UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Automatica, Ottimizzazione e Sistemi Complessi

Novel approaches to improve the efficiency of Machine Learning models

Candidate
Enrico Civitelli

Supervisors
Prof. Fabio Schoen

Prof. Marco Sciandrone

PhD Coordinator
Prof. Fabio Schoen

ciclo XXXV, 2019-2022

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Engineering. Copyright © 2023 by Enrico Civitelli.

To all the people I care...

Acknowledgments

Well, at the very end, this moment arrived. I am sitting in one of the laboratory chairs, which is one of the best places to write these last words. This part can seem easy to write, but it is not: writing this part means that this incredible journey, and a part of my life, is finished.

Fabio and Marco, I need to thank you. I had the pleasure of working in a laboratory that mixes high-level research with a friendly environment, a place in which you like to stay and work. You created this unique laboratory and gave me countless occasions to grow as a researcher. In addition, your passion and knowledge have been an inspiration to me. Finding your research path during a Ph.D. is never an easy job, but thanks to your guidance, I found the perfect spot. During these years, I have grown professionally and as a person.

Thanks to all the Verizon Connect (Florence) people. Collaborating with you has been an honor. I met wonderful people from which I learned more than I could ever imagine. I am sure that this experience with you improved me in every aspect. Thank you for all your patience and time.

Other than them, I have to thank numerous people.

I will start with my family. I did my best to return, at least partially, all the unconditional trust they put in me. There is not much else that I can say without becoming banal: thank you for all your support. This achievement is also yours.

Of course, this section can only be considered complete by mentioning all my lab buddies. Ale, Leo, Matte, Pier, Simo, Tommi A., and Tommi L., you have been the people with which I share most of my time inside Florence's laboratory. Still, despite this, it is difficult for me to think about you as just colleagues. I see you as friends, friends with which I would like to spend many more years discussing stupid ideas and hanging out together. I do not know if it is easy to find such an environment in a "work" place, but I am happy to have found this here. Francesco, Marco, and Tomaso, you are the new entries in the laboratory. I am sorry I did not spend a lot of time with you. In any case, it has been a pleasure to meet you. I know the laboratory is in good hands. Matteo, Pier, and Tommi L.: I think I learned a lot of Ph.D. life lessons in the most complicated way, but in the end, I am happy about my journey, and more importantly, I am glad to have shared part of it with you. Matteo, I think you already know everything. I just want to add a couple of words: I will never truly understand all the trust you put

in me. Despite everything, you were always there for me. I do not have the words to express everything I feel, but I hope I have been able to pay you back at least partially. Matteo, I will do my best to allow us to continue our journey.

Ale A., Aurel, Giuli, Luca, Matte N., and Vishal, I met you during the bachelor/master's degrees. Meeting friends like you in that period is great because all these university years have been, for sure, less heavy thanks to you. I will never forget all the dinners, holidays, and laughs we had together.

I cannot forget to mention all the incredible people I met in Barcelona inside and outside CVC (really too many to name everyone, sorry), plus Pietro. I would like to remember that once, one of them said (wrote, to be honest): "wherever you will be, remember that you have a home in Barcelona". Well, you have to remember that, believe it or not, you helped me in a way you could never imagine, and for this, you will always have my deepest gratitude. Thank you, my friends, really.

Pietro, I just would like to add a couple of words. Living with you for six months has been amazing. Thank you for all the moments we shared. In the end, I think these will be among the things we will remember with more happiness (and, of course, thanks for cooking).

Diego, Fede P., Fede T., Nao, Nico, and Uba; a part of this section is for you. We had a lot of beautiful times together. Here, I just would like to report an episode: I will never forget the video you did for me when I left for my visiting period in Barcelona. I know it is stupid to say, but it showed me the importance of our bonds better than a thousand of words.

Azzurra, despite everything, you deserve a space here. Thank you very much for everything: if I am here, it is also because of you. You helped me in countless ways. The least that I can do is write these few words.

To conclude, I would like to thank my oldest friends in Arezzo (and the outskirts). Guys, I really do not have words to describe our relationship. In my opinion, it is not easy to define someone as a real friend, you, for sure, definitely fit this definition for me. I have been able to finish my studies also because of you. Again, thank you.

