasing trend in number of parameters over time. A similar trend also appears in vision models, specially after the advent of vision transformers ([159];[60]) that beat convolutional neural networks [114] on various tasks, see Figure 6.1 (black line). Although such large models have advantage in terms of accuracy, they suffer from high computational cost in their training and inference phases. Moreover, the high computational complexity of these models causes high energy consumption and memory usage which makes their training and deployment difficult and even sometimes infeasible. Thus, reducing the computational complexity of large deep learning models is crucial.

On the other hand, there has been some efforts in manually redesigning smaller models with similar accuracy as large models which often require more complicated training. In image classification small models such as MobileNet [72] have a similar accuracy as ResNet [67], and in language models, DistilBERT [131] shows close performance to BERT. Meanwhile there have been some efforts in designing models automatically such as [106];[169]. Other methods include those preserving the baseline model's architecture while modifying computations e.g. compressing large models using sparse estimation [112];[129];[54], or simplifying computations by running on low-precision number formats [82];[150]. Some researchers are even pushing frontiers by storing weights and reducing activation to binary [76] or ternary numbers [99].

Training large models are compute intensive using single-precision floating point. This is why hardware manufacturers such as NVIDIA, Google, Transcent, and Huawei started supporting hardware for low-precision number formats such as Bfloat, float16, and int8. Recently researchers try to map single-precision computations on lower bits, see [162];[164]; [56].

The majority of the literature on SGD assumes convex loss function. We weaken this assumption by considering quasi-convex class of loss functions that include convex functions as

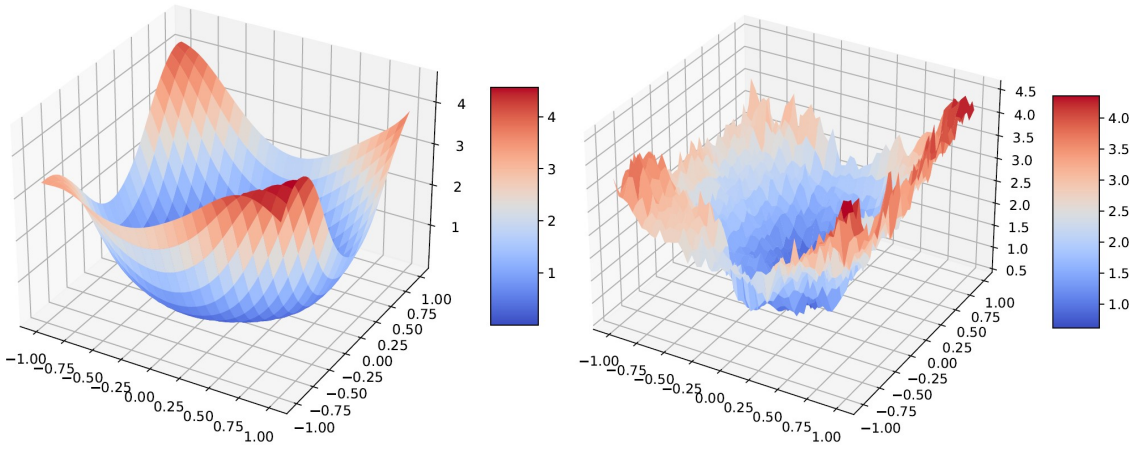


Figure 6.2 ResNet-18 loss landscape in single-precision (left) and low-precision number format (right).

special case. One of the first work on quasi-convex optimization is [85], where is proven that the gradient descent algorithm converges to a stationary point. Later, in [86], the differentiability hypothesis is removed and the convergence result is shown using quasi-subgradients. In the case of perturbed SGD, in [73] the authors are able to deal with bounded biased perturbation on the quasi-subgradient computation. In a subsequent work [74] analyzed the stochastic setting. Recently, [160] studied the low-precision SGD for strongly convex loss functions where the authors used Langevin dynamics. In comparison, our work differs in two aspects (i) we assume quasi-convexity, (ii) our setup adds noise to the SGD and this allows for less stringent assumptions on the noise and its distribution.

6.3 Preliminaries

We start with some preliminary notations about quasi-convexity and floating point number formats in the sequel.

6.3.1 Quasi-convexity

Definition 6.3.1. A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be quasi-convex if $\forall a \in \mathbb{R}$, $f^{-1}[(-\infty, a)] = \{\mathbf{w} \in \mathbb{R}^d | \, f(\mathbf{w}) \in (-\infty, a)\} = \mathcal{S}_{f,a}$ is convex.

Definition 6.3.2. Given a quasi-convex function $f : \mathbb{R}^d \to \mathbb{R}$, the quasi-subgradient of f at $\mathbf{w} \in \mathbb{R}^d$ is defined as $\bar{\partial}^* f(\mathbf{w}) = \{g \in \mathbb{R}^d \mid \langle g, \mathbf{w}' - \mathbf{w} \rangle \leq 0, \ \forall \mathbf{w}' \in \mathcal{S}_{f,f(\mathbf{w})}\}$

In what follows, the optimal value and optimal set of a function f on a set $\mathbf{\Omega}$ are respectively denoted by f^* and $\mathbf{\Omega}^*$, i.e. $f^* = \inf_{\mathbf{w} \in \mathbf{\Omega}} f(\mathbf{w})$ and $\mathbf{\Omega}^* = \operatorname{argmin}_{\mathbf{w} \in \mathbf{\Omega}} f(\mathbf{w})$.

Definition 6.3.3. Let $p > 0$ and $L > 0$. $f : \mathbb{R}^d \to \mathbb{R}$ is said to satisfy the Hölder condition of order p with constant L if

$$f(\mathbf{w}) - f^* \leq L[\text{dist}(\mathbf{w}, \mathbf{\Omega}^*)]^p$$

where $\text{dist}(\mathbf{w}, \mathbf{w}') = \min_{\mathbf{w}'} \|\mathbf{w} - \mathbf{w}'\|$ where $\|\cdot\|$ denotes the Euclidean norm.

The standard theory of SGD convergence relies on convex or even strict convex assumption on the loss function. Clearly quasi-convex assumption is a generalization of convex assumption, i.e. all quasi-convex functions are convex, but the reverse may not be true. To further motivate the quasi-convex assumption we show the ResNet-56 loss-landscape projection in two and three dimensions without skip connections over the CIFAR10 dataset, see Figure 6.3. The convex regions and the quasi-convex regions are highlighted. The quasi-convex regions are larger than the convex regions. This means that our theory is applicable in a larger domain of the loss function.

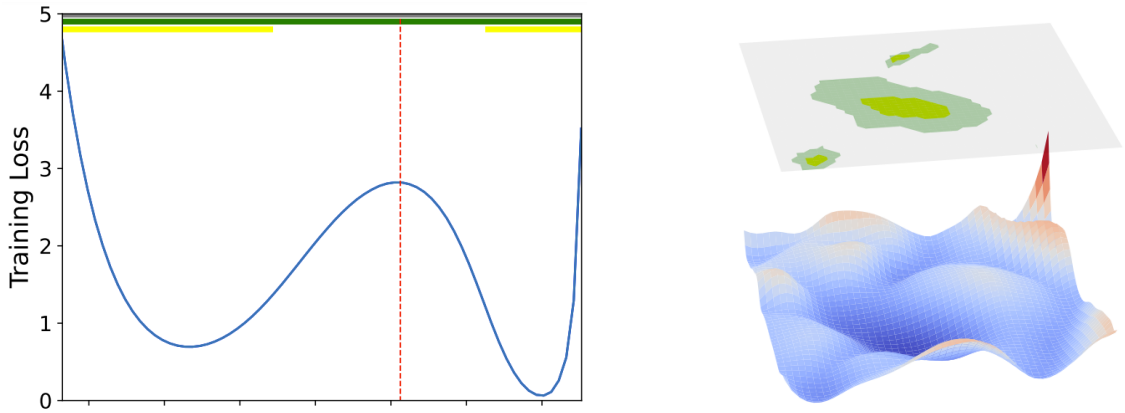


Figure 6.3 The quasi-convex regions (in green) are larger than the convex regions (in yellow).

6.3.2 Floating points

A base $\beta \in \mathbb{N}$, with precision $t \in \mathbb{N}$, and exponent range $[e_{\min}, e_{\max}] \subset \mathbb{Z}$ define a floating point system $\mathbb{F}$, where an element $y \in \mathbb{F}$ can be represented as

$$y = \pm m \times \beta^{e-t}, \tag{6.1}$$

where $m \in \mathbb{Z}$, $0 \leq m < \beta^t$, and $e \in [e_{\min}, e_{\max}]$.

For an $x \in [\beta^{e_{\min}-t}, \beta^{e_{\max}}(1 - \beta^{-t})]$, let the float projection function be $\text{fl}(x) := \mathsf{P}_{\mathbb{F}}(x)$, then

for $x, y \in \mathbb{F}$, the rounding error for basic operations $\text{op} \in \{+, -, \times, /\}$, is

$$\text{fl}(x \,\text{op}\, y) = (x \,\text{op}\, y)(1 + \delta_0), \tag{6.2}$$

where the error is bounded by $\delta_0 \leq \beta^{1-t}$.

When trying to compute a subgradient $g \in \partial f(w)$ for a $w \in \mathbb{F}$ the error is bounded to

$$\text{fl}(g) = g(1 + \delta_1),$$

where $|\delta_1| \leq c\delta_0$ and $c > 0$ depends on the number of operations needed for computing such subgradient. A step of subgradient descent in floating point in $\mathbb{R}^d$ is:

$$\mathbf{w}_{k+1} = \{\mathbf{w}_k - \eta[\mathbf{g}_k(1 + \boldsymbol{\delta}_{1k})]\}\,(1 + \boldsymbol{\delta}_{2k}),$$

where we suppose $\eta \in \mathbb{F}$, $\|\boldsymbol{\delta}_{1k}\| \leq c\delta_0\sqrt{d}$ and $\|\boldsymbol{\delta}_{2k}\| \leq \delta_0\sqrt{d}$ where d is the dimension of $\boldsymbol{\delta}$. We can reformulate it in terms of absolute error

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta(\mathbf{g}_k + \mathbf{r}_k) + \mathbf{s}_k,$$

and if the norm of the subgradient $\mathbf{g}_k$ and $\mathbf{w}_{k+1}$ is bounded, then so are $\|\mathbf{r}_k\| \leq R$ and $\|\mathbf{s}_k\| \leq S$. Note that the infinity norm of the errors are bounded

$$\|\delta_{1k}\|_\infty \leq c\delta_0, \qquad \|\delta_{2k}\|_\infty \leq \delta_0,$$

so

$$R^2 \leq c\delta_0 \|\mathbf{g}_k\|, \quad S^2 \leq \delta_0 \|\mathbf{w}_k\|.$$

6.4 Main results

Although our study is mainly motivated by training deep learning models and floating point errors, they can be applied elsewhere.

6.4.1 Deterministic analysis

Let $\mathsf{P}_\Omega(\cdot)$ be the projection operator over $\boldsymbol{\Omega}$. We start with adapting and improving the results of [73] in the presence of error in the summation

Theorem 6.4.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a quasi-convex function satisfying the Hölder condition of order p and constant L. Let $\mathbf{w}_{k+1} = \mathsf{P}_\Omega(\mathbf{w}_k - \eta\hat{g}_k + \mathbf{s}_k)$ where $\boldsymbol{\Omega} \subset \mathbb{R}^d$ is compact, c is*

the diameter of Ω*, and* $\hat{\mathbf{g}}_k = \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k$*, with* $\mathbf{g}_k \in \bar{\partial}^* f(\mathbf{w}_k)$*,* $\|\mathbf{r}_k\| < R < 1$*,* $\|\mathbf{s}_k\| \leq S$*. Then*

$$\liminf_{k\to\infty} f(\mathbf{w}_k) \leq f^* + L\Gamma^p(c),$$

where

$$\Gamma(c) = \frac{\eta}{2}\left[1 + \left(R + \frac{S}{\eta}\right)^2\right] \bigvee \left[\frac{\eta}{2}\left\{1 - \left(R + \frac{S}{\eta}\right)^2\right\} + c\left(R + \frac{S}{\eta}\right)\right].$$

See the Appendix for the proof.

Remark: Unlike [73], decreasing η does not decrease the error bound always, so we can derive its optimal value by minimizing the bound with respect to η.

Define

$$\eta_1 = \frac{S}{\sqrt{1+R^2}}, \quad \eta_2 = \sqrt{\frac{S(c-2S)}{1-R^2}}, \quad \eta_3 = \frac{c-S}{R}.$$

Corollary 6.4.1.1. *The optimal choice for the step size* η *that minimizes the error bound in Theorem 6.4.1 is reached in at least one of this 3 points* $\{\eta_1, \eta_2, \eta_3\}$*.*

The next result presents a finite iteration version of the previous result. The effect of the number of iterations K, and the starting point $\mathbf{w}_0$ are clearly reflected in the bound for $\min_{k<K} f(\mathbf{w}_k)$.

Theorem 6.4.2. *Let* $f : \mathbb{R}^d \to \mathbb{R}$ *be a quasi-convex function satisfying the Hölder condition of order* p *and constant* L*. Let* $\mathbf{w}'_{k+1} = \mathbf{w}_k - \eta\hat{\mathbf{g}}_k$ *and* $\mathbf{w}_{k+1} = \mathbf{w}'_{k+1} + \mathbf{s}_k$ *where* $\hat{\mathbf{g}}_k = \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k$*, where* $\mathbf{g}_k \in \bar{\partial}^* f(\mathbf{w}_k)$*,* $\|\mathbf{r}_k\| < R$*,* $\|\mathbf{s}_k\| \leq S$*. Then,*

$$\min_{k<K} f(\mathbf{w}_k) \leq f^* + L\left[\Gamma(c_0) + \frac{c_0}{2\eta K}\right]^p,$$

with $c_0 = \|\mathbf{w}_0 - \mathbf{w}^*\|$*.*

See the Appendix for the proof.

6.4.2 Stochastic analysis

Here, we present the stochastic counterpart of Theorem 6.4.1. The theorem requires only mild conditions on the first two moments of the errors. We start by defining the notion of stochastic quasi-subgradient.

Definition 6.4.1. Let $\mathbf{w}$ and $\mathbf{w}'$ be d-dimensional random vectors defined on the probability space $(\mathcal{W}, \mathcal{F}, \mathcal{P})$ and $f : \mathbb{R}^d \to \mathbb{R}$ be a measurable quasi-convex function. Then $\mathbf{g}(\mathbf{w})$ is called a unit noisy quasi-subgradient of f at $\mathbf{w}$ if $\|\mathbf{g}(\mathbf{w})\| \overset{a.s.}{=} 1$ and $\mathcal{P}\left\{\mathcal{S}_{f,f(\mathbf{w})} \cap \mathcal{A}_{\mathbf{w}} \neq \emptyset\right\} = 0$, where $\mathcal{A}_{\mathbf{w}} = \{\mathbf{w}' : \langle \mathbf{g}(\mathbf{w}), \mathbf{w}' - \mathbf{w}\rangle > 0\}$.

Thus, inspired by results of [74], we prove the following theorem that take into account both randomness in the gradient and the computation numerical error.

Theorem 6.4.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuous quasi-convex function satisfying the Hölder condition of order p and constant L . Let $\mathbf{w}_{k+1} = \mathsf{P}_{\Omega}[\mathbf{w}_k - \eta(\hat{\mathbf{g}}_k + \mathbf{r}_k) + \mathbf{s}_k]$ where $\mathbf{\Omega}$ is a convex closed set, $\hat{\mathbf{g}}_k$ is a unit noisy quasi-subgradient of f at $\mathbf{w}_k$, $\mathbf{r}_k$'s are i.i.d. random vectors with $\mathbb{E}\{\mathbf{r}_k\} = \mathbf{0}$ and $\mathbb{E}\{\|\mathbf{r}_k\|^2\} = d\sigma_r^2$, and similarly $\mathbf{s}_k$'s are i.i.d random vectors with $\mathbb{E}\{\mathbf{s}_k\} = \mathbf{0}$ and $\mathbb{E}\{\|\mathbf{s}_k\|^2\} = d\sigma_s^2$. Further assume that $\hat{\mathbf{g}}_k$, $\mathbf{s}_k$, and $\mathbf{r}_k$ are uncorrelated. Then,*

$$\liminf_{k\to\infty} f(\mathbf{w}_k) \leq f^* + L\left[\frac{\eta}{2}(1 + d\sigma_r^2) + \frac{d\sigma_s^2}{2\eta}\right]^p \quad a.s.$$

See the Appendix for the proof.

Similar to Corollary 6.4.1.1 one can derive the optimal step size.

Corollary 6.4.3.1. *The optimal step size η that minimizes the error bound in Theorem 6.4.3 is $\eta^* = \sqrt{\frac{d\sigma_s^2}{d\sigma_r^2+1}}$.*

Remark: It immediately follows the optimal step size is $\frac{\sigma_s}{\sigma_r}$ for large d.

The theorems indicate that the upper bounds increase with S and R, reflecting the effect of accumulation and multiplication errors respectively. Smaller number formats clearly lead to greater values of S and R. As for η, the step size, there is a tradeoff. To make sure that the bounds provide useful information for practical purposes, one should choose the step size such that $\frac{S}{\eta}$ ($\frac{\sigma_s}{\eta}$) is controlled. This essentially means that more accuracy is required for smaller step sizes.

6.5 Experiments

We performed two types of experiments, a simple quasi-convex function and a logistic regression on MNIST dataset.

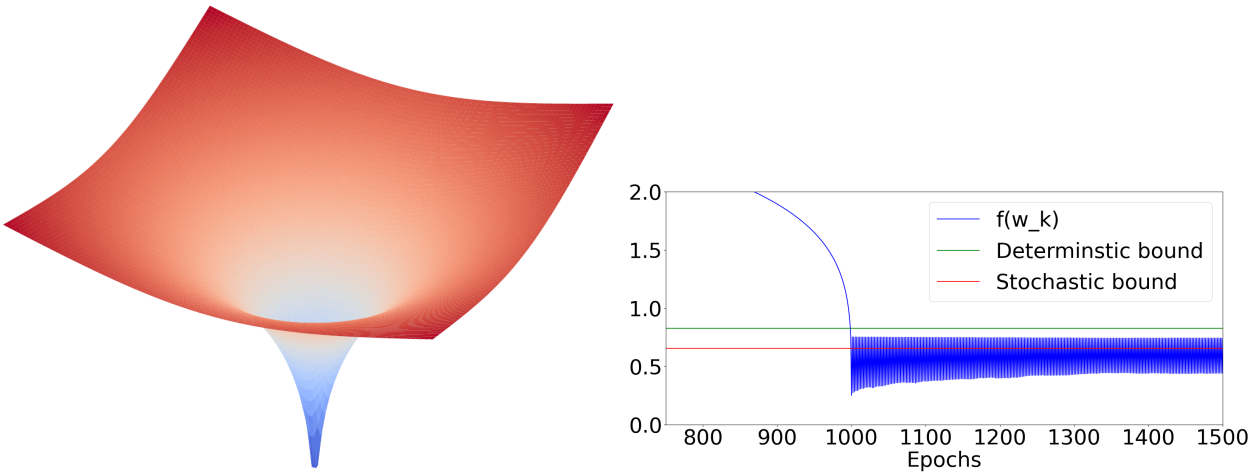


Figure 6.4 A two dimensional $\|\mathbf{w}\|^{0.2}$ quasi-convex function (left panel), and the SGD trace plot confirming the stochastic and deterministic bounds hold (right panel).

6.5.1 Simple quasi-convex function

To asses the bounds obtained in our theorems we start with a simple quasi-convex function that exactly satisfy the Holder's condition. We chose $f(\mathbf{w}) = 3\|\mathbf{w}\|^{0.2}$ where $\mathbf{w} \in \mathbb{R}^{40}$. In this example, the parameters of Holder's condition are $p = 0.2$ and $L = 3$. We added noise to the gradients and to the weight update denoted by $\|\mathbf{r}_k\|$ and $\|\mathbf{s}_k\|$ respectively. This noise has a uniform distribution $\mathbf{r}_{ki} \sim U(-B_r, B_r)$, and $s_{ki} \sim U(-B_s, B_s)$. Figure 6.4 shows the stochastic and deterministic bound for this experiment. Note that, the theoretical bounds holds in both stochastic and deterministic cases.

6.5.2 Logistic regression

Here, we present experimental results of logistic regression on the first two principal components of MNIST dataset. For this experiment, we need to estimate the parameters of the Holder's condition for the loss function in order to compute the bounds. To do so, the Holder's parameters p and L are manually fitted to the loss function that is evaluated at different distances from the optimal point, see Figure 6.5. The optimal point $\mathbf{w}^*$ in our experiments is obtained using single-precision floating point gradient descent (GD) method and is used as a reference to compute the parameters of the bounds f^* and c.

Computation of the gradients involves inner products that are computed by multipliers and accumulators. The accumulator have numerical error relative to its mantissa size. We tested our logistic regression setup using Bfloat number format and reduced accumulator size. Also note that according to Theorem 6.4.3, the values of $d\sigma_s^2$ and $d\sigma_r^2$ are required to compute the

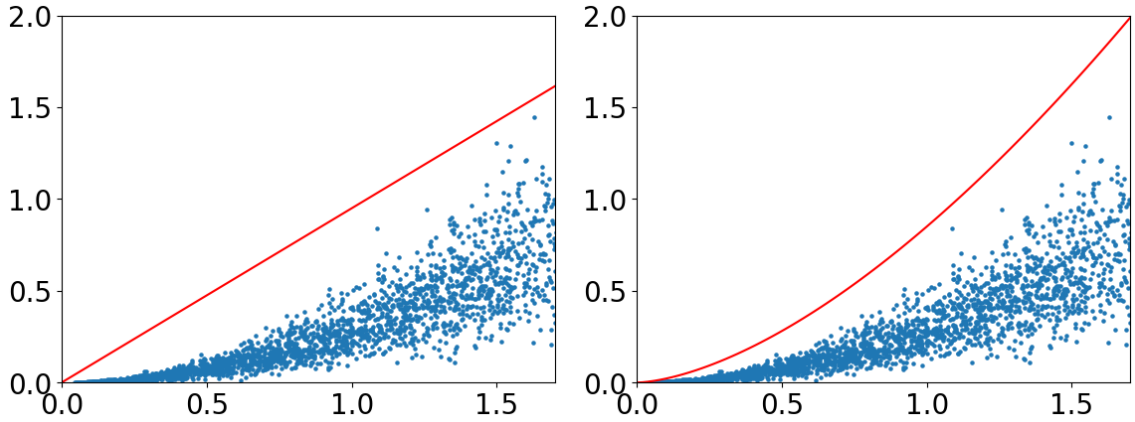


Figure 6.5 The Holder's parameters p and L are manually fitted to the loss function that is evaluated at different distances from the optimal point. The left panel is a linear fit with $p = 1$ and $L = 0.95$. The right panel demonstrate the fit with $p = 1.6$ and $L = 0.85$. We used $p = 1.6$ and $L = 0.85$ for our experiments.

bounds. Thus, in our experiments, we used empirical values of those parameters to compute the bounds. Also note that we did not plot the deterministic bounds for these experiments as they are too pessimistic.

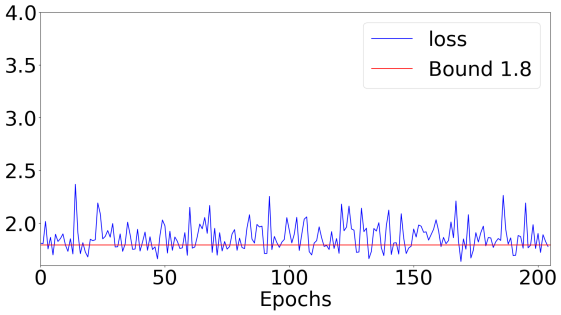


Figure 6.6 Logistic regression trained using single-precision SGD and a fixed learning rate.

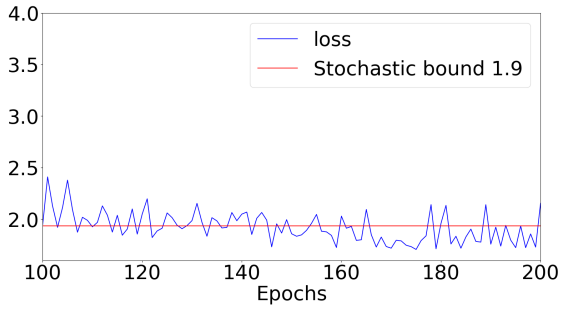


Figure 6.7 Logistic regression trained using Bfloat SGD with accumulator size of 15 ($\mathbf{s}_k \neq 0$ and $\mathbf{r}_k \neq 0$).

In order to evaluate the Holder's condition parameters, p and L, estimated as shown in Figure 6.5, we use a single-precision SGD to confirm if the bounds hold. Figure 6.6 demonstrates that the loss trajectory (blue line) has a limit point in the proximity of the optimal point of the convex loss function. Figure 6.7 demonstrates the loss trajectory when both weight update and gradient computations are performed using Bfloat number format. Note that Bfloat has 8 bit exponent and 7 bit mantissa and is used recently to train deep learning models. The computations are performed using 15-bit accumulator mantissa. Figure 6.8

shows the loss trajectory when the weight update is in single precision and only gradient computations are performed using Bfloat number format. In this experiment, the stochastic bound is numerically equal to the single-precision SGD, indicating that the precision of weight update is more important compared to the precision of the gradients.

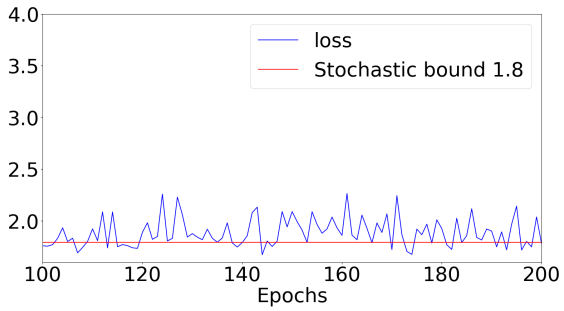


Figure 6.8 Logistic regression trained using Bfloat gradients with accumulator size of 15, and single-precision weight update ($\mathbf{s}_k = 0$ and $\mathbf{r}_k \neq 0$).

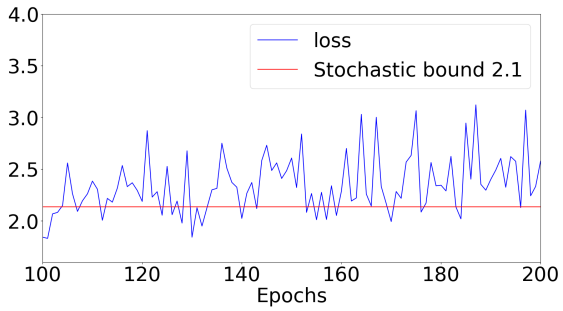


Figure 6.9 Logistic regression trained using Bfloat gradients with accumulator size of 10, and single-precision weight update ($\mathbf{s}_k = 0$ and $\mathbf{r}_k \neq 0$).

Reducing the accumulator mantissa size has a direct effect on the convergence of SGD. Figure 6.9 shows that stochastic bound is increased in the case of 10-bit accumulator size. In this experiment, the loss trajectory oscillates more in the neighbourhood of the optimum point. This indicates the accumulator size plays an important role in reducing the numerical errors of the low-precision SGD computations, and consequently improves the convergence of SGD.

We used the Normalized Gradient Descent (NGD) algorithm to perform the experiments with the deterministic function $f(\mathbf{w}) = \|\mathbf{w}\|^{0.2}$ presented in Section 6.5. The maximum number of epochs is set to 1500. Different values for C_0, B_r, B_s and learning rates were used. The errors, which are manually added, have uniform distribution in each coordinate. Thus, the variances required in Theorem 6.4.3 are $\sigma_r^2 = \frac{B_r^2}{3}$ and $\sigma_s^2 = \frac{B_s^2}{3}$. For the computation of the bound in Theorem 6.4.1, $c = C_0$ was used.

6.5.3 Optimal learning rate

We performed experiments with fixed values for B_s, B_r, but different choices of η to acquire the optimal choice η^* given by Corollary 6.4.3.1 . The experiment is repeated 10 times, for each tested value of η,. Finally, the maximum loss function value observed across all the experiments with the same η is plotted at each epoch.

The results with $B_r = B_s = 0.1$ are shown in Figure 6.10 and Figure 6.11. The loss trajectory (blue line) observed with the value suggested by Corollary 6.4.3.1, $\eta = 0.0348$, is the trajec-

tory that has the lowest level . Our theorems correctly predict that decreasing the value of η is sometimes not beneficial in terms of convergence, see Figure 6.11.

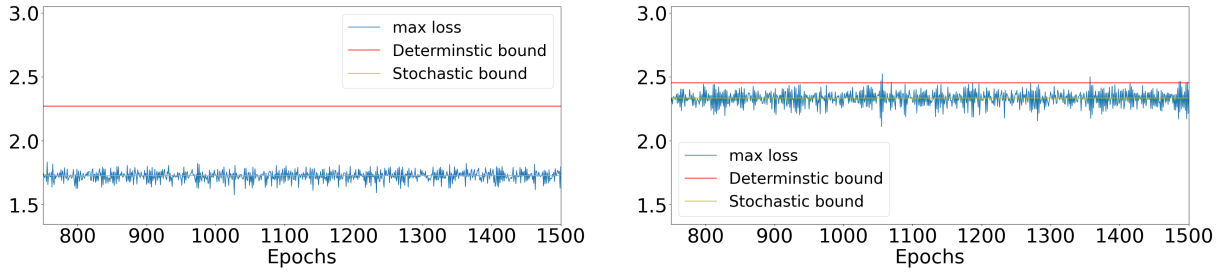


Figure 6.10 Results with $\eta = 0.1$ (left panel) and $\eta = 0.5$ (right panel)

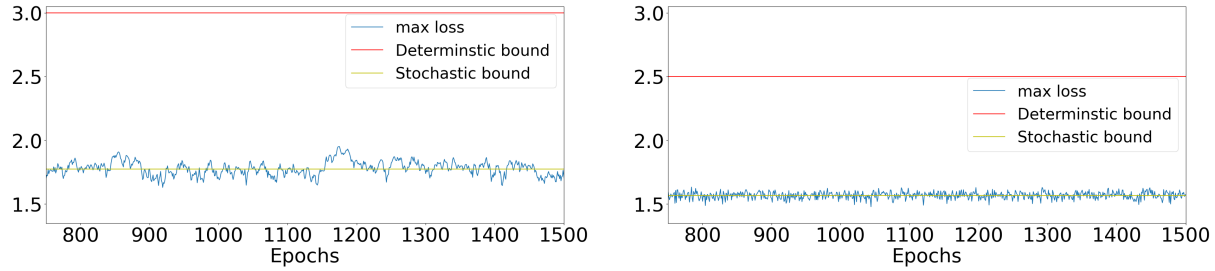


Figure 6.11 Results with $\eta = 0.01$ (left panel) and $\eta = 0.0348$ (right panel), decreasing the value of η leads to worse bound and worse convergence

6.5.4 MNIST image classification

In this section the results obtained on the original MNIST dataset are reported. In contrast to the experiments in the main body of the manuscript, PCA is not used to reduce the size of the inputs.

Figure 6.12 demonstrates that the loss trajectory (blue line) has a limit point in the proximity of the optimal point of the convex loss function. Figure 6.13 shows the loss trajectory when the weight update is in single precision and only gradient computations are performed using Bfloat number format. In this experiment, the stochastic bound is numerically equal to the single-precision SGD, confirming what already observed in Figure 6.8.

6.6 Conclusion

We have studied the convergence of low-precision floating-point SGD for quasi-convex loss functions and extended some existing deterministic and stochastic bounds for convex loss

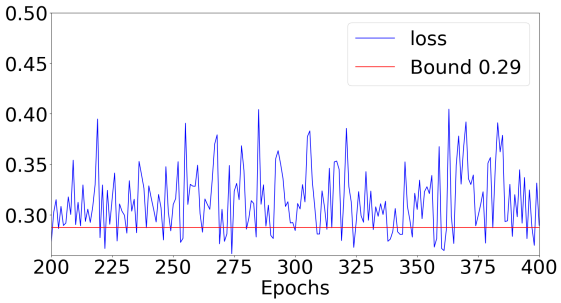


Figure 6.12 Logistic regression trained using single-precision SGD and a fixed learning rate.

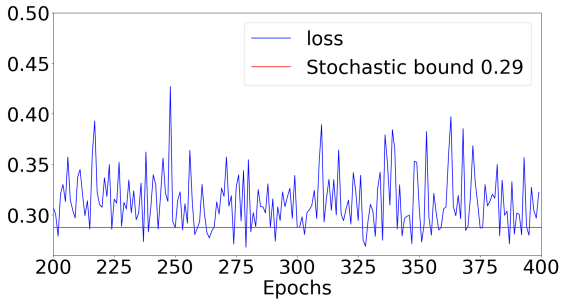


Figure 6.13 Logistic regression trained using Bfloat gradients with accumulator size of 50, and single-precision weight update ($\mathbf{s}_k = 0$ and $\mathbf{r}_k \neq 0$).

functions. In our theoretical setup, we considered numerical errors for weight update and gradient computations. We have also derived the optimal step size as a corollary of our theoretical results. Furthermore, in our experiments, the effect of numerical errors on weight update and gradient computations are demonstrated. Our experiments show that the accumulator mantissa size plays a key role in reducing the numerical error and improving the convergence of SGD. Although our experiments with logistic regression are promising, extension of the experiments for more complex models is an appealing direction as the future work.

CHAPTER 7 ARTICLE 3: STRUCTURED PRUNING OF NEURAL NETWORKS FOR CONSTRAINTS LEARNING

Authors: Matteo Cacciola, Antonio Franginoni, and Andrea Lodi.
Submitted to Operations Research Letters (13/07/2023) [1].

Abstract In recent years, the integration of Machine Learning (ML) models with Operations Research (OR) tools has gained popularity across diverse applications, including cancer treatment, algorithmic configuration, and chemical process optimization. In this domain, the combination of ML and OR often relies on representing the ML model output using Mixed Integer Programming (MIP) formulations. Numerous studies in the literature have developed such formulations for many ML predictors, with a particular emphasis on Artificial Neural Networks (ANNs) due to their significant interest in many applications. However, ANNs frequently contain a large number of parameters, resulting in MIP formulations that are impractical to solve, thereby impeding scalability. In fact, the ML community has already introduced several techniques to reduce the parameter count of ANNs without compromising their performance, since the substantial size of modern ANNs presents challenges for ML applications as it significantly impacts computational efforts during training and necessitates significant memory resources for storage. In this paper, we showcase the effectiveness of pruning, one of these techniques, when applied to ANNs prior to their integration into MIPs. By pruning the ANN, we achieve significant improvements in the speed of the solution process. We discuss why pruning is more suitable in this context compared to other ML compression techniques, and we identify the most appropriate pruning strategies. To highlight the potential of this approach, we conduct experiments using feed-forward neural networks with multiple layers to construct adversarial examples. Our results demonstrate that pruning offers remarkable reductions in solution times without hindering the quality of the final decision, enabling the resolution of previously unsolvable instances.

7.1 Introduction

The concept of embedding learned functions inside Mixed Integer Programming (MIP) formulations, also known as "Learning-Symbolic Programming" or "Constraint Learning", has gained attention in recent literature [5, 15, 133]. Furthermore, there has been an increase in the availability of tools that automatically embed commonly used predictive models into

[1] A pre-print is available in [25].

MIPs [13, 27, 117, 143]. These techniques and tools are especially valuable when employing ML models for predictions and utilizing OR methods for decision making based on those predictions. Unlike the traditional two-stage approaches [45], embedding the predictive model within the decision-making process in an end-to-end optimization framework has been shown to yield superior results. Examples of applications are automatic algorithmic configuration [80, 81], adversarial examples identification [46], cancer treatments development [14], and chemical process optimization [43, 44].

A very relevant case is when the learned function is an ANN, since ANNs are the state-of-the-art models for numerous essential ML tasks in Computer Vision and Natural Language processing. Consequently, there have been efforts in the literature to automate the embedding of ANNs [61]. For instance, [13] enables to incorporate feed-forward architectures with ReLU activation functions into MIPs, utilizing the output of the ANN in the objective function. The maturity of the field is demonstrated by the fact that one of the leading commercial MIP solvers, Gurobi, recently released a package that allows feed-forward ReLU networks to be part of MIP formulations, with compatibility for popular ML packages such as PyTorch, Keras, and scikit-learn.

Unfortunately, even when we consider simple architectures that have only ReLU activation functions, the representation of an ANN in a MIP will introduce binary variables, due to the combinatorial nature of the ReLU function. Additionally, the number of binary variables and the associated constraints that need to be added to the MIP is proportional to the number of parameters in the ANN. Deep Learning has witnessed a clear trend towards developing architectures with a very large number of parameters, which contributes to ANNs high predictive power and state-of-the-art performance in various applications. This, however, poses issues in terms of training costs, storage requirements, and prediction time. Consequently, numerous methods, known as model compression techniques, have been developed to reduce the size of ANNs without compromising their predictive capability. Yet, the large size of the ANNs presents an even more significant scalability challenge when it is embedded into a MIP, due to the potentially exponential growth of the latter computational cost with its size (and, in particular, the number of binary variables). Using a state-of-the-art network in a MIP formulation may easily result in an overwhelming number of binary variables and constraints, rendering the models unsolvable within a reasonable time using any available solver.

In this paper, we demonstrate that pruning methods, originally developed to address specific ML challenges, can be effectively applied in the context of embedding ANNs into MIPs. Specifically, we utilize a structured pruning technique that we previously developed to sig-

nificantly accelerate the solution time for adversarial example identification problems using Gurobi.

The remainder of the paper is organized as follows: Section 7.2 provides a formal definition of the problem concerning the embedding of learned functions in MIP formulations. Additionally, it presents one of the existing formulations from the literature specifically designed for embedding ANNs. In Section 7.3, we introduce pruning techniques and we describe the specific pruning method employed in our experiments. Section 7.4 focuses on the benefits of pruning when incorporating ANNs into MIPs. We discuss the reasons why pruning is advantageous in this context and provide insights on selecting appropriate pruning techniques. Finally, in Section 7.5 we present numerical results to empirically validate that pruning can effectively speed up the solution process of MIPs with embedded ANNs.

7.2 Embedding learned functions in Mixed Integer Programs

We consider a general class of (Mixed-Integer) Nonlinear Programs with "learned constraints". That is, the formulation of the problem would need to involve some functions $g_i(x)$, $i = 1, \dots, k$, defined on the variable space of the optimization decisions, that are "hard" in the sense that no compact algebraic formulation, and not even an efficient computation oracle, is available. Yet, (large) data sets are available, or can be constructed, of outputs $\bar{y} = g_i(\bar{x})$ for given $\bar{x}$. These datasets can be used in several existing ML paradigms (Support Vector Machines, Decision Trees, ANNs, ...) to construct estimates $\bar{g}_i(x)$ of each $g_i(x)$, $i = 1, \dots, k$, with a workable algebraic description that can then be inserted into an optimization model. Thus, we consider the class of Mathematical Programs with Learned Constraints (MPLC)

$$\min_{x,y} cx + by \tag{7.1}$$

$$\text{s.t. } y_i = \bar{g}_i(x) \qquad i = 1, \dots, k \tag{7.2}$$

$$Ax + By \leq d \tag{7.3}$$

$$x \in X \tag{7.4}$$

Linearity in (7.1) and (7.3) is not strictly necessary in our development, but it is often satisfied in applications (see, e.g., [13, 14, 46]) and we assume it for notational simplicity. Indeed, when X in (7.4) contains integrality restrictions on (some of) the x variables, the class already contains Mixed-Integer Linear Programs (MILP), whose huge expressive power does not need to be discussed. Of course, a significant factor in the complexity of (7.1)–(7.4) is the algebraic form of the $\bar{g}_i(x)$, which impacts the class of optimization problems it ultimately belongs to. A significant amount of research is already available on formulations

for embedding feedforward ANNs, in particular with ReLU activations, in a MIP context [5, 15, 46, 78, 133]. In these formulations, the neural network is constructed layer by layer. Denoting the input vector at layer ℓ as o_ℓ, and the corresponding weight matrix and bias vector as W_ℓ and b_ℓ, respectively, one has

$$o_{\ell+1} = \max(\, 0\,,\, W_\ell o_\ell + b_\ell\,)$$

that can be expressed in a MI(L)P form as

$$v_\ell^+ - v_\ell^- = W_\ell o_\ell + b_\ell \tag{7.5}$$
$$0 \le v_\ell^+ \le M^+ z_\ell \tag{7.6}$$
$$0 \le v_\ell^- \le M^- (1 - z_\ell) \tag{7.7}$$
$$o_{\ell+1} = v_\ell^+ \tag{7.8}$$
$$z_\ell \in \{\, 0\,,\, 1\,\}^m \tag{7.9}$$

Constraints (7.6) and (7.7) ensure that both v_ℓ^+ and v_ℓ^- are (component-wise) positive, and since the z_ℓ are (component-wise) binary, that at most one of them is positive. Consequently, constraint (7.5) forces the relations $v_\ell^+ = \max\{\, W_\ell o_\ell + b_\ell\,,\, 0\,\}$ and $v_\ell^- = \min\{\, W_\ell o_\ell + b_\ell\,,\, 0\,\}$ (of course, constraint (7.8) is only there to make apparent what the output of the layer is). Denoting by n the number of neurons in layer ℓ and by m the number of neurons in layer $\ell + 1$, system (7.5)–(7.9) contains m binary variables, $n + 2m$ continuous variables, and $3m$ constraints. A significant aspect of this model (fragment) is the use of big-M constraints (7.6) and (7.7). It is well known that the choice of the value for the constants M can significantly impact the time required to solve an instance. Indeed, the Optimized Big-M Bounds Tightening (OBBT) method has been developed in [46] to find effective values for this constant.