Okay, I think I am done. I tried my best to write these words, and I really hope I gave all of you an idea of how important you are to me. I shared billions of meaningful moments with you: from the good ones to the darkest ones, the ones where it rained too hard to stand up. You have been the best friends I could ever desire for this journey. I sincerely thank you for

all of these reasons (even if, probably, there are many more).

Enrico

Abstract

In this thesis, we discuss the problem of efficiency in Machine Learning from a general point of view. Specifically, we relate efficiency to the resources/time used in training or testing and propose four possible ways to increase it. Generally speaking, efficiency can be increased using two main strategies: reducing the model's dimensions or selecting a subset of good input features.

The first part of this dissertation concerns the problem of best subset selection in logistic regression. As a matter of fact, in some contexts, acquiring the input features can be hard. In these situations, we need to apply methods designed to obtain good prediction performances using only a subset of features, since trying to reduce the model complexity has no point. In this thesis we propose a feature selection algorithm based on a piece-wise linear approximation of the logistic function in conjunction with an optimization algorithm solved by means of a two-block decomposition strategy.

The second part of this thesis is concerned with the problem of pruning the nodes of fully connected or the filters in convolutional layers in neural network architectures. Nowadays, deep learning techniques are applied in many different fields. For this reason, having algorithms designed to reduce the model's complexity is extremely useful for researchers to increase the applicability of their models. In this second part of the thesis, we develop two algorithms based on different approaches. Specifically, node pruning is built on top of the recently released neural network training on the spectral domain. In contrast, channel pruning is a preliminary analysis based on a novel bilevel approach that takes inspiration from neural architectural search approaches.

The third part of this thesis concerns the problem of removing Batch Normalization in ResNet-like models. We show that weight initialization is key to training ResNet-like normalization-free networks. In particular, we propose an effective initialization strategy for a slightly modified residual block.

For all these parts, we show both theoretical and empirical results to strengthen the soundness of the proposed approaches.

Contents

Chapter 1

Introduction

In the last decades, Machine Learning has been one of the major active and prolific fields of research in computer science. From the first seminal works on this field [85, 125], in which the applications were limited to specific sectors, we moved to research and applications [40, 47, 73] with significant impact on everyday life. To give an example of the growth in this field, in 1990, Carnegie Mellon University released one of the first papers on autonomous driving [41] based on an experiment conducted a few years before[1]. Of course, as it is possible to imagine, this prototype was minimal. Still, starting from this pioneering work, just 30 years later, we are starting to experience self-driving vehicles in our daily life. Besides this example, researchers applied Machine Learning techniques in many different scenarios.

One of the possible reasons behind this Machine Learning explosion and ubiquity we are witnessing in these last years can be found in the powerful hardware at our disposal nowadays. This computational power allows researchers to manage an incredible amount of data in a reasonable amount of time and to define (and use) more complex and resource-demanding Deep Learning models. Figures 1.1 and 1.2 give an idea of the rate of how the complexity is increasing over time. The two Figures refer to different topics. Figure 1.1 shows how, in less than one year, from models with less than 100 million parameters, we can now exploit models with more than 600 million parameters to solve the same image classification task. On the other hand, Figure 1.2 shows a (maybe even worst) trend for the language models. As a matter of fact, from 94 million parameters, we moved to more than 500