As previously mentioned, the state-of-the-art solver Gurobi now includes an open-source Python package that automatically embeds ANNs with ReLU activation into a Gurobi model. Additionally, starting from the 10.0.1 release, Gurobi has the capability to detect if a model contains a block of constraints representing the relationship $y = g(x)$, where $g(\cdot)$ is an ANN, in order to then apply the aforementioned OBBT techniques to enhance the solution process. Despite showing a substantial improvement with respect to the previous version, the capabilities of Gurobi to solve these MIPs are still limited. In particular, when embedding an ANN into a MIP, Gurobi is not able to solve the problem in a reasonable time unless the number of layers and neurons in the ANN is small.

7.3 Artificial Neural Networks Pruning

As mentioned in the introduction, the size of state-of-the-art ANNs has been growing exponentially over the years. While these models deliver remarkable performance, they come with high computational costs for training and inference, as well as substantial memory requirements for storage. To address this issue, various techniques have been developed to reduce these costs without significantly compromising the predictive power of the network. One such technique is pruning, which involves reducing the ANN size by eliminating unnecessary parameters. Consider for instance a linear layer with input x_{inp}, output x_{out}, and weight and bias tensors W and b, i.e., $x_{out} = Wx_{inp} + b$. Thus, pruning entails removing certain entries from W or b. That is, pruning, say, the parameter $W_{1,1}$ results in the first coordinate of x_{inp} being ignored in the scalar product when computing the first coordinate of x_{out}. Pruning

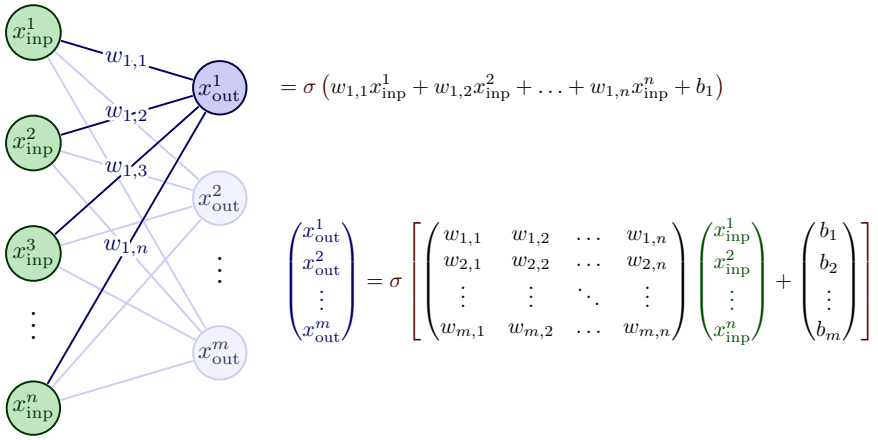


Figure 7.1 Unpruned network

individual weight entries can offer some advantages, but it is generally suboptimal. Since most of the computation is performed on GPUs, there is little computational benefit unless entire blocks of computation, such as tensor multiplications, are removed. Removing entire structures of the ANN is known as *structured* pruning, in contrast to *unstructured* pruning that involves eliminating single weights. In the example of the linear layer, structured pruning would aim to remove entire neurons by deleting rows from the W tensor (along with the corresponding b entry in most cases). Figures 7.1, 7.2, and 7.3 illustrate the difference between these two pruning techniques.

The literature on pruning techniques for neural networks is vast and encompasses a wide range of approaches. One simple and commonly used method is magnitude-based pruning, which involves removing parameters with small magnitudes. This was first introduced in [64] and has been widely adopted since. However, more sophisticated strategies have also been

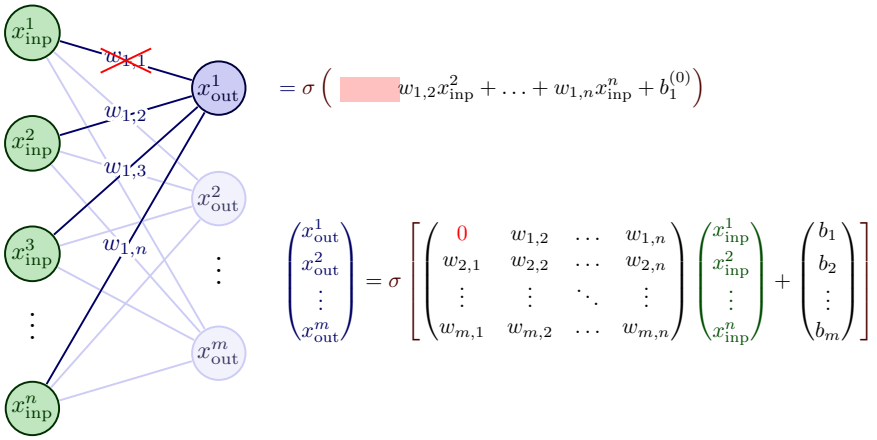


Figure 7.2 Unstructured pruning

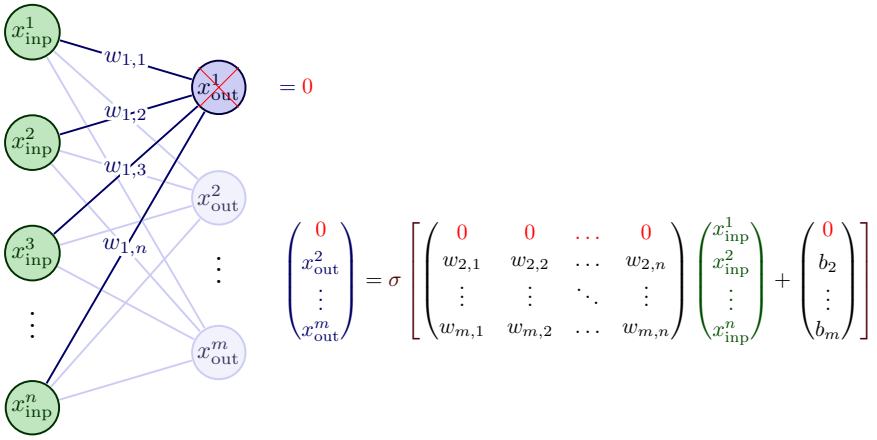


Figure 7.3 Structured pruning

proposed, such as Bayesian methods [11, 110, 118, 165], combinations of pruning with other compression techniques [4, 52, 113], and zero accuracy drop pruning [10, 28, 63, 158].

A relevant subset of pruning techniques uses a regularization term to enforce sparsity in tensor weight. It is common practice in Machine Learning to add a regularization term $R(w)$ to the standard loss function $L(X, Y, w)$, where w is the vector containing the ANNs parameters and (X, Y) is the training set. Usually, $R(w)$ penalizes the magnitude of the parameters (e.g., $R(w) = |w|_2^2$) and it is known to improve the generalization performances of the model. If the form of $R(w)$ is chosen carefully, e.g., $R(w) = |w|_1$, it can also lead to a sparse parameter vector w. When a network parameter is zero, it can typically be removed without changing the model output for any given input. Hence, if $R(w)$ is chosen appropriately to induce all the weights of some neurons to be zero, then such neurons can be removed from the network. Many regularization terms have been proposed both for structured and unstructured pruning,

including but not limited to l_1 norm, BerHu term [90], group lasso, and l_p/l_q norms [29].

7.3.1 The Structured Perspective Regularization Term

In the literature, the majority of pruning techniques rely on heuristics to determine the impact of removing a parameter or a structure from the ANN. This trend persists in recent works [53, 126, 136, 157], including methods that still utilize simple magnitude-based criteria [30, 55, 100, 163]. Only a few techniques attempt to develop a theoretically-grounded methodology [26, 111, 118, 161], and these methods do not primarily focus on structured pruning. In light of this, a pruning technique was developed in [23] that is motivated by strong theoretical foundations and specifically addresses structured pruning. In [23], the pruning problem is addressed by starting with a naïve exact MIP formulation and then deriving a stronger formulation by leveraging the Perspective Reformulation technique [39]. Analogously to what is done in [47–49] for individual variables rather than groups of them, an efficient way to solve the continuous relaxation of this problem is obtained by projecting away the binary variables, resulting in an equivalent problem to standard ANN training with the inclusion of the new *Structured Perspective Regularization* (SPR) term

$$z(W;\alpha,M) = \begin{cases} 2\sqrt{(1-\alpha)\alpha}||W||_2 & \text{if } \frac{||W||_\infty}{M} \leq \sqrt{\frac{\alpha}{1-\alpha}}||W||_2 \leq 1 \\ \frac{\alpha M}{||W||_\infty}||W||_2^2 + (1-\alpha)\frac{||W||_\infty}{M} & \text{if } \sqrt{\frac{\alpha}{1-\alpha}}||W||_2 \leq \frac{||W||_\infty}{M} \leq 1 \\ \alpha||W||_2^2 + (1-\alpha) & \text{otherwise,} \end{cases}$$

where M is a constant, α is a tunable hyper-parameter and W is the weight tensor corresponding to the structure we want to prune (e.g., the weight matrix of a neuron). That is, in order to prune the ANN one trains it using as loss function

$$L(X,Y,W) + \lambda \sum_{j\in\mathcal{N}} z(W_j;\alpha,M),$$

where W_j is the weight matrix corresponding to neuron j and $\mathcal{N}$ is the set of neurons of the ANN. Coupled with a final magnitude-based pruning step, this approach has been shown to provide state-of-the-art pruning performances thanks to the unique and interesting properties of the SPR term. This potentially comes at the expense of extra hyperparameter tuning effort for α and M, which is unlikely to be a major issue in this application since ANNs that can be embedded in a MILP, even after pruning, cannot possibly have the extremely large size common in applications like Computer Vision and Naturale Language Processing, and therefore their training and tuning time is unlikely to be a major factor.

7.4 Pruning as a Speed-Up Strategy

As previously mentioned, in the context of embedding ANNs in MIPs, scalability becomes a significant challenge as the number of (binary) variables and constraints grows proportionally with the number of parameters in the embedded ANN, but the cost of solving the MI(L)P may well grow exponentially in the number of (binary) variables. It therefore makes even more sense to employ the ML compression techniques that are used to reduce the computational resources required by ANNs. Many compression techniques other than pruning exist in the ML literature. However, not all of them are effective in the context of MIPs with embedded ANNs. For instance, quantization techniques aim to train networks that have weight values in a discrete (relatively small) set of $\mathbb{R}$ [32, 150]. One possibility is to directly implement the ANN using a lower bit number format than the standard Float-32 one [24, 82]. Quantization is a very popular technique in ML since can decrease both backward- and forward-pass computational effort, at the same time reducing the memory footprint of the resulting model. However, in the contest of MIPs, quantization does not bring any advantage, since the resulting problem from embedding a quantized ANN is not significantly different, from an Operations Research point of view, to the one where a non-quantized model has been embedded. Indeed, weights are coefficients in (7.5)–(7.9), and having them in a small set of (integer) values may at most have a minor impact on the solution time.

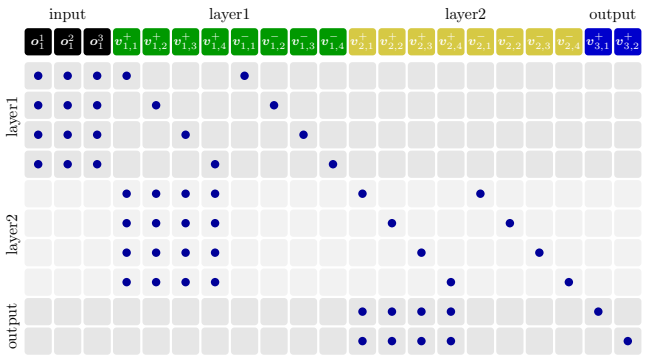


Figure 7.4 Constraints matrix

Other methods, like low-rank decomposition and parameter-sharing techniques [33, 40, 83, 155], modify the internal operations of layers; this means that they cannot directly be used in this context without the development of new, specific formulations and new algorithms that can automatically detect them in a MIP problem.

By contrast, structured pruning techniques perfectly fit the needs of embedding an ANN in a MIP. Even unstructured pruning may have some impact, since when a weight is removed

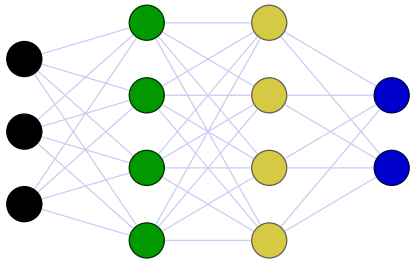


Figure 7.5 Corresponding network

(i.e., set to zero) the corresponding entry in the MIP constraints matrix is also set to zero, leading to a sparser constraints matrix. However, entirely removing variables or constraints is more effective; in the case of a feed-forward ANN, this corresponds to performing structured pruning on neurons, as visualized in Figures 7.4- 7.5- 7.6- 7.7.

It is interesting to remark that, for ML purposes, pruning techniques only bring advantages at inference time, but reducing the number of parameters only reduces linearly the computational cost of the forward pass. By contrast, removing neurons of a network brings an exponential speed up in the time required to solve the resulting MIP formulations. Hence, pruning is arguably more relevant for OR than for ML, despite having been developed in the latter area. In particular, structured pruning—as opposed to unstructured one—is crucial in that it allows using existing automatic structure detection algorithms, such as that implemented in Gurobi, while unstructured pruning is very likely to result in a different structure of the constraints matrix that would not be recognizable, thereby preventing the use of OBBT techniques that are crucial in this context.

Based on the considerations above, we argue about modifying the existing pipeline for embed-

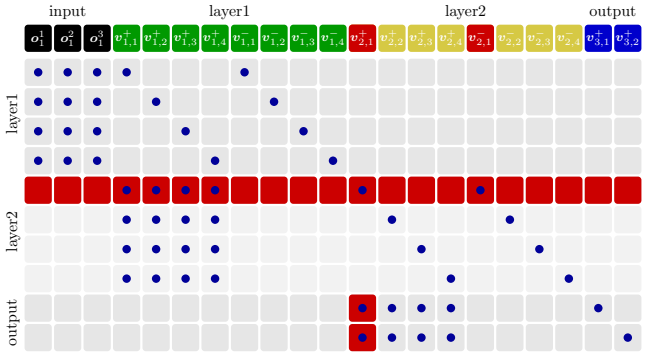


Figure 7.6 Constraints matrix, in red the removed constraints and variables due to neurons pruning

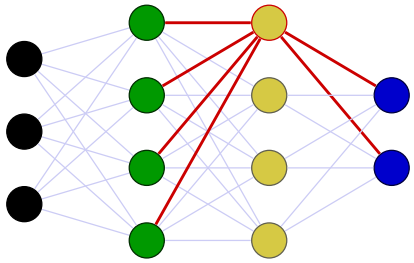


Figure 7.7 Corresponding network, the highlighted part is the pruned one.

ding ANNs in MIPs. After training the ANN (or during training, depending on the technique used), we prune the model before embedding it in the MIP formulation of the problem in hand. This approach either reduces the solution time of the MIP with the same generalisation performances, or, possibly, allows one to include larger, therefore more expressive ANNs, capable of achieving higher accuracy while still maintaining the ability to solve the resulting MIPs within a reasonable time. In particular, we will employ the Structured Perspective Regularization, i.e., we train the ANN by adding the SPR term to the loss, which will lead to a weight tensor with a structured sparsity. After fixing to zero (i.e., removing) neurons whose weights are all below a fixed threshold, we fine-tune the network with a standard loss for a few more epochs (see [23] for details). The obtained ANN is then embedded in the MIP, and it will require the addition of fewer variables and constraints with respect to its unpruned counterpart.

7.5 Experiments

7.5.1 Building adversarial examples

We test the effectiveness of pruning in the task of finding an adversarial example of a given network. In particular, we focus on the verification problem [9], which consists in finding a *slight* modification of an input that is originally correctly classified by the network in such a way that the modified one is assigned to a chosen class by the ANN. More formally, assume we are given a trained ANN $\bar{g}(\cdot) : \mathbb{R}^n \to [0,1]^C$ and one input x such that $\bar{g}(x)$ has its maximum value at the coordinate corresponding to the correct class of x. Denoting with k this coordinate and with h the coordinate with the second highest value of $\bar{g}(x)$, the problem

we want to solve is

$$\max_{\bar{x},y} y_h - y_k \tag{7.10}$$

$$\text{s.t. } y = \bar{g}(\bar{x}) \tag{7.11}$$

$$\Delta \geq x - \bar{x} \tag{7.12}$$

$$\Delta \geq \bar{x} - x \tag{7.13}$$

$$\bar{x} \in \mathbb{R}^n \tag{7.14}$$

where Δ is a given distance bound. Clearly, (7.10)–(7.14) is a special case of the MPLC class (7.1)–(7.4). In particular, the constraint (7.11) encodes an ANN function, so it needs to be handled with the techniques we presented in Section 7.2. We selected this problem since it is of great interest to ML researchers. Furthermore, it can in principle be relevant to test the robustness of networks of any size, and therefore it allows to explore the boundaries of what MPLC approaches (with or without pruning) can achieve.

7.5.2 General setup and notation

To test the effectiveness of our pruning techniques, we ran some experiments on network robustness using the MNIST dataset. We used the same settings of the notebook available at this link, where formulation (7.10)-(7.14) is solved with $\Delta = 5$.

We trained the ANNs using the Pytorch SGD optimizer with no weight decay and no momentum. We used 128 as batch size and we trained the network for 50 epochs with a constant learning rate equal to 0.1. All the networks are Pytorch sequential models containing only Linear and ReLU layers. For the pruned networks, we performed a (limited) 3 by 3 grid search to choose the λ factor that multiplies the SPR term and the α hyper-parameter needed in its definition (M is automatically set as in [23]). After 50 training epochs, the model is fine-tuned for 10 epochs without using any regularization. Note that the objective of the grid search is to find the smallest network that keeps basically the same out-of-sample accuracy of the original one, and better results could conceivably be obtained by employing end-to-end techniques that take into account the optimization process in the computation of the loss [115, 116].

In tables 7.1 and 7.2, the first column reports the network architecture of the used ANN and if pruning was used, while the Δ parameter value of (7.10)-(7.14) can be found in the first row. We compare the result of the baseline approach (i.e., without pruning) and the result obtained using the pruning method with the best hyper-parameters found. We report

the validation accuracy (in percentage), the time needed by Gurobi to solve the obtained MIP (in seconds), and the number of branch-and-bound nodes explored during that time. Additionally, for the pruned networks, we report the value of λ and α and the architecture of the network after pruning. When referring to a network architecture, the terms LxN refer to a sequence of L layers each of them containing N neurons. When multiple terms follow each other, it indicates their order in the network. For example, 2x20-3x10 stands for a network that starts with 2 layers of 20 neurons and continues with 3 layers of 10 neurons. Each experiment is repeated 3 times and a time limit of 1800 seconds is given to Gurobi.

7.5.3 Detailed results

Table 7.1 shows the results using Δ=5 on 4 different architectures with an increasing number of neurons and layers. When pruning small architectures, like the 2x50 and 2x100 networks, pruning the ANN results in at least halving the time used by Gurobi. Moreover, the accuracy of the pruned models is higher than the baseline, this is, likely, because pruning has also a regularization effect.

The results on the 2x200 architecture show that the baseline is not able to solve the problems in the given time for two out of three runs. Instead, our method always leads to MIPs that are easily solved by Gurobi while maintaining the same accuracy as the baseline.

Finally, we report the results using the 6x100 networks, significantly bigger with respect to the previous ones. The baseline, once again, cannot solve two out of the three problems in the given time limit. Instead, our method is able to succeed in all cases, at the cost of losing a little bit of accuracy (0.3 percent in the best case).

As a last remark, we notice that for all the MIPs we solved relatively to unpruned network, no counterexample existed in the given neighborhood (i.e., the optimal value of (7.10)-(7.14) is negative). This remains true for the corresponding pruned counterparts, confirming that the pruned and unpruned versions of the MIPs are qualitatively very similar.

7.5.4 Investigating the quality of the solutions

To better validate the quality of our results, we solved again the adversarial problem (7.10)-(7.14) using Δ=20 and employing the same networks trained in the previous experiments. This was aimed to find adversarial examples in the given region to better understand the effect of pruning on the resulting MIP. We report the results in Table 7.2, where the “accuracy" and “pruned architecture" columns have been removed since they are the same as in the previous table. For all the experiments, a counter-example existed in the given region, and

Table 7.1 Results using $\Delta = 5$.

$\Delta = 5$					
Arch	λ-α	Acc.	Time	Nodes	Pruned Arch
2x50 Baseline		97.55	14.88	5820	
		97.47	20.65	12040	
		97.25	8.07	9497	
2x50 Pruned	0.5-0.9	97.77	3.29	3328	1x39-1x43
		97.49	3.93	6482	1x30-1x42
		97.73	1.96	3992	1x39-1x42
2x100 Baseline		97.96	39.29	2971	
		97.76	35.01	3112	
		97.97	39.00	3019	
2x100 Pruned	0.5-0.9	98.08	15.99	3066	1x61-1x80
		98.01	15.65	3107	1x61-1x87
		98.04	17.67	2951	1x63-1x86
2x200 Baseline		98.14	1800.37	424758	
		98.04	1800.18	401361	
		97.95	781.90	58656	
2x200 Pruned	0.5-0.5	97.96	18.66	3029	1x56-1x144
		98.13	24.65	3600	1x57-1x144
		98.04	28.19	2997	1x59-1x140
6x100 Baseline		97.60	474,76	15261	
		97.77	1800.02	798306	
		97.67	818.19	14334	
6x100 Pruned	1.0-0.1	97.52	231.02	3173	1x39-1x82 2x61-1x60-1x54
		97.47	79.22	7566	1x44-1x71-1x49 -1x53-1x49-1x45
		97.21	44.24	11417	1x37-1x72-1x48 -1x51-1x48-1x46

in the last column of Table 7.2, named “Found", we report if Gurobi was able to find one adversarial example in the given time limit. Unsurprisingly, for all the MIPs corresponding to pruned networks, Gurobi was able to find an adversarial example within a time considerably inferior to the 1800 seconds limit. Moreover, all the adversarial examples obtained using a pruned network were also adversarial for the unpruned counterpart with the same starting architecture. This empirically proved that, in our setting, pruning can be even used to solve the adversarial example problem for the unpruned counterpart and it is again a good indication that pruning does not heavily affect the resulting MIP. This is in accordance with the ML literature, where there is a good consensus that not-too-aggressive pruning of ANNs does not significantly impacts their robustness [8, 167], and therefore the existence—or not—of the counter-example in our application. Finally, the times reported in Table 7.2 show that the speed-up is still very significant even with the new value of Δ and that in some cases the Baseline is not able to find any adversarial example.

We conclude this section by noting that additional experiments, which are not included in

this paper for the sake of brevity, have shown that a high setting of the OBBT parameter [46] of Gurobi is crucial to obtain good performances both for pruned and unpruned instances, confirming the importance of structured pruning.

Table 7.2 Results using $\Delta = 20$.

$\Delta = 20$				
Arch	λ-α	Time	Nodes	Found
2x50 Baseline		1.85	1	YES
		5.06	1221	YES
		1.66	1	YES
2x50 Pruned	0.5-0.9	2.73	127	YES
		2.90	1128	YES
		0.64	1	YES
2x100 Baseline		17.35	1217	YES
		147.67	7532	YES
		102.67	3252	YES
2x100 Pruned	0.5-0.9	6.66	2079	YES
		6.03	127	YES
		2.16	1	YES
2x200 Baseline		439.17	40075	YES
		563.24	17597	YES
		508.14	6014	YES
2x200 Pruned	0.5-0.5	2.56	1	YES
		18.65	5433	YES
		9.60	1202	YES
6x100 Baseline		1800.06	138918	NO
		1800.03	237328	NO
		1800.10	184954	NO
6x100 Pruned	1.0-0.1	15.53	1	YES
		7.51	1	YES
		129.70	27045	YES

7.6 Conclusions and future directions

This paper has demonstrated the effectiveness of pruning artificial neural networks in accelerating the solution time of mixed-integer programming problems that incorporate ANNs. The choice of the sparsity structure for pruning plays a crucial role in achieving significant speed-up, and we argued that structured pruning is superior to unstructured one. Further research in this area can focus on gaining a deeper understanding of which sparsity structures are most suitable for improving the solution time of MIPs. Exploring the trade-off between pruning-induced sparsity and solution quality is another interesting avenue for future investigations. By advancing our understanding of pruning techniques and their impact on MIPs, we can enhance the efficiency and scalability of embedding ANNs in optimization problems.

Acknowledgments

The authors are grateful to Pierre Bonami for his generous and insightful feedback. This work has been supported by the NSERC Alliance grant 544900- 19 in collaboration with Huawei-Canada

CHAPTER 8 GENERAL DISCUSSION

The studies presented in Chapter 5 and 6 examine two different Model Compression methods, while Chapter 7 introduces a new pipeline that combines Pruning and Constraints Learning. Although these approaches tackle the compression challenge from different angles, the first two contributions do share certain common elements, and the final work is closely related to the first one. In the following sections, we explore and discuss these points of convergence and reflect on the thesis contribution as a whole.

The objective of this thesis was to craft *MC* methodologies by exploiting the capabilities of *OR* techniques. We introduced approaches that are not only theoretically sound but also amenable to explanation. Broadly speaking, this serves as a demonstration of the synergistic collaboration between *OR* and *ML*, showcasing how these two disciplines can unite to yield remarkable outcomes.

In Chapter 5, we developed a pruning method using a *MIP* model of the problem and then strengthened this formulation before (approximately) solving its linear relaxation. This led to the *SPR*, a novel structured sparsity-inducing regularization term. The *OR* technique used to obtain the stronger formulation is proven to provide better results, optimal in some cases, in the sense that the derived formulation is the *tightest* possible. Moreover, the *SPR* has some partially-novel properties that can help improve our understanding of the pruning dynamics. Finally, the experimental results are very encouraging, being competitive with the *SOTA* and improving it in different cases.

In Chapter 6, we provided theoretical results for the convergence of low-bit quantized *NNs* training. As in the previous chapter, we made use of *OR* tools, the quasi-convex theory in this case. We were able to find the provable optimal step size and provide some theorems estimating the expected error convergence depending on the used number format. We successfully extended our results to the stochastic setting, giving practitioners reliable and practical tools to choose a number format suited to their needs. This work also significantly helped improve the general understanding of quantized Neural Networks' properties and their behavior when trained.

Lastly, Chapter 7 explored the application of Pruning to accelerate *OR* solvers within the realm of *CL*. Notably, the pruning technique employed here is the very same one developed in Chapter 5. This concluding chapter provided a satisfying sense of closure, as it completes a cyclic journey. Initially, we harnessed *OR* techniques to address Pruning, a challenge within the purview of *ML*. Subsequently, this very *ML* technique was repurposed as a speed-

enhancing tool in *CL*, which is an *OR* problem. Both the first and third contributions of this work exemplify how *OR* and *ML* can work hand in hand, showcasing their mutual compatibility and potential for collaboration. Once again, our exploration of Pruning has deepened, this time by investigating its impact on *NN* embedded in *MIP* models. This highlights the value of uniting *OR* and *ML*, shedding light on new perspectives and encouraging further joint exploration.

CHAPTER 9 CONCLUSIONS AND RECOMMENDATIONS

This thesis introduced an Operations Research approach to the realm of Model Compression, demonstrating how the utilization of theoretically rooted methodologies from *OR* within the domain of Machine Learning can yield diverse advantages. Furthermore, we extended our exploration by presenting an innovative application wherein an *ML* technique developed using OR tools is subsequently employed in a context where an ML structure aids *OR*. In the subsequent sections of this chapter, we provide succinct summaries of each of our contributions and subsequently delve into discussions surrounding limitations and potential avenues for future research.

9.1 Summary of Works

In our first contribution, we tackled the pruning problem, focusing on reducing the memory used to store the model as well as effectively lightening the computational resources needed to run the model. In the *MIP* model we formulate, we achieved the first objective by penalizing the number of parameters assuming a value different from zero, while the second goal is pursued by enforcing a structured sparsity that is exploitable by GPUs. We proceeded by using the Perspective function to obtain a provable better reformulation of the initial model. Finally, we projected away a group of variables to obtain a new regularization term that we call *SPR*. We analyzed the properties of the *SPR* in different regions of the space, deriving from it some enlightening insights on its functioning. We concluded this work with an extensive experimental section, showing results more than competitive with the *SOTA* across different *NN* architectures and datasets.

Chapter 6 investigated the topic of low-bit quantization training under novel perspectives. We relaxed the convexity hypothesis into quasi-convexity, being the first one to study such a class of functions in this context. We focused on the convergence properties of *NN* that are trained using low-bit number formats, considering numerical errors both in gradient computation and weights update. We provided a deterministic analysis that bounds the convergence error and we derived the optimal step size that minimizes it. We generalized these results to the stochastic setting making them practically usable for practitioners, giving them the possibility to predict what errors to expect based on the used number format and to choose an appropriate step size. We concluded with an experimental section on both artificial and real-world data, confirming the correctness of our results.

In Chapter 7, we explored a novel connection between Pruning and Constraints Learning. Given that the main bottleneck of *CL* when using *NN* is scalability, we realized that *MC* techniques could be used in this context despite being designed for different purposes. We were able to identify Pruning as the most promising *MC* technique in this setting and to select neuron pruning as the most appropriate pruning methodology to enhance a *MIP* solver performance. We empirically proved the validity of our approach in the relevant application of Adversarial Example Search, showing a massive solving time reduction and that the optimal solution quality is not affected while pruning the *NN*.

9.2 Limitations and Future Research

The promising outcomes achieved through the application of the *SPR* for structured pruning are indeed encouraging, yet they do not preclude the potential for further enhancements. Our current approach involves solving the linear relaxation of our formulation, necessitating approximations due to the non-convex nature of *NNs*. Exploring more sophisticated strategies is conceivable. For instance, one avenue entails developing a Branch-and-Bound scheme that iteratively solves the relaxation, each time fixing distinct integer variables. In parallel, an approach guided by the reduced costs of integer variables could aid in determining which prunable entities should be removed. These potential directions hold promise for refining our methodology and achieving even more compelling results.

While our method refrains from assuming any specific *NN* architecture, there are instances where tailoring a general approach to specific contexts proves advantageous. For example, we envision the potential for successful adaptations to Recurrent Neural Networks and Graph Neural Networks. Naturally, such adaptations would necessitate substantial implementation effort. As previously highlighted in Chapter 5, it is important to acknowledge that conducting experiments can be exceedingly time-consuming due to the sheer scale of prevalent *NNs* and datasets. However, the prospect of adapting and refining our approach to address specialized scenarios remains a promising avenue for future exploration.

As it is common in the *ML* field, we used a grid search to tune the hyper-parameters that appear in the definition of the *SPR*. This type of search is known to be sub-optimal and could be improved by using more sophisticated direct-search algorithms like MADS [7].

In Chapter 6, we make the assumption that the trained function is quasi-convex, a condition that, although less stringent than full convexity, is typically satisfied only in local regions for the majority of *NNs*. Extending our findings to encompass various function classes poses challenges, primarily owing to the nature of our analysis. Nevertheless, these results could

serve as a foundational guide or an indicative framework for anticipating the behavior during the training of quantized *NN* models, offering valuable insights into potential outcomes.

An external constraint that has limited our analysis pertains to the scarcity of gradient error bounds that are tailored specifically to *NNs*. A thorough exploration of these error bounds would undoubtedly prove immensely advantageous for the broader realm of *ML*, significantly expediting advancements in the domain of quantized *NNs*.

The experimental section in Chapter 6 compellingly validates the accuracy and real-world applicability of our findings. Nonetheless, a more extensive set of experiments involving larger networks and datasets would undeniably enhance the comprehensiveness of our analysis.

The methodologies employed in the first two contributions of this thesis are inherently different from each other. However, a link emerges, particularly in the context of quantizing a *NN* with binary numbers, which can be perceived as a special instance of Pruning. This observation potentially paves the way for the creation of hybrid techniques that leverage a fusion of Pruning and Quantization, exploiting the opportunity to treat one as a specialized iteration of the other.

In Chapter 7, we argued for the advantageous integration of neuron pruning within the context of *CL*, since this reduces the number of variables and constraints of the model. However, this is not the only approach to achieve faster solution times. It is well known in the *OR* field that problems whose constraint matrices have specific structures can be solved extremely efficiently (e.g., Benders decomposition). The same approach is possible for *NNs*, finding what architectures lead to *MIP* models with exploitable structures is surely an exciting direction. This would also require the development of new pruning methods that produce *NNs* with such architectures, leaving plenty of space for more contributions.

It is also possible an approach that proceeds in reverse order. Analyzing the different structures of *MIP* formulations that are possible to obtain when pruning *NNs* with different methods, we could develop new *OR* techniques exploiting those structures to accelerate the solution process.

We decided to use Pruning in our third contribution since it is a natural candidate to be applied in *CL* and we have already gained a deep understanding of this technique through the development of the first contribution. Observing the impressive results we achieved, we are naturally encouraged to repeat the same process with different *MC* techniques or to design new ones specific to the *CL* context.

REFERENCES

[1] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-trim: Convex pruning of deep neural networks with performance guarantee," *Advances in neural information processing systems*, vol. 30, 2017.

[2] A. Aghasi, A. Abdi, and J. Romberg, "Fast convex pruning of deep neural networks," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 1, pp. 158–188, 2020.

[3] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learned compression of images and neural networks," *CoRR*, vol. abs/1704.00648, 2017. [Online]. Available: http://arxiv.org/abs/1704.00648

[4] J. M. Alvarez and M. Salzmann, "Compression-aware training of deep networks," *Advances in neural information processing systems*, vol. 30, 2017.

[5] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Mathematical Programming*, vol. 183, no. 1-2, pp. 3–39, 2020.

[6] A. Atamtürk and A. Gómez, "Safe screening rules for l0-regression from perspective relaxations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 421–430.

[7] C. Audet and J. E. Dennis, "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, 2006. [Online]. Available: https://doi.org/10.1137/040603371

[8] M. Ayle, B. Charpentier, J. Rachwan, D. Zügner, S. Geisler, and S. Günnemann, "On the robustness and anomaly detection of sparse neural networks," *arXiv preprint arXiv:2207.04227*, 2022.

[9] B. Batten, P. Kouvaros, A. Lomuscio, and Y. Zheng, "Efficient neural network verification via layer-based semidefinite relaxations and linear cuts." in *IJCAI*, 2021, pp. 2184–2190.

[10] C. Baykal, L. Liebenwein, I. Gilitschenski, D. Feldman, and D. Rus, "Sensitivity-informed provable pruning of neural networks," *SIAM Journal on Mathematics of Data Science*, vol. 4, no. 1, pp. 26–45, 2022.

[11] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BJ_wN01C-

[12] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013. [Online]. Available: http://arxiv.org/abs/1308.3432

[13] D. Bergman, T. Huang, P. Brooks, A. Lodi, and A. U. Raghunathan, "Janos: An integrated predictive and prescriptive modeling framework," *INFORMS Journal on Computing*, vol. 34, no. 2, pp. 807–816, 2022. [Online]. Available: https://doi.org/10.1287/ijoc.2020.1023

[14] D. Bertsimas, A. O'Hair, S. Relyea, and J. Silberholz, "An analytics approach to designing combination chemotherapy regimens for cancer," *Management Science*, vol. 62, no. 5, pp. 1511–1531, 2016.

[15] D. Bienstock, G. Muñoz, and S. Pokutta, "Principled deep neural network training through linear programming," *arXiv preprint arXiv:1810.03218*, 2018.

[16] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *Proceedings of machine learning and systems*, vol. 2, pp. 129–146, 2020.

[17] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese, "On mathematical programming with indicator constraints," *Mathematical programming*, vol. 151, pp. 191–223, 2015.

[18] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186.

[19] P. S. Bradley and O. L. Mangasarian, "Feature selection via concave minimization and support vector machines." in *ICML*, vol. 98, 1998, pp. 82–90.

[20] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in neural information processing systems*, vol. 2, 1989.

[21] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[22] M. Cacciola, A. Frangioni, X. Li, and A. Lodi, "Deep neural networks pruning via the structured perspective regularization," in *OPT2021: 13th Annual Workshop on Optimization for Machine Learning*, 2021.

[23] ——, "Deep neural networks pruning via the structured perspective regularization," 2022. [Online]. Available: https://arxiv.org/abs/2206.14056

[24] M. Cacciola, A. Frangioni, M. Asgharian, A. Ghaffari, and V. Partovi Nia, "On the convergence of stochastic gradient descent in low-precision number formats," in *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods - ICPRAM*, INSTICC. SciTePress, 2023, pp. 542–549.

[25] M. Cacciola, A. Frangioni, and A. Lodi, "Structured pruning of neural networks for constraints learning," *arXiv preprint arXiv:2307.07457*, 2023.

[26] M. A. Carreira-Perpinan and Y. Idelbayev, ""learning-compression" algorithms for neural net pruning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8532–8541.

[27] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener, "Omlt: Optimization & machine learning toolkit," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 15 829–15 836, 2022.

[28] J. Chee, M. Renz, A. Damle, and C. D. Sa, "Model preserving compression for neural networks," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=gt-l9Hu2ndd

[29] T. Chen, B. Ji, T. Ding, B. Fang, G. Wang, Z. Zhu, L. Liang, Y. Shi, S. Yi, and X. Tu, "Only train once: A one-shot neural network training and pruning framework," in *NeurIPS*, 2021.

[30] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1518–1528.

[31] A. Cichocki and S.-i. Amari, "Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities," *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[32] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.

[33] R. Dabre and A. Fujita, "Recurrent stacking of layers for compact neural machine translation models," *CoRR*, vol. abs/1807.05353, 2018. [Online]. Available: http://arxiv.org/abs/1807.05353

[34] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "BNN+: improved binary network training," *CoRR*, vol. abs/1812.11800, 2018. [Online]. Available: http://arxiv.org/abs/1812.11800

[35] X. Ding, g. ding, X. Zhou, Y. Guo, J. Han, and J. Liu, "Global sparse momentum sgd for pruning very deep neural networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/f34185c4ca5d58e781d4f14173d41e5d-Paper.pdf

[36] H. Dong, K. Chen, and J. Linderoth, "Regularization vs. relaxation: A convexification perspective of statistical variable selection," 2018.

[37] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[38] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *CoRR*, vol. abs/1705.07565, 2017. [Online]. Available: http://arxiv.org/abs/1705.07565

[39] C. D'Ambrosio, A. Frangioni, and C. Gentile, "Strengthening the sequential convex minlp technique by perspective reformulations," *Optimization letter 13, Springer*, vol. 13, 2019. [Online]. Available: https://doi.org/10.1007/s11590-018-1360-9

[40] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun, "Understanding deep architectures using a recursive convolutional network," *CoRR*, 12 2013. [Online]. Available: https://arxiv.org/abs/1312.1847

[41] M. ElAraby, G. Wolf, and M. Carvalho, "Oamip: Optimizing ann architectures using mixed-integer programming," in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, A. A. Cire, Ed. Cham: Springer Nature Switzerland, 2023, pp. 219–237.

[42] L. Enderich, F. Timm, and W. Burgard, "Holistic filter pruning for efficient deep neural networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2596–2605.

[43] I. Fahmi and S. Cremaschi, "Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models," *Computers & Chemical Engineering*, vol. 46, pp. 105–123, 2012.

[44] F. A. Fernandes, "Optimization of fischer-tropsch synthesis using neural networks," *Chemical Engineering & Technology: Industrial Chemistry-Plant Equipment-Process Engineering-Biotechnology*, vol. 29, no. 4, pp. 449–453, 2006.

[45] K. J. Ferreira, B. H. A. Lee, and D. Simchi-Levi, "Analytics for an online retailer: Demand forecasting and price optimization," *Manufacturing & service operations management*, vol. 18, no. 1, pp. 69–88, 2016.

[46] M. Fischetti and J. Jo, "Deep neural networks and mixed integer linear optimization," *Constraints*, vol. 23, no. 3, pp. 296–309, 2018.

[47] A. Frangioni, C. Gentile, E. Grande, and A. Pacifici, "Projected perspective reformulations with applications in design problems," *Operations Research*, vol. 59, no. 5, pp. 1225–1232, 2011.

[48] A. Frangioni, F. Furini, and C. Gentile, "Approximated perspective relaxations: a project&lift approach," *Computational Optimization and Applications*, vol. 63, no. 3, pp. 705–735, 2016.

[49] ——, "Improving the approximated projected perspective reformulation by dual information," *Operations Research Letters*, vol. 45, no. 5, pp. 519–524, 2017.

[50] A. Frangioni and C. Gentile, "Perspective cuts for a class of convex 0–1 mixed integer programs," *Mathematical Programming*, vol. 106, no. 2, pp. 225–236, 2006.

[51] 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

[52] E. Frantar and D. Alistarh, "Optimal brain compression: A framework for accurate post-training quantization and pruning," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=ksVGCOlOEba

[53] ——, "SPDY: Accurate pruning with speedup guarantees," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 6726–6743. [Online]. Available: https://proceedings.mlr.press/v162/frantar22a.html

[54] T. Furuya, K. Suetake, K. Taniguchi, H. Kusumoto, R. Saiin, and T. Daimon, "Spectral pruning for recurrent neural networks," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 3458–3482.

[55] N. Gamboa, K. Kudrolli, A. Dhoot, and A. Pedram, "Campfire: Compressible, regularization-free, structured sparse training for hardware accelerators," *arXiv preprint arXiv:2001.03253*, 2020.

[56] A. Ghaffari, M. S. Tahaei, M. Tayaranian, M. Asgharian, and V. Partovi Nia, "Is integer arithmetic enough for deep learning training?" *Advances in Neural Information Processing Systems*, 2022, to appear.

[57] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[58] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[59] M. Golub, G. Lemieux, and M. Lis, "Dropback: Continuous pruning during training," *CoRR*, vol. abs/1806.06949, 2018. [Online]. Available: http://arxiv.org/abs/1806.06949

[60] P. Goyal, Q. Duval, I. Seessel, M. Caron, M. Singh, I. Misra, L. Sagun, A. Joulin, and P. Bojanowski, "Vision models are more robust and fair when pretrained on uncurated images without supervision," *arXiv preprint arXiv:2202.08360*, 2022.

[61] B. Grimstad and H. Andersson, "Relu networks as surrogate models in mixed-integer linear programs," *Computers & Chemical Engineering*, vol. 131, p. 106580, 2019.