[1] `https://www.youtube.com/watch?v=ntIczNQKfjQ`

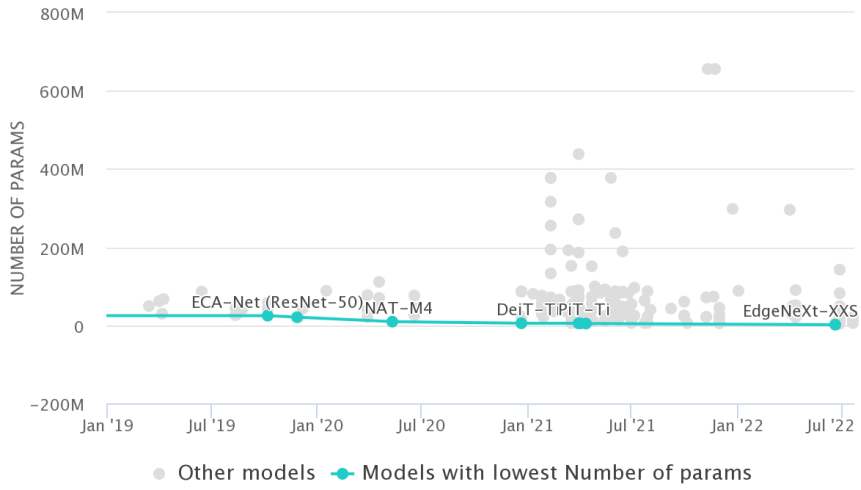


Figure 1.1: Models dimensions according to the year they have been released. This image has been adapted from "paperswithcode.com".

billion parameters in less than five years.

Looking at Figure 1.1 and 1.2 one question naturally arise: why are we increasing the models complexity at this rate?

Roughly speaking, as empirically proved by Figure 1.3, model complexity correlates to better predictive performance. In the context of image classification, Figure 1.3 shows how increasing the Giga FLOPs of a model (the number of floating point operations needed to compute the output) leads to best prediction performances. To correctly link these three figures, it is worth recalling that a higher number of FLOPs correlates to increased parameters. Moreover, in Deep Learning, over-parameterized models often benefit from nice generalization and theoretical properties [4, 33, 121, 127].

Considering these two aspects and recalling that the hardware at our disposal allows us to deal with big models, it is easy to imagine why there is this tendency to design and test models with many parameters.

Despite these significant benefits over-parametrized networks enjoy, it is essential to mention some non-negligible drawbacks. Huge models need a long training time and a great amount of resources. Moreover, and probably this is the most critical issue, in some scenarios, it is not possible to always have the necessary amount of computational power to run a model

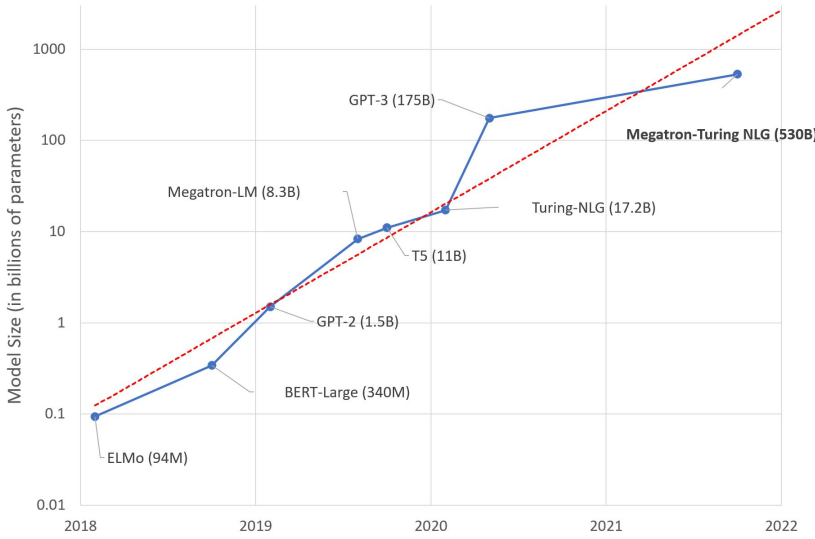


Figure 1.2: Language model dimension according to the year they have been released. This image has been adapted from "huggingface.co"

in a reasonable amount of time. For instance, in contexts such as robotics, autonomous driving, or edge computing, the available computational power has to be shared with other essential processes. In some cases, using high-end hardware is also not possible due to energy consumption constraints. In other words, despite their outstanding performance, huge models often have too high computational times to be used in such applications.

In addition to the problems mentioned above, in this dissertation, we would like to highlight also a different way to define the complexity of using a Machine Learning model. In some contexts (for instance, healthcare applications), even if the computational resources are not a problem, acquiring the input data can be hard (for example, the reader can imagine the time necessary to obtain the results of some medical analysis or the complexity to get some samples to analyze). To increase the efficiency in these situations, we need to apply methods designed to obtain good prediction performance using only a subset of features since trying to reduce the model complexity has no point.

Recently, researchers put a lot of effort into designing solutions as ef-

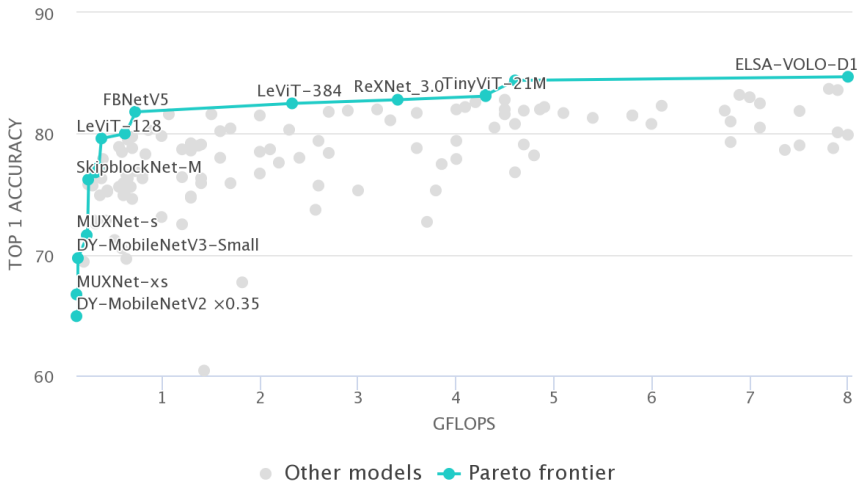


Figure 1.3: Comparison between Giga FLOPS (GFLOPs) and reached Top1 accuracy on ImageNet.

ficiently as possible to avoid the above-mentioned problems, either designing efficient architectures, reducing the complexity of existing models, using fewer resources dependent algorithms, or selecting the input features [13, 21, 29, 47, 51, 69, 74, 90, 101, 112, 136, 137].

As mentioned above, we can approach the problem of designing efficient Machine Learning models following different paradigms. Since this dissertation proposes four different ways to reduce model's complexity (in training or testing time), we structured the manuscript as self-contained chapters from which the reader can refer to deepen a specific approach. Despite the self-contained structure of each Chapter, we would like to highlight that Chapters 2, 3, and 4 deal with the same underlying problem of fitting the best possible model taking into consideration an ℓ_0 pseudo-norm as guidance. As a matter of fact, inducing sparsity in the model's parameters or input features leads to an improvement in both memory or time requirements.

Finally, we organized the rest of this work as follows:

- In Chapter 2, we address the problem of feature selection. As discussed before, one possible way to decrease the difficulties of applying a model in real-world cases is related to the hardness of obtaining the features

needed to perform a reliable solution in some scenarios. In these situations, a robust method to select only a subset of good features is desirable even for simple Machine Learning models. More specifically, in this Chapter, we focus on the problem of best subset selection in logistic regression exploiting the well-known Information Criteria to balance the predictive performance and the number of features used. This Chapter is adapted from the work [31];

- In Chapter 3, we address the problem of Neural Network pruning, and, in particular, we focus on structural pruning of fully connected layers. Generally speaking, fully connected layers have the highest number of parameters among the possible modules that define a Neural Network. Despite the tremendous computational efficiency of fully connected layers, the tendency is to design, when possible, fully convolutional networks because of the reduced memory required. Anyhow, in some cases, fully connected layers may be mandatory (for instance, LSTM or Transformer models). In this Chapter, exploiting a spectral reformulation of the network, we design a framework to rank the importance of each output node for the final prediction according to their associated eigenvalue. This Chapter is adapted from the work [23];

- In Chapter 4, we address the problem of pruning convolutional layers to decrease the inference time. More specifically, it is possible to define a bilevel optimization problem to address the issue of searching, given an initial network architecture, a sub-network with a reduced number of convolutional filters. In this Chapter, we develop, under suitable assumptions, a convergence analysis of a Penalty Decomposition approach applied to such a bilevel problem. It is also important to highlight that this Chapter, differently from the others, has to be intended as a work-in-progress work and a natural complement to the work described in Chapter 3. For this reason, here we present the theoretical analysis and some proof of concept results;

- In Chapter 5, we address the problem of removing Batch Normalization from ResNet-like model. Batch Normalization has allowed the training of extremely deep neural networks due to its effect on preventing exploding or vanishing gradients. A direct consequence of well-behaved gradients yielded is an "easiest" training phase. In this Chapter, we theoretically analyze the behavior of the gradient inside a residual block

and, according to the performed analysis, we designed an initialization strategy that, in conjunction with a minor modification of the standard ResNet structure, allows successfully training of the network without Batch Normalization. This Chapter is adapted from the work [32];

- In Chapter 6, we draw some concluding remarks about the proposed approaches.