[62] N. S. Guang-Bin Huang, P. Saratchandran, "A Generalized Growing and Pruning RBF (GGAP-RBF) Neural Network for Function Approximation," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 16, no. 1, pp. 57–67, Jan 2005.

[63] M. E. Halabi, S. Srinivas, and S. Lacoste-Julien, "Data-efficient structured pruning via submodular optimization," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=K2QGzyLwpYG

[64] S. Han, "Efficient methods and hardware for deep learning," Ph.D. dissertation, Stanford University, 2017.

[65] 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 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1135–1143. [Online]. Available: http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network.pdf

[66] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5, [NIPS Conference]*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, p. 164–171.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[68] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[69] Y. He and S. Han, "ADC: automated deep compression and acceleration with reinforcement learning," *CoRR*, vol. abs/1802.03494, 2018. [Online]. Available: http://arxiv.org/abs/1802.03494

[70] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[71] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *CoRR*, vol. abs/2102.00554, 2021. [Online]. Available: https://arxiv.org/abs/2102.00554

[72] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[73] Y. Hu, X. Yang, and C.-K. Sim, "Inexact subgradient methods for quasi-convex optimization problems," *European Journal of Operational Research*, vol. 240, no. 2, pp. 315–327, 2015.

[74] Y. Hu, C. Yu, and C. Li, "Stochastic subgradient method for quasi-convex optimization problems," *Journal of nonlinear and convex analysis*, vol. 17, pp. 711–724, 01 2016.

[75] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Computer Vision–ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI 15*. Springer, 2018, pp. 317–334.

[76] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[77] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[78] J. Huchette, G. Muñoz, T. Serra, and C. Tsay, "When deep learning meets polyhedral theory: A survey," *arXiv preprint arXiv:2305.00241*, 2023.

[79] S. ichi Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4, pp. 185–196, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092523129390006O

[80] G. Iommazzo, C. D'Ambrosio, A. Frangioni, and L. Liberti, "A learning-based mathematical programming formulation for the automatic configuration of optimization solvers," in *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part I 6*. Springer, 2020, pp. 700–712.

[81] G. Iommazzo, C. d'Ambrosio, A. Frangioni, and L. Liberti, "Learning to configure mathematical programming solvers by mathematical programming," in *Learning and Intelligent Optimization: 14th International Conference, LION 14, Athens, Greece, May 24–28, 2020, Revised Selected Papers 14*. Springer, 2020, pp. 377–389.

[82] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[83] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *CoRR*, vol. abs/1405.3866, 2014. [Online]. Available: http://arxiv.org/abs/1405.3866

[84] E. Karnin, "Simple procedure for pruning back-propagation trained neural networks," *Neural Networks, IEEE Transactions on*, vol. 1, pp. 239 – 242, 07 1990.

[85] K. Kiwiel and K. Murty, "Convergence of the steepest descent method for minimizing quasiconvex functions," *Journal of Optimization Theory and Applications*, vol. 89, 04 1996.

[86] K. C. Kiwiel, "Convergence and efficiency of subgradient methods for quasiconvex minimization," *Mathematical Programming*, vol. 90, pp. 1–25, 2001.

[87] A. Krizhevsky, "Learning multiple layers of features from tiny images," (canadian institute for advanced research), Tech. Rep., 2009.

[88] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[89] S. Kundu, M. Nazemi, P. A. Beerel, and M. Pedram, "Dnr: A tunable robust pruning framework through dynamic network rewiring of dnns," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 344–350. [Online]. Available: https://doi.org/10.1145/3394885.3431542

[90] S. Lambert-Lacroix and L. Zwald, "The adaptive berhu penalty in robust regression," *Journal of Nonparametric Statistics*, vol. 28, no. 3, pp. 487–514, 2016.

[91] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2554–2564.

[92] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[93] Y. LeCun, J. Denker, S. Solla, R. Howard, and L. Jackel, "Optimal brain damage," in *Advances in Neural Information Processing Systems (NIPS 1989), Denver, CO*, D. Touretzky, Ed., vol. 2. Morgan Kaufmann, 1990.

[94] E. Lee and C.-Y. Lee, "Neuralscale: Efficient scaling of neurons for resource-constrained deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1478–1487.

[95] N. Lee, T. Ajanthan, and P. Torr, "SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1VZqjAcYX

[96] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with admm," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11713

[97] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *CoRR*, vol. abs/1608.08710, 2016. [Online]. Available: http://arxiv.org/abs/1608.08710

[98] ——, "Pruning filters for efficient convnets," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rJqFGTslg

[99] X. Li, B. Liu, Y. Yu, W. Liu, C. Xu, and V. Partovi Nia, "S3: Sign-sparse-shift reparametrization for effective training of low-bit shift networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 555–14 566, 2021.

[100] Y. Li, S. Gu, K. Zhang, L. V. Gool, and R. Timofte, "Dhp: Differentiable meta pruning via hypernetworks," in *European Conference on Computer Vision*. Springer, 2020, pp. 608–624.

[101] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. E. Gonzalez, "Train large, then compress: rethinking model size for efficient training and inference of transformers," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 5958–5968.

[102] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, "Provable filter pruning for efficient neural networks," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJxkOlSYDH

[103] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1529–1538.

[104] 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*, vol. 31, no. 2, pp. 574–588, 2019.

[105] T. Lin, S. U. Stich, L. Barba, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SJem8lSFwB

[106] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[107] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[108] 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

[109] A. Lodi, M. Toma, and R. Guerrieri, "Very low complexity prompted speaker verification system based on hmm-modeling," *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, vol. 4, 01 2002.

[110] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," *Advances in neural information processing systems*, vol. 30, 2017.

[111] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1Y8hhg0b

[112] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[113] Y. Ma, R. Chen, W. Li, F. Shang, W. Yu, M. Cho, and B. Yu, "A unified approximation framework for compressing and accelerating deep neural networks," in *2019 IEEE 31st*

International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019, pp. 376–383.

[114] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 181–196.

[115] J. Mandi, E. Demirović, P. J. Stuckey, and T. Guns, "Smart predict–and–optimize for hard combinatorial optimization problems," in *Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 1603–1610.

[116] J. Mandi, V. Bucarey, M. M. K. Tchomba, and T. Guns, "Decision-focused learning: through the lens of learning to rank," in *International Conference on Machine Learning*. PMLR, 2022, pp. 14 935–14 947.

[117] D. Maragno, H. Wiberg, D. Bertsimas, S. I. Birbil, D. d. Hertog, and A. Fajemisin, "Mixed-integer optimization with constraint learning," *arXiv preprint arXiv:2111.04469*, 2021.

[118] D. Molchanov, D. Vetrov, and A. Ashukha, "Variational dropout sparsifies deep neural networks," in *34th International Conference on Machine Learning, ICML 2017*, 2017, pp. 3854–3863.

[119] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019.

[120] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 264–11 272.

[121] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1. Morgan-Kaufmann, 1989. [Online]. Available: https://proceedings.neurips.cc/paper/1988/file/07e1cd7dca89a1678042477183b7ac3f-Paper.pdf

[122] V. P. Nia and M. Belbahri, "Binary quantizer," *Journal of Computational Vision and Imaging Systems*, vol. 4, no. 1, p. 3, Dec. 2018. [Online]. Available: https://openjournals.uwaterloo.ca/index.php/vsl/article/view/334

[123] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Computation*, vol. 4, no. 4, pp. 473–493, 1992.

[124] A. B. Owen, "A robust hybrid of lasso and ridge regression," *Contemporary Mathematics*, 2007. [Online]. Available: http://finmath.stanford.edu/~owen/reports/hhu.pdf

[125] B. Polyak, "A general method for solving extremum problems," *Soviet Mathematics. Doklady*, vol. 8, 01 1967.

[126] J. Rachwan, D. Zügner, B. Charpentier, S. Geisler, M. Ayle, and S. Günnemann, "Winning the lottery ahead of time: Efficient early network pruning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 18 293–18 309.

[127] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[128] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009. [Online]. Available: https://doi.org/10.1137/080726380

[129] R. K. Ramakrishnan, E. Sari, and V. P. Nia, "Differentiable mask for pruning convolutional and recurrent networks," in *2020 17th Conference on Computer and Robot Vision (CRV)*. IEEE, 2020, pp. 222–229.

[130] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[131] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[132] M. Schmidt, N. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[133] T. Serra, C. Tjandraatmadja, and S. Ramalingam, "Bounding and counting linear regions of deep neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4558–4566.

[134] T. Serra, A. Kumar, and S. Ramalingam, "Lossless compression of deep neural networks," in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings*. Springer, 2020, pp. 417–430.

[135] T. Serra, X. Yu, A. Kumar, and S. Ramalingam, "Scaling up exact neural network compression by relu stability," *Advances in neural information processing systems*, vol. 34, pp. 27 081–27 093, 2021.

[136] M. Shen, H. Yin, P. Molchanov, L. Mao, J. Liu, and J. M. Alvarez, "Structural pruning via latency-saliency knapsack," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=cUOR-_VsavA

[137] X. Sheng, C. Hanlin, G. Xuan, L. Kexin, L. Jinhu, and Z. Baochang, "Efficient structured pruning based on deep feature stabilization," *Neural Computing and Applications*, vol. 1433-3058, 2021. [Online]. Available: https://doi.org/10.1007/s00521-021-05828-8

[138] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[139] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[140] S. P. Singh and D. Alistarh, "Woodfisher: Efficient second-order approximations for model compression," *CoRR*, vol. abs/2004.14340, 2020. [Online]. Available: https://arxiv.org/abs/2004.14340

[141] Y. Sui, M. Yin, Y. Xie, H. Phan, S. Aliari Zonouz, and B. Yuan, "Chip: Channel independence-based pruning for compact neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 604–24 616, 2021.

[142] Y. Tang, Y. Wang, Y. Xu, D. Tao, C. XU, C. Xu, and C. Xu, "Scop: Scientific control for reliable neural network pruning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 10 936–10 947. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/7bcdf75ad237b8e02e301f4091fb6bc8-Paper.pdf

[143] A. Thebelt, J. Kronqvist, M. Mistry, R. M. Lee, N. Sudermann-Merx, and R. Misener, "Entmoot: A framework for optimization over ensemble tree models," *Computers & Chemical Engineering*, vol. 151, p. 107343, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098135421001216

[144] L. Theis, I. Korshunova, A. Tejani, and F. Huszár, "Faster gaze prediction with dense networks and fisher pruning," *CoRR*, vol. abs/1801.05787, 2018. [Online]. Available: http://arxiv.org/abs/1801.05787

[145] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HyGIdiRqtm

[146] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[147] M.-R. Vemparala, N. Fasfous, A. Frickenstein, M. A. Moraly, A. Jamal, L. Frickenstein, C. Unger, N.-S. Nagaraja, and W. Stechele, "L2pf-learning to prune faster," in *Computer Vision and Image Processing: 5th International Conference, CVIP 2020, Prayagraj, India, December 4-6, 2020, Revised Selected Papers, Part III*. Springer, 2021, pp. 249–261.

[148] Y. Wang, X. Zhang, L. Xie, J. Zhou, H. Su, B. Zhang, and X. Hu, "Pruning from scratch," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 12 273–12 280, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6910

[149] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[150] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.

[151] J. Wu, "Introduction to convolutional neural networks," *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, no. 23, p. 495, 2017.

[152] X. XIAO, Z. Wang, and S. Rajasekaran, "Autoprune: Automatic network pruning by regularizing auxiliary parameters," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer,

F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/4efc9e02abdab6b6166251918570a307-Paper.pdf

[153] H. Yang, W. Wen, and H. Li, "Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures," in *International Conference on Learning Representations*, 2020.

[154] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[155] J. Ye, L. Wang, G. Li, D. Chen, S. Zhe, X. Chu, and Z. Xu, "Learning compact recurrent neural networks with block-term tensor decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[156] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9194–9203.

[157] S. Yu, A. Mazaheri, and A. Jannesari, "Topology-aware network pruning using multi-stage graph embedding and reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25 656–25 667.

[158] X. Yu, T. Serra, S. Ramalingam, and S. Zhe, "The combinatorial brain surgeon: Pruning weights that cancel one another in neural networks," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 25 668–25 683. [Online]. Available: https://proceedings.mlr.press/v162/yu22f.html

[159] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 104–12 113.

[160] R. Zhang, A. G. Wilson, and C. De Sa, "Low-precision stochastic gradient langevin dynamics," in *International Conference on Machine Learning*. PMLR, 2022, pp. 26 624–26 644.

[161] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 184–199.

[162] X. Zhang, S. Liu, R. Zhang, C. Liu, D. Huang, S. Zhou, J. Guo, Q. Guo, Z. Du, T. Zhi *et al.*, "Fixed-point back-propagation training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2330–2338.

[163] Y. Zhang, Y. Yao, P. Ram, P. Zhao, T. Chen, M. Hong, Y. Wang, and S. Liu, "Advancing model pruning via bi-level optimization," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=t6O08FxvtBY

[164] K. Zhao, S. Huang, P. Pan, Y. Li, Y. Zhang, Z. Gu, and Y. Xu, "Distribution adaptive int8 quantization for training cnns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, 2021, pp. 3483–3491.

[165] Y. Zhou, Y. Zhang, Y. Wang, and Q. Tian, "Accelerate cnn via recursive bayesian pruning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3306–3315.

[166] T. Zhuang, Z. Zhang, Y. Huang, X. Zeng, K. Shuang, and X. Li, "Neuron-level structured pruning using polarization regularizer," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 9865–9877. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/703957b6dd9e3a7980e040bee50ded65-Paper.pdf

[167] X. Zhuang, Y. Ge, B. Zheng, and Q. Wang, "Adversarial network pruning by filter robustness estimation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[168] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[169] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[170] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B*, vol. 67, no. 2, pp. 301–320, 2005.

APPENDIX A DEEP NEURAL NETWORKS PRUNING VIA THE STRUCTURED PERSPECTIVE REGULARIZATION

Authors: Matteo Cacciola, Antonio Frangioni, Xinlin Li, and Andrea Lodi.
Published at OPT2021: 13th Annual Workshop on Optimization for Machine Learning (13/12/2021)[1].

Abstract In Machine Learning, Artificial Neural Networks (ANNs) are a very powerful tool, broadly used in many applications. Often, the selected (deep) architectures include many layers, and therefore a large amount of parameters, which makes training, storage and inference expensive. This motivated a stream of research about compressing the original networks into smaller ones without excessively sacrificing performances. Among the many proposed compression approaches, one of the most popular is *pruning*, whereby entire elements of the ANN (links, nodes, channels, etc.) and the corresponding weights are deleted. Since the nature of the problem is inherently combinatorial (what elements to prune and what not), we propose a new pruning method based on Operations Research tools. We start from a natural Mixed-Integer Programming model for the problem, and we use the Perspective Reformulation technique to strengthen its continuous relaxation. Projecting away the indicator variables from this reformulation yields a new regularization term, which we call the Structured Perspective Regularization. That leads to structured pruning of the initial architecture. We test our method on some ResNet architectures applied to CIFAR-10, CIFAR-100 and ImageNet datasets, obtaining competitive performances w.r.t. the state of the art for structured pruning.

A.1 Introduction

The striking practical success of Artificial Neural Networks (ANN) has been initially driven by the ability of adding more and more parameters to the models, which has led to vastly increased accuracy. This brute-force approach, however, has numerous drawbacks: besides the ever-present risk of overfitting, massive models are costly to store and run. This clashes with the ever increasing push towards edge computing of ANN, whereby neural models have to be run on low power devices such as smart phones, smart watches, and wireless base stations [65, 96, 109]. While one may just resort to smaller models, the fact that a large mode trained even for a few epochs performs better than smaller ones trained for much

[1]The full reference is [22].

longer lends credence to the claim [101] that the best strategy is to initially train large and over-parameterized models and then shrink them through techniques such as pruning and low-bit quantization.

A relevant aspect of the process is the choice of the elements to be pruned. Owing to the fact that both ANN training and inference is nowadays mostly GPU-based, pruning an individual weight may yield little to no benefit in case other weights in the same "computational block" are retained, as the vector processing nature of GPUs may not be able to exploit un-structured forms of sparsity. Therefore, in order to be effective pruning has to be achieved simultaneously on all the weights of a given element, like a channel or a filter, so that the element can be deleted entirely. The choice of the elements to be pruned therefore depends on the target ANN architecture, an issue that has not been very clearly discussed in the literature so far. This motivates a specific feature of our development whereby we allow to arbitrarily partition the weight vector and measure the sparsity in terms of the number of partitions that are eliminated, as opposed to just the number of weights.

In this work, we develop a novel method to perform structured pruning during training through the introduction of a Structured Perspective Regularization (SPR) term. More specifically, we start from a natural exact Mixed-Integer Programming (MIP) model of the sparsity-aware training problem where we consider, in addition to the loss and ℓ_2 regularization, also the ℓ_0 norm of the structured set of weights. A novel application of the Perspective Reformulation technique leads to a tighter continuous relaxation of the original MIP model and ultimately to the definition of the SPR term. Our approach is therefore principled, being grounded on an exact model rather than based on heuristic score functions to decide what entities to prune as prevalent in the literature so far. It is also flexible as it can be adapted to any kind of structured pruning, provided that the prunable entities are known before the training starts, and the final expected amount of pruning is controlled by the hyper-parameter providing the weight of the ℓ_0 term in the original MIP model. While our approach currently only solves a relaxation of the integer problem, it would clearly be possible to exploit established Operations Research techniques to improve on the quality of the solution, and therefore of the pruning. Yet, the experimental results show that our approach is already competitive with, and often significantly better than, the state of the art. Furthermore, since we perform pruning during training by just changing the regularization term our approach can use standard training techniques and its cost is not significantly higher than the usual training without sparsification.

A.2 Related works

The field of pruning is experiencing a growing interest in the Machine Learning (ML) community, starting from the seminal work [64] that obtained unexpectedly good results from a trivial magnitude-based approach. The same magnitude-based approach was extended in [51] with a re-training phase where the non-pruned weight are re-initialised to their starting values.

From the structured pruning side, a possibility is adding to the network parameters a scaling factor for each prunable entity, multiplying all the corresponding parameters; then, sparsity is enforced by adding the ℓ_1 norm of the scaling factors vector. In [129] a pruning mask is defined, i.e., a differentiable approximation of a thresholding function that pushes the scaling factors to 0 when they are lower than a fixed threshold, avoiding numerical issues.

MIP techniques have been successfully used in the ANN context, but mostly in applications unrelated to pruning like construction of adversarial examples (with fixed weights) [46]. In [5], the approach is extended to a larger class of activation functions and stronger formulations are defined. An exception is [41], where a score function is defined to asses the importance of a neuron and then a MIP is used to minimize the number of neurons that need to be kept at each layer to avoid large accuracy drops.

The link between Perspective Reformulation techniques and sparsification has been recently recognized [6, 36], but typically in the context of regression problems that are much simpler than ANNs. In particular, all the above papers count (the equivalent of) each weight individually, and therefore they do not consider structured pruning of sets of related weights as it is required for ANNs. Furthermore, the sparsification approach is applied to input variables selection in settings that typically have orders of magnitude fewer elements to be sparsified than the present one.

A.3 Mathematical model

We are given a dataset X, an ANN model architecture where the set of parameters $W = \{ w_j \mid j \in I \}$ is partitioned in N subsets, that we call prunable entities, such that $W = \cup_{i=1}^{N} E_i$ and a loss function $L(\cdot)$. If the value of a parameter w_j is zero it could be eliminated from the model (pruned) but, for the reasons discussed above, we are only interested in pruning the entities E_i, which is possible only if $w_j = 0$ for all $j \in E_i$. We should consider a three-objective optimization problem where we: i) minimize the loss, ii) minimize some standard regularization term aiming at improving the model's generalization capabilities, and iii) maximize the number of pruned entities E_i. As customary in this setting, we approach

this by a scaling of the three objective functions by means of hyperparameters whose optimal values are found by standard grid search techniques.

Starting from the usual ML framework with ℓ_2 regularization, we consider the following MIP

$$\min L(X, W) + \lambda[\alpha \sum_{j \in I} w_j^2 + (1 - \alpha) \sum_{i=1}^{N} y_i] \tag{A.1}$$

$$-My_i \leq w_j \leq My_i \qquad \forall w_j \in E_i \quad i = 1, .., N \tag{A.2}$$

$$y_i \in \{0, 1\} \qquad i = 1, .., N \tag{A.3}$$

where $\alpha \in [0, 1]$ and $\lambda > 0$ are scalar hyper-parameters while M is an upper bound on the absolute value of the parameters. The binary variable y_i is 0 if the corresponding prunable entity is pruned, 1 if it is not. The standard "big-M" constraints (A.2) ensure that if $y_i = 0$ then $0 \leq w_j \leq 0$ for all parameters in the entity E_i, while if $y_i = 1$ the parameters can take any possible useful value (since M is an upper bound). In other words, the term "$\sum_{i=1}^{N} y_i$" represents the ℓ_0 norm of the structured set of weights, i.e., the cardinailty of the set $\{\, i \in \{1, \ldots, N\,\} : \exists j \in E_i \text{ s.t. } w_j \neq 0\}$.

In general, solving (A.1)–(A.3) directly is not computationally efficient, and therefore a standard strategy is to consider its *continuous relaxation* whereby (A.3) is relaxed to $y_i \in [0, 1]$. If $L(\cdot)$ is convex, as in our case, this is easily solvable but its solution in both the y and w variables, which is what one could use to perform the pruning, can be rather different from the optimal solution of (A.1)–(A.3), therefore leading to inefficient prunings. To improve on that, we consider the Perspective Reformulation [50]

$$\min \Big\{ L(X, W) + \lambda[\alpha \sum_{i=1}^{N} \sum_{j \in E_i} w_j^2 / y_i + (1 - \alpha) \sum_{i=1}^{N} y_i] \;:\; \text{(A.2)} \;,\; \text{(A.3)} \Big\} \tag{A.4}$$

of problem (A.1)–(A.3). It can be shown that (A.4) has the same integer optimal solution as the original problem, but its continuous relaxation (the Perspective Relaxation) is "better" in a well-defined mathematical sense: its optimal objective value is (much) closer to the true optimal value of (A.1)–(A.3), which typically implies that its optimal solution is more similar to the true optimal solution.

While one can expect that the solution of the relaxation of (A.4) can provide a better guide to the pruning procedure, its form makes it more difficult to apply standard training techniques to solve it. Following the lead of [47, 48], we then proceed at simplifying the model by projecting away the y variables, the details about this step are reported in the appendix. All in all, we can rewrite the continuous relaxation of (A.4) as

$$\min \Big\{ L(X, W) + \lambda \sum_{i=1}^{N} z_i(w_i; \alpha, M) \Big\}, \tag{A.5}$$

where $w^i = [\, w_j \,]_{j \in E_i}$ and

$$z_i(w_i; \alpha, M) = \begin{cases} \sqrt{\frac{1-\alpha}{\alpha}}(1+\alpha)||w^i||_2 & \text{if } \frac{||w^i||_\infty}{M} \leq \sqrt{\frac{\alpha}{1-\alpha}}||w^i||_2 \leq 1 \\ \frac{M}{||w^i||_\infty}||w^i||_2^2 + (1-\alpha)\frac{||w^i||_\infty}{M} & \text{if } \sqrt{\frac{\alpha}{1-\alpha}}||w^i||_2 \leq \frac{||w^i||_\infty}{M} \leq 1 \\ ||w^i||_2^2 + (1-\alpha) & \text{otherwise.} \end{cases}$$

We call $z_i(w_i; \alpha, M)$ the *Structured Perspective Regularization* (SPR) w.r.t. the structure specified by the sets E_i. It is easily seen that the SPR behaves like the ordinary ℓ_2 regularization in parts of the space but it is significantly different in others. Due to being derived from (A.4), we can expect (with a proper choice of the hyperparameters) the SPR to promote sparsity—in terms of the sets E_j—better than the ℓ_2 norm. Indeed, SPR for $i \in I$ depends on the ℓ_∞ norm of w_i, which means that it is zero only if all the components of w_i are zero. In other words, while using the, say, ordinary ℓ_1 would promote sparsity on each weight individually, the SPR can be expected to better promote structured sparsity as required by our application. However, the solution of (A.5) can be sought for with all of the usual algorithms for training ANNs (SGD, Adam, etc.), and therefore should not, in principle, be more costly then non-sparsity-inducing training.

The above mentioned solution scheme still needs some minor fixes and improvements to fully adapt to our scenario, details about these changes can be found in the appendix,

A.4 Experiments

We tested our method on the task of filter pruning in Deep Convolutional Neural Networks; that is, the prunable entities are the filters of the convolutional layers. More specifically, the weights in a convolutional layer with n_{inp} input channels, n_{out} output channels and $k \times k$ kernels is a tensor with four dimensions (n_{inp}, n_{out}, k, k): our prunable entities correspond to the sub-tensors with the second coordinate fixed, and therefore have $n_{inp} \times k \times k$ parameters. The code used to run the experiments was written starting from the public repository CifarCode and ImagenetCode.

For our experiments, we used 3 very popular datasets: CIFAR-10 [87], CIFAR-100 [87] and ImageNet [88]. As architectures, we focused on resnet [68] and, in particular, we used resnet-20 and ResNet-56 for both CIFAR datasets and resnet-18 for the ImageNet dataset.

For all the experiments, we used Pytorch (1.7.1 and 1.8.1) with Cuda, the CrossEntropyLoss and the SGD optimizer with 0.9 momentum.

After the first phase, we considered as pruned all weights with absolute value lower than 1e-4

and we decided to prune all entities with more than 95% of parameters pruned.

To get the values of M_i's, we used the maximum absolute values of the weights for each layer of a network with the same architecture but trained without our regularization term (for resnet-20 and resnet-56 we trained it, for resnet-18 we used the pretrained version available from torchvision).

Further details on the hardware that were used, general settings and detailed results tables can be found in the appendix for each of the datasets.

As shown in Table 5.1, training ResNet-20 on CIFAR-10, we were able to prune more than 23% of the parameters by still increasing the accuracy of the original model, while we can prune almost 70% of the model by still preserving more than 90% accuracy.

Using ResNet-20 on the CIFAR-100 dataset, we can again prune more than 15% of parameters while improving the accuracy of the original model (Table 5.2). Due to the fact that CIFAR-100 is more challenging than CIFAR-10, it was not possible to prune very many parameters without losing a significant amount of accuracy: we can still achieve more than 68% accuracy by pruning a few more than 30% of the parameters, but accuracy drop to less than 67% if we try to prune more.

Finally, Table 5.3 reports results of training the ResNet-56 architecture on CIFAR-10: once again pruning about 30% of the parameters improves accuracy and we can keep more than 93% accuracy while pruning more than a half of the network.

Results on ImageNet using ResNet-18, Table 5.7, show that even in a very large and difficult dataset our method is able to improve the original model results by a consistent margin, reaching almost 71% accuracy while pruning more than 6% of the parameters. Pruning more than 50% of the network causes a drop of 2.5% in the accuracy, while a way more consistent decrease happens when we prune about 80% of the parameters.

A.4.1 Comparison with state-of-the-art methods

In this section, we compare our results (denoted as SPR) with some of the state of the art algorithms for structured pruning. We report results from [75] (denoted by SSS), [137] (denoted by EPFS), [147] (denoted by L2PF), [98] (denoted by PFFEC), [103] (denoted as HRANK) and [42] (denoted by HFP).

Since not all the above papers reported the results for all our metrics (for example, some works only reported the percentage of parameters pruned), in some cases we had to do some conversions that naturally came with some mild approximation. Moreover, in [75], only plots were presented, so we had to approximately deduce the data from some points of the figures

(figure 2(a) and figure 2(c) of [75], we denote the points as P1, P2, etc.). For ImageNet the top5 accuracy is not reported in [42], so we marked the corresponding field in our table with a "N/A". Finally, we report results for different settings of each method since they were present in the original papers; however, it should be remarked that not all of them are filter pruning methods, some rather being general structured pruning ones.

Regarding ResNet-20 on CIFAR-10, Table A.1, our results outperform the other methods in most of the cases, meaning that we can reach equal or better accuracy while pruning a larger amount of parameters. Whenever we do not outperform the other methods we have comparable performances and we can have a better accuracy or sparsity maintaining the other metric close. This is for instance the case of L2PF, that achieves 89.9% accuracy with 73.96% sparsity, while we achieve a little higher sparsity (74.00%) loosing a little accuracy (89.47%).

Table A.1 Results of state of the art method on CIFAR-10 using ResNet-20.

Method	Setting	Accuracy	Pruned parameters (%)	
SSS	P1	90.80	120000	(44.44)
	P2	91.60	40000	(14.81)
	P3	92.00	10000	(3.70)
	P4	92.50	0	(0.00)
EPFS	B-0.6	91.91	70000	(24.60)
	B-0.8	91.50	100000	(36.90)
	F-0,05	90.83	130000	(51.10)
	C-0.6-0.05	90.98	150000	(56.00)
L2PF	LW	89.90	199687	(73.96)
SPR	λ1.1-α0.3	89.47	199809	(74.00)
	λ0.5-α0.3	91.23	140535	(52.05)
	λ0.2-α0.3	92.69	64188	(23.77)

Table A.2 Results of state of the art method on CIFAR-100 using ResNet-20.

Method	Setting	Accuracy	Pruned parameters (%)	
SSS	P1	65.50	120000	(44.44)
	P2	67.10	40000	(14.81)
	P3	68.10	10000	(3.70)
	P4	69.20	0	(0.00)
SPR	λ1.0-α0.1	66.28	123984	(45.92)
	λ1.3-α0.3	68.36	84960	(31.47)
	λ0.7-α0.3	69.20	42480	(15.73)

On CIFAR-100 using ResNet-20, Table A.2, we clearly outperform SSS, as we can achieve more than 68.3% accuracy wile pruning more than 30% of parameters while SSS could prune only 14.81% to obtain a little bit more than 67% accuracy. In Table A.3 we can observe a similar situation to ResNet-20 on CIFAR-10 for ResNet-56 on the same dataset. The only result we did not outperform was the HPF 93.30 accuracy with 50% sparsit,y but we can obtain a little bit more sparsity (54.21%) with almost the same accuracy (93.13%).

On ImageNet using ResNet-18, Table A.4, we can see that even if our method does not outperform the other ones, we are able to prune consistently more parameters without a very large accuracy drop. Likely some more parameter tuning could lead us to even more competitive results.

Table A.3 Results of state of the art method on CIFAR-10 using ResNet-56

Method	Setting	Accuracy	Pruned parameters (%)	
PFFEC	A	93.10	80000	(9.40)
	B	93.06	120000	(13.70)
EPFS	B-0.6	92.89	240000	(27.70)
	B-0.8	92.34	500000	(58.60)
	F-0.01	92.96	170000	(20.00)
	F-0.05	92.09	510000	(60.10)
	C-0.6-0.05	92.53	570000	(67.10)
HFP	0.5	93.30	425000	(50.00)
	0.7	92.31	608430	(71.58)
HRank	P1	90.72	580000	(68.10)
	P2	93.17	360000	(42.40)
	P3	93.52	140000	(16.80)
SPR	λ0.5-α0.001	92.55	633960	(74.58)
	λ1.1-α1e-4	93.13	460800	(54.21)
	λ0.8-α0.1	93.98	268128	(31.54)

Table A.4 Results of state-of-the-art method on ImageNet using ResNet-18

Method	Setting	Accuracy (top1)	Accuracy (top5)	Pruned parameters (%)	
EPFS	F-0.05	67.81	88.37	3690000	(34.60)
HFP	0.20	69.15	N/A	2354869	(22.07)
	0.35	68.53	N/A	3976709	(37.27)
SPR	λ1.3-α0.3	67.26	87.71	5796625	(54.33)

A.5 Conclusions and future directions

Based on an exact MIP model for the problem of pruning ANNs, we proposed a new regularization term, based on the projected Perspective Reformulation, designed to promote structured sparsity. The proposed method is able to prune any kind of structures and the amount of pruning is tuned by appropriate hyper-parameters. We tested our method on some classical datasets and architectures and we compared the results with some of the state-of-the-art structured pruning methods. The results have shown that our method is competitive.

These results are even more promising in view of the fact that further improvements should be possible. Indeed, we are currently solving the continuous relaxation of our proposed exact starting model, albeit a "tight" one due to the use of the Perspective Reformulation technique. By a tighter integration with other well-established MIP techniques, further improvements are foreseeable.

A.6 Appendix

A.6.1 Eliminating the y variables

To remove these variables we compute a closed formula $\tilde{y}(w)$ for the optimal value of the y variables in the continuous relaxation of (A.4) assuming that w are the optimal weights. When w is fixed, the problem decomposes over the subsets, and therefore we only need to consider each fragment

$$f_i(W, y_i) = \lambda[\alpha \textstyle\sum_{j \in E_i} w_j^2 / y_i + (1 - \alpha) y_i]$$

separately. Since f_i is convex in y if $y > 0$, we just need to find the root of the derivative

$$\frac{\partial f_i(W, y_i)}{\partial y_i} = \lambda \left[-\alpha \sum_{w_j \in E_i} \frac{w_j^2}{y_i^2} + (1 - \alpha) \right] = 0 \quad \Longleftrightarrow \quad y_i = \sqrt{\frac{\alpha \sum_{w_j \in E_i} w_j^2}{1 - \alpha}}$$

(we are only interested in positive y) and then project it on the domain. Note that, technically, $f_i(W, y_i)$ is nondifferentiable for $y_i = 0$ but that value is only achieved when $w_j = 0$ for all $j \in E_i$, in which case the choice is obviously optimal. The constraints that defines the domain of y_i can be rewritten as $y_i \geq |w_j|/M$ for all $j \in E_i$, together with $y_i \in [0, 1]$; putting everything together, we obtain

$$\tilde{y}_i(w) = \min \Big\{ \max \big\{ |w_j|/M \, : \, j \in E_i \big\} \, , \, \sqrt{\alpha \textstyle\sum_{j \in E_i} w_j^2 / (1 - \alpha)} \big\} \, , \, 1 \Big\},$$

where we note that we do not need to enforce positivity since all the quantities are positive.

A.6.2 Final improvements of the algorithm

Remarkably, the SPR depends on the choice of M, which is, in principle, nontrivial. Indeed, all previous attempts of using PR techniques for promoting sparsity [6, 36] have been using the "abstract" nonlinear form $(1 - y_i)w_i = 0$ of (A.2) (assuming $E_i = \{ i \}$ as in those treatments). This still yields the same Perspective Reformulation but it is not conducive to projecting away the y variables as required by our approach. While M could in principle be treated as another hyperparameter, in a (deep) ANN, different layers can have rather different optimal upper bounds on the weights; hence, using a single constant M for all the prunable entities is sub-optimal. The ideal choice would be to compute one constant M_i for each entity E_i; however, entities in the same layer are often similar to each other, so we decided to compute a different constant for each layer of the network and then use it for all

entities belonging to that layer (see below).

Furthermore, all the development so far has assumed that all prunable entities E_i are equally important. However, this may not be true, since different entities can have different number of parameters and therefore impact differently on the overall memory and computational cost. To take this feature into account, we modify our regularization

by scaling the term corresponding to entity E_i by $u_i / \sum_{j=1}^{N} u_j$, u_i being the number of parameters belonging to entity E_i.

Finally, we perform a fine-tuning phase. After the ANN has been trained with the SPR, we perform the pruning (eliminating all the entities where $||w^i||_\infty$ is smaller than a given tolerance (see below) and then we re-train the pruned network with the standard ℓ_2 regularization, starting from the value of the weights (for the non-pruned entities) obtained at the end of the previous phase rather than re-initializing them.

A.6.3 Time complexity study

Our method is composed of two steps, in the first one the computation of an additional regularization term is required, while the second one is a fine-tuning phase that is common to many other pruning methods and is just a normal backpropagation. So, we can do a comparison between the cost of an epoch in the first step and in the second one to have a "worst case scenario" of a comparison with other methods that use a regularization term to prune. From Table A.5, we can notice that for easier data sets (small input size) the cost of one epoch with our regularization term is roughly the double of a normal one, while for the hardest data set, so in the contest of real applications, this gap decreases and the two costs are almost the same.

Table A.5 Average computation times (seconds) for one epoch with and without the SPR term

Architecture and data set	time SPR	time without SPR
ResNet-20 on CIFAR-10	13.05	6.51
ResNet-56 on CIFAR-10	36.58	16.99
ResNet-20 on CIFAR-100	22.99	11.26
ResNet-18 on Imagenet	2,433.14	2,401.05

Table A.6 Ablation study on hyperparameter (ResNet-56 on CIFAR-10)

λ	α	Accuracy	Pruned parameters
1.7	1e-4	90.12	89.04
1.4	1e-4	88.47	87.09
1.7	0.3	90.59	85.76
0.5	0.3	94.04	6.03

A.6.4 Detail on grid search and ablation study

As we stated in the first paragraph of Section 3, α and λ hyperparameters are found through grid search. We performed a different grid search for each of the data set / architecture pair we used: for ResNet20 on CIFAR10, we had $\lambda \in [0.2, 0.5, 0.8, 1.1, 1.4, 1.7]$ and $\alpha \in [1e{-}4, 1e{-}3, 5e-3, 1e-2, 0.1, 0.3]$; for Resnet56 on CIFAR-10, we had $\lambda \in [0.5, 0.8, 1.1, 1.4, 1.7]$; for Resnet20 on CIFAR-100, we had $\lambda \in [0.1, 0.4, 0.7, 1.0, 1.3, 1.6]$ and $\alpha \in [1e{-}3, 1e{-}2, 0.1, 0.3]$; finally for Resnet18 on Imagenet, we had $\lambda \in [0.5, 0.8, 1.0, 1.3]$ and $\alpha \in [1e{-}3, 1e{-}2, 0.1, 0.3]$. The λ parameter is just the usual regularization parameter, the α parameter instead tunes how much of the regularization is computed in a "structured" way. From Table A.6, we can see that to obtain the maximum level of sparsity λ needs to be high, while α needs to be close to 0, while for lower level of sparsity can be sufficient and sometimes more effective to just increase α. Finally, for very low level of pruning, also λ needs to be decreased.

Table A.7 Accuracy before and after the finetuing phase (ResNet18 on CIFAR.10)

λ	α	Accuracy before	Accuracy after
1.1	0.01	82.40	85.56
1.7	0.3	85.28	87.33
1.1	0.3	88.22	89.47
0.5	0.3	90.62	91.23
0.2	0.3	92.46	92.69
Original model		92.03	-

Finally, we report an observation on the importance of the finetuning phase. From Table A.7, we can see that this step is crucial to obtain higher accuracy when the pruning determined a big drop in this value, while is less important when the accuracy stays high despite the pruning.

A.6.5 Observation on the structure of the pruned network

From the experiments, we noticed that our algorithm heavily prunes the last layers of the network. This is due to the fact that the gain in the objective function is bigger for these last layers since their filters contain way more parameters than filters belonging to the first layers. When we allow our pruning algorithm to heavily prune the network at the cost of a consistent accuracy drop or when the model is so over-parametrized that even pruning a lot of parameters slightly affect the accuracy, we notice that all final layers are fully pruned. Differently, when we prune less parameters to avoid big accuracy drops the final layers that are not fully pruned tend to be the same for different configurations of the hyperparameters, i.e., likely identifying essential structures of the model. For example, pruning ResNet18 on the Imagenet data set, the layer with the last residual connection is not pruned in almost all our experiments.

A.6.6 CIFAR10 and CIFAR100

These experiments were performed on a single GPU, either a TESLA V100 32GB or NVIDIA Ampere A100 40GB. The model was trained for 300 epochs and then fine tuned for 200 ones. The dataset was normalized, then we performed data augmentation through random crop and horizontal flip. Mini batches of size 128 (64 for CIFAR-100) were used for training. The learning rate was initialized to 1e-1 or 1e-2 and then it is divided by 10 at epochs 120, 200, 230, 250, 350, 400 and 450.

These experiments were performed on single TITAN V 8GB GPU. The model was trained for 150 epochs and fine tuned for 50 ones. The preprocessing was the same as for the CIFAR datasets. We used mini batches of 256 and 0.1 learning rate that was divided by 10 every 35 epochs. On the ImageNet tests, as usual for datasets with so many classes, we decided to report also the top5 accuracy, that is the percentage of samples where the correct label was on the 5 higher scored classes by the model.

Table A.8 Results of our algorithm on CIFAR-10 using ResNet-20

Learning rate	λ	α	Accuracy	Pruned parameters (%)	
0.1	1.1	0.01	85.56	231597	(85.78)
0.1	1.1	1e-4	86.35	227394	(84.22)
0.1	1.7	0.3	87.33	217233	(80.46)
0.1	0.8	0.1	88.14	206694	(76.55)
0.1	1.1	0.3	89.47	199809	(74.00)
0.1	0.5	0.01	90.06	187767	(69.54)
0.1	0.5	0.3	91.23	140535	(52.05)
0.1	0.2	0.3	92.69	64188	(23.77)
Original model			92.03	0	(0.00)

Table A.9 Results of our algorithm on CIFAR-100 using ResNet-20.

Learning rate	λ	α	Accuracy	Pruned parameters (%)	
0.01	1.3	1e-3	65.28	149184	(55.25)
0.01	1.0	0.1	66.28	123984	(45.92)
0.01	1.6	0.3	66.64	104256	(38.61)
0.01	1.3	0.3	68.36	84960	(31.47)
0.01	0.7	0.3	69.20	42480	(15.73)
Original model			68.55	0	(0.00)

Table A.10 Results of our algorithm on CIFAR-10 using ResNet-56

Learning rate	λ	α	Accuracy	Pruned parameters (%)	
0.1	1.7	1e-4	90.12	756882	(89.04)
0.1	1.1	0.01	91.35	720135	(84.72)
0.1	0.5	0.01	92.55	633960	(74.58)
0.01	1.1	1e-4	93.13	460800	(54.21)
0.01	0.8	0.1	93.98	268128	(31.54)
Original model			93.35	0	(0.00)

Table A.11 Results of our algorithm on ImageNet using ResNet-18

Learning rate	λ	α	Accuracy (top1)	Accuracy (top5)	Pruned parameters (%)	
0.1	0.5	0.3	70.89	89.79	652644	(6.11)
0.1	1.3	0.3	67.26	87.71	5796625	(54.33)
0.1	1.3	0.01	62.27	84.43	8607247	(80.67)
Original model			69.76	89.08	0	(0.00)

APPENDIX B PROOFS OF CHAPTER 6 RESULTS

B.1 Auxiliary results

Let $\mathbf{w} \in \mathbb{R}^d$ and $r \in \mathbb{R}^+$. Define $\mathbb{B}(\mathbf{w}, r) = \left\{\mathbf{w}' \in \mathbb{R}^d | \ \|\mathbf{w}' - \mathbf{w}\| \leq r\right\}$. Given that f is quasi-convex and satisfies a Holder condition, and $\mathbf{\Omega}$ is a compact set, it follows that $\mathbf{\Omega}^* \neq \emptyset$ and $f^* > -\infty$. We start with two technical lemmas needed for the proof of Theorem 6.4.1.

Lemma B.1.1. *Let $\mathbf{g} \in \mathbb{R}^d$ be a d-dimensional vector and $R, C, B, \eta \in \mathbb{R}^+$ be real constants such that $\|\mathbf{g}\| = 1$, $R < 1$ and $C > B$. Define $F(\mathbf{w}, \mathbf{r}) = \eta\|\mathbf{g} + \mathbf{r}\|^2 - 2\langle \mathbf{w}, \mathbf{g} + \mathbf{r}\rangle$ where $\mathbf{r}, \mathbf{w} \in \mathbb{R}^d$. Consider the following optimization problem,*

$$F^* = \max_{\mathbf{w},\mathbf{r} \in \mathbb{R}^d} F(\mathbf{w}, \mathbf{r}), \tag{B.1}$$

subject to:

$$\begin{aligned} &\langle \mathbf{g}, \mathbf{w}\rangle \geq B, \\ &\|\mathbf{r}\|^2 \leq R^2, \\ &\|\mathbf{w}\|^2 \leq C^2. \end{aligned} \tag{B.2}$$

Then

$$F^* \leq \max\left\{\eta\left(1 + R^2\right) + 2R\sqrt{\eta^2 + C^2 - 2\eta B} - 2B, \eta\left(R^2 - 1\right)\right\}.$$

Corollary B.1.0.1. *The upper bound in Lemma B.1.1 is attained if $d \geq 2$.*

Theorem 6.4.1 is proved by contradiction using a standard argument for this type of results.

For the sake of simplicity, we prove the theorem assuming that $S = 0$. The proof for the general case, i.e. $S \neq 0$, is similar upon noticing that $\mathbf{w}_{k+1} = \mathsf{P}_\Omega\left[\mathbf{w}_k - \eta\left(\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k + \frac{\mathbf{s}_k}{\eta}\right)\right]$ and replacing R with $R + \frac{S}{\eta}$. Given that η is assumed to be fixed, this change does not cause any difficulty.

B.2 Proof of Theorem 6.4.1

Proof. Given that c is the diameter of $\mathbf{\Omega}$, the desired result clearly holds if $\Gamma(c) \geq c$. Thus, assume $\Gamma(c) < c$. Suppose that $\exists\ \delta > 0,\ \bar{k} > 0\ s.t.\ f(\mathbf{w}_k) > f^* + L(\Gamma(c) + \delta)^p \quad \forall\, k > \bar{k}$ and $\mathbf{w}^* \in \mathbf{\Omega}^*$. Given that $f(\cdot)$ satisfies the Hölder condition, we have for all $k > \bar{k}$

$$f(\mathbf{w}) - f^* \leq L\,\mathrm{dist}(\mathbf{w}, \mathbf{\Omega}^*)^p \leq L(\Gamma(c) + \delta)^p < f(\mathbf{w}_k) - f^*, \qquad \forall\, \mathbf{w} \in \mathbb{B}(\mathbf{w}^*, \Gamma(c) + \delta) \quad ,$$

which implies that $\mathbb{B}(\mathbf{w}^*, \Gamma(c)+\delta) \subset \mathcal{S}_{f,f(\mathbf{w}_k)},\ \forall\, k > \bar{k}$. Using Lemma 6 of [86]

$$\langle \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}, \mathbf{w}_k - \mathbf{w}^* \rangle \geq \Gamma(c) + \delta, \quad \forall\, k > \bar{k},$$

hence one can use Lemma B.1.1 with $B = \Gamma(c)+\delta$, $C = c$, $\mathbf{g} = \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}$, $\mathbf{w} = \mathbf{w}^* - \mathbf{w}_k$, $\mathbf{r} = \mathbf{r}_k$, $R = R$ and $\eta = \eta$. This yields

$$\begin{aligned} &\langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{g}_k \rangle + \langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{r}_k \rangle - \frac{\eta}{2} \| \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k \|^2 \\ &\geq \min \left\{ \Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]}, \frac{\eta}{2}\left(1-R^2\right) \right\}. \end{aligned} \tag{B.3}$$

It is shown that for δ is small enough

$$\Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} < \frac{\eta}{2}\left(1-R^2\right).$$

Given that $\Gamma(c)$ is the maximum of two terms, we show the inequality holds for both terms, and hence for their maximum. If $\Gamma(c) = \frac{\eta}{2}(1+R^2)$, then

$$\begin{aligned} \Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} &= \delta - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} \leq 0 \\ &\leq \frac{\eta}{2}\left(1-R^2\right), \end{aligned} \tag{B.4}$$

where the first inequality follows from the fact that we can choose δ arbitrarily small and

second one from $R < 1$. Otherwise, $\Gamma(c) = \frac{\eta(1-R^2)}{2} + Rc$, thus

$$\begin{aligned}
\left[\Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right)\right]^2 &= \frac{\eta^2}{4}\left(1+R^2\right)^2 + \left[\Gamma(c)+\delta\right]^2 - \eta\left[\Gamma(c)+\delta\right]\left(1+R^2\right) \\
&= \frac{\eta^2}{4}\left(1+R^2\right)^2 + \left[\frac{\eta\left(1-R^2\right)}{2} + Rc + \delta\right]^2 \\
&\quad - \eta\left[\frac{\eta\left(1-R^2\right)}{2} + Rc + \delta\right]\left(1+R^2\right) \\
&= \frac{\eta^2}{4}\left(1+R^2\right)^2 + \frac{\eta^2\left(1-R^2\right)^2}{4} + R^2c^2 + \eta Rc\left(1-R^2\right) - \frac{\eta^2\left(1-R^4\right)}{2} \\
&\quad - \eta Rc\left(1+R^2\right) + \delta^2 + 2\delta Rc + \delta\eta\left(1-R^2\right) - \delta\eta\left(1+R^2\right) \\
&= \frac{\eta^2}{2}\left(1+R^4\right) + R^2c^2 - 2\eta cR^3 - \frac{\eta^2\left(1-R^4\right)}{2} + \delta^2 + 2\delta Rc - 2\delta\eta R^2 \\
&= R^4\eta^2 + R^2c^2 - 2\eta cR^3 + \delta^2 + 2\delta Rc - 2\delta\eta R^2 \\
&= R^2\left(c^2 + \eta^2R^2 - 2\eta Rc - 2\eta\delta\right) + \delta^2 + 2\delta Rc \\
&= R^2\left[\eta^2 + c^2 - \eta^2\left(1-R^2\right) - 2\eta Rc - 2\eta\delta\right] + \delta^2 + 2\delta Rc \\
&= R^2\left\{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]\right\} + \delta^2 + 2\delta Rc.
\end{aligned}$$

Taking the square root of the first and last term

$$\Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) = \sqrt{R^2\left\{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]\right\} + \delta^2 + 2\delta Rc}. \tag{B.5}$$

Then, the first term in the min operator of (B.3) is

$$\begin{aligned}
&\Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} \qquad \text{(B.6)} \\
&= \sqrt{R^2\left\{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]\right\} + \delta^2 + 2\delta Rc} - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} \xrightarrow{\delta\to 0} 0.
\end{aligned}$$

Hence, if δ is small enough

$$\Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} \leq \frac{\eta}{2}\left(1-R^2\right). \tag{B.7}$$

If δ is small enough, Eq. (B.3), (B.4) and (B.7) yield

$$\begin{aligned}
&\langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{g}_k\rangle + \langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{r}_k\rangle - \frac{\eta}{2}\left\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k\right\|^2 \\
&\geq \Gamma(c) + \delta - \frac{\eta}{2}\left(1+R^2\right) - R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]}.
\end{aligned} \tag{B.8}$$

It follows from the argument leading to Eq. (B.5) that if $\Gamma(c) \geq \frac{\eta(1-R^2)}{2} + Rc$,

$$\Gamma(c) + \delta - \frac{\eta}{2}(1+R^2) \geq \sqrt{R^2\left\{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]\right\} + \delta^2 + 2\delta Rc}. \tag{B.9}$$

Having noted that $\Gamma(c) > \frac{\eta}{2}$, we obtain $\eta^2 + c^2 - 2\eta(\Gamma(c)+\delta) < c^2$. This yields

$$\begin{aligned} R^2\left\{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]\right\} + \delta^2 + 2\delta Rc &= \left\{R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} + \delta\right\}^2 \\ &\quad - 2\delta R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} + 2R\delta c \\ &\geq \left\{R\sqrt{\eta^2 + c^2 - 2\eta\left[\Gamma(c)+\delta\right]} + \delta\right\}^2. \end{aligned} \tag{B.10}$$

It follows from Eq. (B.8), (B.9) and (B.10) that

$$\langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{g}_k\rangle + \langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{r}_k\rangle - \frac{\eta}{2}\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k\|^2 \geq \delta,$$

leading to

$$\begin{aligned} \|\mathbf{w}_{k+1} - \mathbf{w}^*\| &\leq \|\mathbf{w}_k - \mathbf{w}^*\|^2 - 2\eta\left(\langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{g}_k\rangle + \langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{r}_k\rangle - \frac{\eta}{2}\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k\|^2\right) \\ &\leq \|\mathbf{w}_k - \mathbf{w}^*\|^2 - 2\eta\delta, \end{aligned}$$

where we used the definition of $\mathbf{w}_{k+1}$ and the property of the projection map $\mathsf{P}_\Omega(\cdot)$ in the first inequality.

Thus

$$\|\mathbf{w}_{k+1} - \mathbf{w}^*\|^2 \leq \|\mathbf{w}_k - \mathbf{w}^*\| - 2\eta\delta \leq \ldots \leq \|\mathbf{w}_0 - \mathbf{w}^*\| - 2\eta(k - \bar{k} + 1)\delta, \quad \forall\, k > \bar{k}. \tag{B.11}$$

The desired result then follows upon noticing that the upper bound tends to $-\infty$ as k tends to $+\infty$ which leads to a contradiction.

■

B.3 Intuition behind Lemma B.1.1

By scrutinizing the proof of our Theorem 6.4.1 and Theorem 3.1 of [73], we notice that the key to the proof is

$$\begin{aligned}\|\mathbf{w}_{k+1}-\mathbf{w}^*\|^2 \leq \|\mathbf{w}_k - t\hat{\mathbf{g}}_k - \mathbf{w}^*\|^2 &= \|\mathbf{w}_k-\mathbf{w}^*\|^2 - 2\eta\langle \mathbf{w}_k-\mathbf{w}^*, \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}\rangle \\ &\quad - 2\eta\langle \mathbf{w}_k-\mathbf{w}^*, \mathbf{r}_k\rangle + \eta^2\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}+\mathbf{r}_k\|^2 \\ &= \|\mathbf{w}_k-\mathbf{w}^*\|^2 - 2\eta\left(B + \langle \mathbf{w}_k-\mathbf{w}^*, \mathbf{r}_k\rangle - \frac{\eta}{2}\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}+\mathbf{r}_k\|^2\right),\end{aligned}$$

where B should be such that $\exists \delta' > 0 \mid \left(B + \langle \mathbf{w}_k-\mathbf{w}^*, \mathbf{r}_k\rangle - \frac{\eta}{2}\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}+\mathbf{r}_k\|^2\right) > \delta'$. The final bound will significantly depend on B. More precisely, greater B values lead to greater, hence worse, bounds.

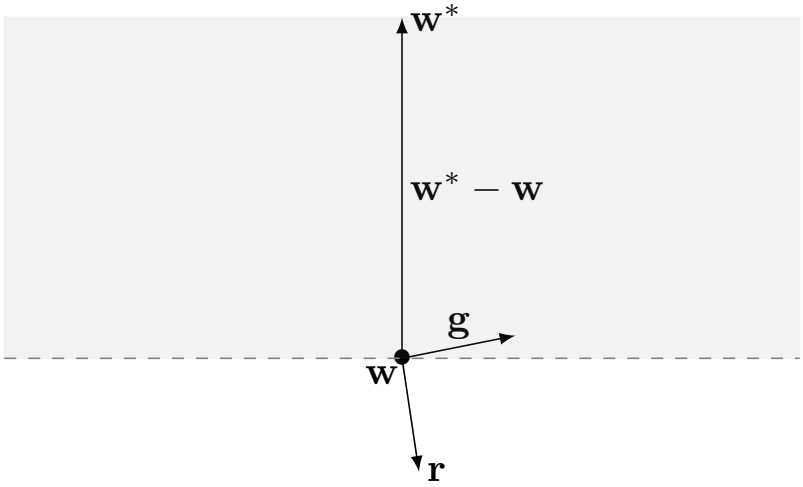


Figure B.1 Given $\mathbf{w}$ and $\mathbf{w}^*$, $\mathbf{g}$ needs to have positive scalar product with $\mathbf{w}^*-\mathbf{w}$ (gray zone). So it s not possible for $\mathbf{r}$ to be opposite to $\mathbf{w}^*-\mathbf{w}$ and colinear with $\mathbf{g}$ at the same time.

In [73] the authors use

$$\begin{aligned}&|\mathbf{w}_k-\mathbf{w}^*\|^2 - 2\eta\left(B + \langle \mathbf{w}_k-\mathbf{w}^*, \mathbf{r}_k\rangle - \frac{\eta}{2}\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}+\mathbf{r}_k\|^2\right) \\ &\leq \|\mathbf{w}_k-\mathbf{w}^*\|^2 - 2\eta\left(B - Rc - \frac{\eta}{2}(1+R)^2\right),\end{aligned} \tag{B.12}$$

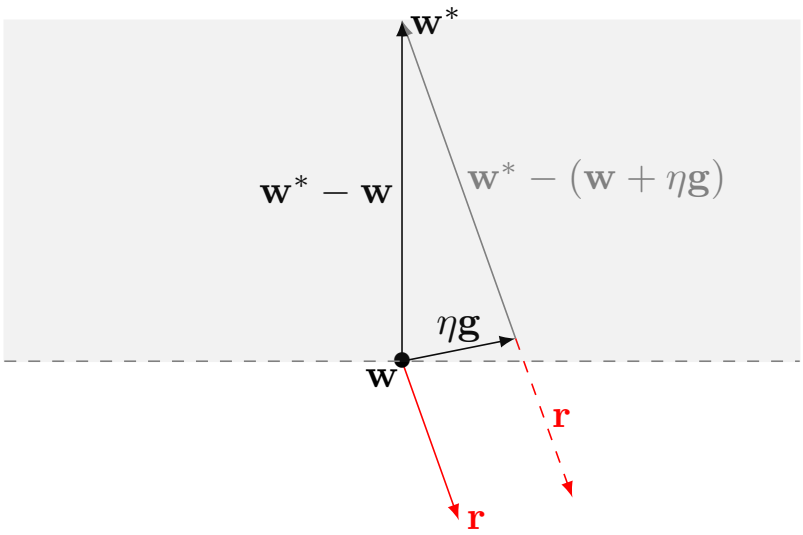


Figure B.2 Given $\mathbf{w}$, $\mathbf{w}^*$ and $\mathbf{g}$ the worst case seems to be $-\mathbf{r} = \mathbf{w}^* - (\mathbf{w} + \eta\mathbf{g})$, scaled to have norm R.

and then choose $B = Rc + \frac{\eta}{2}(1+R)^2 + \delta$. To obtain Eq. (B.12), [73] uses

$$\langle \mathbf{w}_k - \mathbf{w}^*, \mathbf{r}_k \rangle \geq -\|\mathbf{w}_k - \mathbf{w}^*\|\|\mathbf{r}_k\|, \tag{B.13}$$

$$\|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} + \mathbf{r}_k\| \leq \|\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}\| + \|\mathbf{r}_k\|. \tag{B.14}$$

The equality holds in (B.13) and (B.14) if $\mathbf{r}_k = -\alpha(\mathbf{w}_k - \mathbf{w}^*)$ and $\mathbf{r}_k = \beta\mathbf{g}_k$ with $\alpha,\ \beta > 0$, respectively. This then implies that $\langle \mathbf{g}_k, \mathbf{w}_k - \mathbf{w}^* \rangle = \langle \mathbf{r}_k/\beta, -\mathbf{r}_k/\alpha \rangle = -\frac{1}{\alpha\beta}\|\mathbf{r}_k\|^2 < 0$, which contradicts the definition of quasi sub-differential for $\mathbf{g}_k \in \bar{\partial}^* f(\mathbf{w}_k)$. This means that equality cannot hold in Eq. (B.13) and (B.14) simultaneously. That is why, we used Lemma B.1.1 instead of Eq. (B.12) to prove Theorem 6.4.1. Figure B.1 provides the insight for our argument.

Figure B.2 shows that the worst case scenario happens if $\mathbf{r}_k$ is collinear with $\mathbf{w}_k + \eta\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|} - \mathbf{w}^*$.

B.4 Proof of Lemma B.1.1

Loosely speaking, the idea is to use KKT's conditions and to prove that the optimal solutions are on the boundary of the feasible set.

Proof. The gradient of the objective function is

$$\nabla F(\cdot) = (-2(\mathbf{g} + \mathbf{r}), 2\eta(\mathbf{g} + \mathbf{r}) - 2\mathbf{w}).$$

The objective function is convex so the optimum is reached in at least a point on the boundary of the feasible region. We recall that the weak Slater's conditions for Problem B.1 are satisfied if $\exists\, \mathbf{r}, \mathbf{w} \in \mathbb{R}^d$ such that

$$\begin{aligned} \langle \mathbf{g}, \mathbf{w} \rangle &\geq B, \\ \|\mathbf{r}\|^2 &< R^2, \\ \|\mathbf{w}\|^2 &< C^2. \end{aligned}$$

These conditions are fulfilled by $\mathbf{r} = 0$ and $\mathbf{w} = B\mathbf{g}$, hence we can use the KKT's equations to obtain necessary conditions on the optimal solutions of Problem (B.1). From the KKT's condition on the gradient component with respect to $\mathbf{r}$

$$\frac{\partial F}{\partial \mathbf{r}} = \lambda \frac{\partial (\|\mathbf{r}\|^2 - R^2)}{\partial \mathbf{r}} \Leftrightarrow 2\eta(\mathbf{g} + \mathbf{r}) - 2\mathbf{w} = \lambda 2\mathbf{r}$$

with $\lambda \in \mathbb{R}^+$.

If $\eta = \lambda$, then $\mathbf{w} = \eta\mathbf{g}$, and thus the objective function is simplified to $\eta + \eta\|\mathbf{r}\|^2 - 2\eta$. In this case, any $\mathbf{r}$ such that $\|\mathbf{r}\| = R$ achieves the optimal value, being $\eta(R^2 - 1)$. Otherwise,

$$\mathbf{r} = \frac{\mathbf{w} - \eta\mathbf{g}}{\eta - \lambda}. \tag{B.15}$$

The objective function can be written as

$$F(\mathbf{r}, \mathbf{w}) = \eta + 2\langle \eta\mathbf{g} - \mathbf{w}, \mathbf{r} \rangle + \eta\|\mathbf{r}\|^2 - 2\langle \mathbf{w}, \mathbf{g} \rangle.$$

Let $(\mathbf{r}^*, \mathbf{w}^*)$ be an optimal solution of Problem (B.1). We prove the following facts by contradiction:

a) $\langle \mathbf{r}^*, \eta\mathbf{g} - \mathbf{w}^* \rangle \geq 0$.

Assume $\langle \mathbf{r}^*, \eta\mathbf{g} - \mathbf{w}^* \rangle < 0$. Consider $\tilde{\mathbf{r}} = -\mathbf{r}^*$, the solution $(\tilde{\mathbf{r}}, \mathbf{w}^*)$ is feasible since $\|\tilde{\mathbf{r}}\| = \|\mathbf{r}^*\| \leq R$. Moreover, $\langle \eta\mathbf{g} - \mathbf{w}^*, \tilde{\mathbf{r}} \rangle > 0 > \langle \eta\mathbf{g} - \mathbf{w}^*, \mathbf{r}^* \rangle$, this yields

$$F(\mathbf{r}^*, \mathbf{w}^*) - F(\tilde{\mathbf{r}}, \mathbf{w}^*) = \langle \eta\mathbf{g} - \mathbf{w}^*, \mathbf{r}^* \rangle - \langle \eta\mathbf{g} - \mathbf{w}^*, \tilde{\mathbf{r}} \rangle < 0.$$

The last inequality violates the optimality of $\mathbf{r}^*$, therefore the claim is proved.

b) $\|\mathbf{r}^*\| = R$.

Assume $\|\mathbf{r}^*\| < R$. Consider $\tilde{\mathbf{r}} = R\frac{\mathbf{r}^*}{\|\mathbf{r}^*\|}$, using point (a) we have

$$\langle \eta\mathbf{g} - \mathbf{w}^*, \tilde{\mathbf{r}}\rangle - \langle \eta\mathbf{g} - \mathbf{w}^*, r^*\rangle = \left(\frac{R}{\|\mathbf{r}^*\|} - 1\right)\langle \eta\mathbf{g} - \mathbf{w}^*, \mathbf{r}^*\rangle \geq 0.$$

Moreover, $R = \|\tilde{\mathbf{r}}\| > \|\mathbf{r}^*\|$, hence

$$F(\mathbf{r}^*, \mathbf{w}^*) - F(\tilde{\mathbf{r}}, \mathbf{w}^*) = 2\left[\left(\frac{R}{\|\mathbf{r}^*\|} - 1\right)\langle \eta\mathbf{g} - \mathbf{w}^*, \mathbf{r}^*\rangle\right] + \eta(\|\mathbf{r}^*\| - R) < 0.$$

The last inequality violates the optimality of $\mathbf{r}^*$ and hence the result follows.

c) $\langle \mathbf{w}^*, \mathbf{g}\rangle = B$.

Assume $\langle \mathbf{w}^*, \mathbf{g}\rangle > B$. Then $\tilde{\mathbf{w}} = \mathbf{w}^* - (\langle \mathbf{w}^*, \mathbf{g}\rangle - B)\mathbf{g}$, $(\mathbf{r}^*, \tilde{\mathbf{w}})$ is a feasible point since

$$\|\tilde{\mathbf{w}}\|^2 = \|\mathbf{w}^*\|^2 + (\langle \mathbf{w}^*, \mathbf{g}\rangle - B)^2 - 2\left[\langle \mathbf{w}^*, \mathbf{g}\rangle - B\right]\langle \mathbf{w}^*, \mathbf{g}\rangle < \|\mathbf{w}^*\|^2$$

and $\langle \tilde{\mathbf{w}}, \mathbf{g}\rangle = B$.

$$\begin{aligned} F(\mathbf{w}^*, \mathbf{r}^*) - F(\tilde{\mathbf{w}}, \mathbf{r}^*) =& 2\langle \tilde{\mathbf{w}} - \mathbf{w}^*, \mathbf{g} + \mathbf{r}^*\rangle = -2\langle [\langle \mathbf{w}^*, \mathbf{g}\rangle - B]\,\mathbf{g}, \mathbf{g} + \mathbf{r}^*\rangle \\ =& -2(\langle \mathbf{w}^*, \mathbf{g}\rangle - B) - 2\langle [\langle \mathbf{w}^*, \mathbf{g}\rangle - B]\mathbf{g}, \mathbf{r}^*\rangle \\ \leq& -2[\langle \mathbf{w}^*, \mathbf{g}\rangle - B](1 - \|\mathbf{g}\|\|\mathbf{r}^*\|) < 0. \end{aligned}$$

The last inequality violates the optimality of $\mathbf{w}^*$, this completes the proof of this part.

It follows from Eq. (B.15), (a) and (b) that

$$\mathbf{r}^* = \alpha(\eta\mathbf{g} - \mathbf{w}^*) \;\text{ with }\; \alpha = \frac{R}{\|\eta\mathbf{g} - \mathbf{w}^*\|}. \tag{B.16}$$

The desired result follows from (B.16), (B.2) ,

$$\begin{aligned} F(\mathbf{w}^*, \mathbf{r}^*) &= \eta(1 + R^2) + 2R\sqrt{\eta^2 + \|\mathbf{w}\|^2 - 2\eta B} - 2B \\ &\leq \eta(1 + R^2) + 2R\sqrt{\eta^2 + c^2 - 2\eta B} - 2B. \end{aligned}$$

■

We now present the proof of Corollary B.1.0.1

Proof. Assume $d \geq 2$ and $\|\mathbf{w}^*\| < C$. Consider $\mathbf{y}$ s.t. $\langle \mathbf{g}, \mathbf{y}\rangle = 0$ and $\langle \mathbf{w}^*, \mathbf{y}\rangle \geq 0$. Since $d \geq 2$ we can assume, without loss of generality, that $\|\mathbf{y}\| = 1$. Furthermore, define $\tilde{\mathbf{w}} = \mathbf{w}^* + a\mathbf{y}$

with $a \in \mathbb{R}$, then $\exists\ a > 0\ s.t.\ \|\tilde{\mathbf{w}}\| = C$. The point $(\mathbf{r}^*, \tilde{\mathbf{w}})$ is a feasible point since $\langle \tilde{\mathbf{w}}, \mathbf{g} \rangle = \langle \mathbf{w}^*, \mathbf{g} \rangle = B$.

We consider two cases:

- $\mathbf{g}$ is not collinear with $\mathbf{w}^*$.

In this case $\exists\, \mathbf{y}$ such that $\langle \mathbf{w}^*, \mathbf{y} \rangle > 0$. Eq. (B.16) yields

$$\begin{aligned} F(\mathbf{w}^*, \mathbf{r}^*) - F(\tilde{\mathbf{w}}, \mathbf{r}^*) &= 2\langle \tilde{\mathbf{w}} - \mathbf{w}^*, \mathbf{g} + \mathbf{r}^* \rangle \\ &= 2\langle a\mathbf{y}, \mathbf{g} + \mathbf{r}^* \rangle \\ &= 2\langle a\mathbf{y}, \mathbf{r}^* \rangle \\ &= 2\langle a\mathbf{y}, \alpha[\eta\mathbf{g} - \mathbf{w}^*] \rangle \\ &= -2a\alpha\langle \mathbf{y}, \mathbf{w}^* \rangle \\ &< 0. \end{aligned}$$

The last equation violates the optimality of $\mathbf{w}^*$, thus

$$\|\mathbf{w}^*\| = C. \tag{B.17}$$

- $\mathbf{g}$ is collinear with $\mathbf{w}^*$.

In this case (c) implies $\mathbf{w}^* = B\mathbf{g}$. Consider $\tilde{\mathbf{r}} = \beta[(\eta - B)\mathbf{g} - a\mathbf{y}]$, then

$$\begin{aligned} F(\mathbf{w}^*, \mathbf{r}^*) - F(\tilde{\mathbf{w}}, \tilde{\mathbf{r}}) =& 2\left[\langle \eta\mathbf{g} - \mathbf{w}^*, \mathbf{r}^* \rangle - \langle \eta\mathbf{g} - \tilde{\mathbf{w}}, \tilde{\mathbf{r}} \rangle - \langle \mathbf{g}, \mathbf{w}^* - \tilde{\mathbf{w}} \rangle\right] \\ =& 2\left\{\langle (\eta - B)\mathbf{g}, \mathrm{sgn}(\eta - B)R\mathbf{g} \rangle - \langle (\eta - B)\mathbf{g} - a\mathbf{y}, \beta\left[(\eta - B)\mathbf{g} - a\mathbf{y}\right] \rangle\right\} \\ =& 2\left\{\langle \mathrm{sgn}(\eta - B)(\eta - B)\mathbf{g}, R\mathbf{g} \rangle - \beta\langle (\eta - B)\mathbf{g} - a\mathbf{y}, (\eta - B)\mathbf{g} - a\mathbf{y} \rangle\right\} \\ =& 2\left\{|\eta - B|R - \beta\left[(\eta - B)^2 + a^2\right]\right\} \\ =& 2\left\{|\eta - B|R - R\sqrt{(\eta - B)^2 + a^2}\right\} \\ =& R\sqrt{(\eta - B)^2 + a^2}\left(\frac{|\eta - B|}{\sqrt{(\eta - B)^2 + a^2}} - 1\right) \\ <& 0, \end{aligned}$$

where the second equality follows from $\mathbf{w}^* = B\mathbf{g}$, Eq. (B.16) and $\langle \mathbf{g}, \mathbf{y} \rangle = 0$, and the fourth and fifth equalities are implied by $\langle \mathbf{g}, \mathbf{y} \rangle = 0$. The last equation violates the optimality of

$(\mathbf{w}^*, \mathbf{r}^*)$, thus (B.17) must be fulfilled. The desired result then follows,

$$F(\mathbf{w}^*, \mathbf{r}^*) = \eta(1 + R^2) + 2R\sqrt{\eta^2 + c^2 - 2\eta B} - 2B.$$

■

Corollary 4.1.1

Proof. Minimizing the bound in Theorem 6.4.1 is equivalent to minimizing

$$\max\left\{\frac{\eta}{2}\left[1 + \left(R + \frac{S}{\eta}\right)^2\right], \frac{\eta}{2}\left[1 - \left(R + \frac{S}{\eta}\right)^2\right] + c\left[R + \frac{S}{\eta}\right]\right\}.$$

Define the function $G(\eta)$ as

$$\begin{aligned} G(\eta) :=& \max\left\{\frac{\eta}{2}(1 + R^2 + \frac{S^2}{\eta^2} + \frac{2SR}{\eta}), \frac{\eta(1 - R^2 - \frac{S^2}{\eta^2} - \frac{2RS}{\eta})}{2} + Rc + \frac{Sc}{\eta}\right\} \\ =& \max\left\{\frac{\eta}{2}(1 + R^2) + \frac{S^2}{2\eta} + SR, \frac{\eta}{2}(1 - R^2) - \frac{S^2}{2\eta} - RS + Rc + \frac{Sc}{\eta}\right\}. \end{aligned}$$

Note that

$$G(\eta) = \begin{cases} \frac{\eta}{2}(1 + R^2) + \frac{S^2}{2\eta} + SR & \text{if } t \geq \frac{c-S}{R} = \eta_3, \\ \frac{\eta}{2}(1 - R^2) - \frac{S^2}{2\eta} - RS + Rc + \frac{Sc}{\eta} & \text{otherwise.} \end{cases}$$

The minimum of the first part is achieved at $\eta_1 = \frac{S}{\sqrt{1+R^2}}$ while the minimum of the second part is achieved at $\eta_2 = \sqrt{\frac{S(c-2S)}{1-R^2}}$. Since $G(\eta)$ is the maximum of two convex functions, it attains its global minimum in the set $\{\eta_1.\eta_2, \eta_3\}$.

■

B.5 Proof of Theorem 6.4.2

The theorem follows immediately from Eq. (B.11) if $\mathbf{w}_{k+1} = \mathsf{P}_\Omega(\mathbf{w}_k - \eta\hat{\mathbf{g}}_k + \mathbf{s}_k)$. However, the key to prove Theorem 6.4.2 is $\|\mathbf{w}_{k+1} - \mathbf{w}^*\| \leq \|\mathbf{w}_k - \mathbf{w}^*\|$, $\forall\, 0 < k < \tilde{k}$ where $\tilde{k} = \inf\{k |\, f(\mathbf{w}_k) > f^* + L\,[\Gamma(C_0)]^p\}$. It therefore suffices to establish this last inequality to complete the proof of Theorem 6.4.2.

Proof. Fix $K > 0$. Suppose $\delta > 0$ and $f(\mathbf{w}_k) > f^* + L(\Gamma(C_0) + \delta)^p, \quad \forall\, k \leq K$. It is shown

by induction that

$$\|\mathbf{w}_k - \mathbf{w}^*\| \leq C_0 - 2\eta\delta k, \quad \forall\, k \leq K.$$

The case $k = 0$ is trivial.

Suppose $\|\mathbf{w}_j - \mathbf{w}^*\| \leq C_0 - 2\eta\delta j \leq C_0,\ \forall j \leq k$, then the argument of the proof Theorem 6.4.1 can be repeated with $c = C_0$. This is possible because the projection operator in the definition of $\mathbf{w}_{k+1}$ of Theorem 6.4.1 was needed only to bound the norm of $\|\mathbf{w}_k - \mathbf{w}^*\|$ when Lemma B.1.1 is used. Using a similar argument leading to Eq. (B.11) and the induction hypothesis, we have

$$\|\mathbf{w}_{k+1} - \mathbf{w}^*\|^2 \leq \|\mathbf{w}_k - \mathbf{w}^*\| - 2\eta\delta \leq C_0 - 2\eta\delta(k+1),$$

If $\delta \geq \frac{C_0}{2\eta K}$, the above inequality leads to a contradiction since $f(\mathbf{w}_K) > f^*$. Thus $\exists\, k \leq K\ s.t.\ f(\mathbf{w}_k) \leq f^* + L(\Gamma(C_0) + \delta)^p$. This completes the proof. ■

B.6 Proof of Theorem 6.4.3

Proof. Let $\mathcal{F}_n = \sigma\{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_n\}$ be the σ-algebra generated by $\{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_n\}$. Suppose $\mathbf{w}_n \notin \mathbf{\Omega}^*$, it follows from the definition of $\mathbf{w}_{n+1}$

$$\begin{aligned}
\|\mathbf{w}_{n+1} - \mathbf{w}^*\|^2 &\leq \|\mathbf{w}_n - \eta\hat{\mathbf{g}}(\mathbf{w}_n) - \eta\mathbf{r}_n - \mathbf{s}_n - \mathbf{w}^*\|^2 \\
&\leq \|\mathbf{w}_n - \mathbf{w}^*\|^2 - 2\eta\langle\hat{\mathbf{g}}(\mathbf{w}_n), \mathbf{w}_n - \mathbf{w}^*\rangle - 2\eta\langle\mathbf{w}_n - \mathbf{w}^*, \mathbf{r}_n\rangle \\
&\quad - 2\langle\mathbf{w}_n - \mathbf{w}^*, \mathbf{s}_n\rangle + \eta^2\|\hat{\mathbf{g}}(\mathbf{w}_n) + \mathbf{r}_n + \frac{\mathbf{s}_n}{\eta}\|^2.
\end{aligned}$$

Taking conditional expectation given $\mathcal{F}_n$

$$\begin{aligned}
\mathbb{E}\left\{\|\mathbf{w}_{n+1} - \mathbf{w}^*\|^2|\mathcal{F}_n\right\} &\leq \|\mathbf{w}_n - \mathbf{w}^*\|^2 - 2\eta\mathbb{E}\left\{\langle\hat{\mathbf{g}}(\mathbf{w}_n), \mathbf{w}_n - \mathbf{w}^*\rangle|\mathcal{F}_n\right\} \qquad \text{(B.18)} \\
&\quad - 2\eta\mathbb{E}\left\{\langle\mathbf{w}_n - \mathbf{w}^*, \mathbf{r}_n\rangle|\mathcal{F}_n\right\} \\
&\quad - 2\mathbb{E}\left\{\langle\mathbf{w}_n - \mathbf{w}^*, \mathbf{s}_n\rangle|\mathcal{F}_n\right\} + \eta^2\mathbb{E}\left\{\|\hat{\mathbf{g}}(\mathbf{w}_n) + \mathbf{r}_n + \frac{\mathbf{s}_n}{\eta}\|^2|\mathcal{F}_n\right\} \\
&\leq \|\mathbf{w}_n - \mathbf{w}^*\|^2 - 2\eta\left(\frac{f(\mathbf{w}) - f^*}{L}\right)^{\frac{1}{p}} - 2\eta\langle\mathbf{w}_n - \mathbf{w}^*, \mathbb{E}\left\{\mathbf{r}_n\right\}\rangle \\
&\quad - 2\langle\mathbf{w}_n - x^*, \mathbb{E}\left\{\mathbf{s}_n\right\}\rangle + \eta^2\left(1 + \mathbb{E}\left\{\|\mathbf{r}_n\|^2\right\} + 2\mathbb{E}\left\{\langle\hat{\mathbf{g}}(\mathbf{w}_n), \mathbf{r}_n\rangle|\mathcal{F}_n\right\}\right) \\
&\quad + \mathbb{E}\left\{\|\mathbf{s}_n\|^2\right\} + 2\eta\mathbb{E}\left\{\langle\hat{\mathbf{g}}(\mathbf{w}_n), \mathbf{s}_n\rangle|\mathcal{F}_n\right\} + 2\eta\mathbb{E}\left\{\langle\mathbf{s}_n, \mathbf{r}_n\rangle\right\} \\
&\leq \|\mathbf{w}_n - \mathbf{w}^*\|^2 - 2\eta\left(\frac{f(\mathbf{w}) - f^*}{L}\right)^{\frac{1}{p}} + \eta^2\left(1 + d\sigma_r^2\right) + d\sigma_s^2, \qquad \text{(B.19)}
\end{aligned}$$

where Lemma B.6.1 is used to derive the second inequality. The third inequality follows from the ortogonality of $\mathbf{g}(\mathbf{w}_n)$, $\mathbf{r}_n$, $\mathbf{s}_n$ and the fact that $\mathbb{E}\{\mathbf{r}_n\} = \mathbb{E}\{\mathbf{s}_n\} = 0$. Consider now $\delta > 0$ and

$$\boldsymbol{\Omega}_\delta = \boldsymbol{\Omega} \cap \mathcal{S}_{f,a},$$

with $a = f^* + L\left[\frac{\eta}{2}(1 + d\sigma_r^2) + \frac{d\sigma_s^2}{2\eta} + \delta\right]^p$. Since f is continuos, $\exists \mathbf{y} \in \mathbb{R}^d |\ \mathbf{y}_\delta \in \boldsymbol{\Omega}$ and $f(\mathbf{y}_\delta) = f^* + L(\delta)^p$. Define the process

$$\tilde{\mathbf{w}}_{k+1} = \begin{cases} \mathsf{P}_\Omega\left[\eta \mathbf{w}_k - \eta \hat{\mathbf{g}}\left(\tilde{\mathbf{w}}_k\right)\right], & \text{if } \tilde{\mathbf{w}}_k \notin \boldsymbol{\Omega}_\delta, \\ \mathbf{y}_\delta & \text{otherwise.} \end{cases}$$

We show that $\tilde{\mathbf{w}}_k \notin \boldsymbol{\Omega}_\delta$, $\forall\, k > K$, $K \in \mathbb{N}$ leads to a contradiction. Without loss of generality we consider $K = 0$. Assume $\tilde{\mathbf{w}}_k \notin \boldsymbol{\Omega}_\delta$ for any k and let $\hat{\mathcal{F}}_k = \{\tilde{\mathbf{w}}_0, \tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_k\}$. Since $f(\tilde{\mathbf{w}}_k) \geq f^* + L\left[\frac{\eta}{2}(1 + d\sigma_r^2) + \frac{d\sigma_s^2}{2\eta} + \delta\right]^p$, using Eq. (B.19) it follows that $\forall\, \mathbf{w}^* \in \boldsymbol{\Omega}^*$,

$$\begin{aligned} \mathbb{E}\left\{\|\tilde{\mathbf{w}}_{k+1} - \mathbf{w}^*\|^2 | \,\hat{\mathcal{F}}_k\right\} &\leq \|\tilde{\mathbf{w}}_k - \mathbf{w}^*\|^2 - 2\eta\left[\left(\frac{f(\tilde{\mathbf{w}}_k) - f^*}{L}\right)^{\frac{1}{p}} - \frac{t}{2}(1 + d\sigma_r^2) - \frac{d\sigma_s^2}{2\eta}\right] \\ &\leq \|\tilde{\mathbf{w}}_k - \mathbf{w}^*\|^2 - 2\eta\delta. \end{aligned}$$

Theorem B.6.2 then implies $\sum_{k=0}^{+\infty} 2\eta\delta < +\infty$. This is a contradiction and hence the proof is complete. ■

Lemma B.6.1. *(Lemma 2.4 of [74]). Let be $\mathbf{w} \notin \boldsymbol{\Omega}^*$ and $\hat{\mathbf{g}}(\mathbf{w})$ a unit noisy quasi subgradient of f at $\mathbf{w}$. Then, $\forall\, \mathbf{w}^* \in \boldsymbol{\Omega}^*$ it holds that, given any $\mathcal{F} = \sigma\{\mathbf{w}_1, ..., \mathbf{w}_k\}$ where $\mathbf{w}_i$ are random variables $\forall i = 1, .., k$, then*

$$\mathbb{E}\left\{\langle \hat{g}(\mathbf{w}), \mathbf{w}^* - \mathbf{w}\rangle | \mathcal{F}\right\} \geq \left(\frac{f(\mathbf{w}) - f^*}{L}\right)^{\frac{1}{p}}.$$

Lemma B.6.2. *(Lemma 2.5 of [74]). Let $\{Y_k\}$, $\{Z_k\}$ and $\{W_k\}$ be three sequences of non-negative random, and let $\{\mathcal{F}_k\}$ be a filtration. Suppose that the following conditions are satisfied for each k:*

a) Y_k, Z_k and W_k are functions of the random variables in $\mathcal{F}_k$;

b) $\mathbb{E}\{Y_{k+1} | \mathcal{F}_k\} \leq Y_k - Z_k + W_k$;

c) $\sum_{k=0}^{\infty} W_k < +\infty$.

Then $\sum_{k=0}^{\infty} Z_k < +\infty$*, and the sequence* $\{Y_k\}$ *converges to a nonnegative random variable* Y*, almost surely.*