Chapter 2

Best Feature Selection in Logistic Regression

In this chapter, the problem of best subset selection in logistic regression is addressed. In particular, we take into account formulations of the problem resulting from the adoption of information criteria, such as AIC or BIC, as goodness-of-fit measures. There exist various methods to tackle this problem. Heuristic methods are computationally cheap, but are usually only able to find low quality solutions. Methods based on local optimization suffer from similar limitations as heuristic ones. On the other hand, methods based on mixed integer reformulations of the problem are much more effective, at the cost of higher computational requirements, that become unsustainable when the problem size grows. We thus propose a new approach, which combines mixed-integer programming and decomposition techniques in order to overcome the aforementioned scalability issues. We provide a theoretical characterization of the proposed algorithm properties. The results of a vast numerical experiment, performed on widely available datasets, show that the proposed method achieves the goal of outperforming state-of-the-art techniques.

2.1 Preamble

In this Chapter, we are interested in the problem of best features subset selection in logistic regression. This variant of standard logistic regression requires to find a model that, in addition to accurately fitting the data, exploits a limited number of features. In this way, the obtained model only employs the most relevant features, with benefits in terms of inference performance (since some features are not necessary for good predictions) and interpretable. Moreover, generally speaking, Logistic regression possesses a number of useful properties. As example, it is relatively simple; it is readily interpretable (since the weights are linearly associated to the features); statistical confidence measures can quickly be obtained; the model can be updated by simple gradient descent steps if new data are available; in practice it often has good predictive performance, especially when the size of train data is too limited to exploit more complex models.

In order to compare the quality of models that exploit different features, i.e., models with different complexity, goodness-of-fit (GOF) measures have been proposed. These measures allow to evaluate the trade-off between accuracy of fit and complexity associated with a given model. Among the many GOF measures [16, 17, 74] that have been proposed in the literature, those based on information criteria (IC) [3, 60, 130] are some of the most popular [68]. Models based upon these Information Criteria are very popular in the statistics literature.

In case the selection of the model is based on one of the aforementioned IC, the underlying optimization problem consists of minimizing a function which is the sum of a convex part (the negative log-likelihood) and a penalty term, proportional to the number of employed variables; it is thus a sparse optimization problem.

Problems of this kind are often solved by heuristic procedures [44] o by ℓ_1-regularization [75, 87, 149]. In fact, specific optimization algorithms exist to directly handle the zero pseudo-norm [11, 95, 98]. However, none of the aforementioned methods is guaranteed to find the best possible subset of features under a given GOF measure.

With problems where the convex part of the objective is simple, such as least squares linear regression, approaches based on mixed-integer formulations allow to obtain certified optima, and have thus had an increased popularity in recent years [18, 39, 54, 107, 108]. Logistic likelihood, although convex, cannot however be inserted in a standard MIQP model. Still, [128]

showed that, by means of a cutting-planes based approximation, a good surrogate MILP problem can be defined and solved, at least for moderate problem sizes, providing a high quality classification model.

The aim of this Chapter is to introduce a novel technique that, exploiting mixed-integer modeling, is able to produce good solutions to the best subset selection in logistic regression problem, being at the same time reasonably scalable with respect to problem size. To reach this goal, we make use of a decomposition strategy.

The main contributions described in this Chapter are:

- The definition of a strong necessary optimality condition for optimization problems with an ℓ_0 penalty term;
- The definition of a decomposition scheme, with a suitable variables selection rule, allowing to improve the scalability of the method from [128], with guarantees of convergence to points satisfying the aforementioned condition;
- Practical suggestions to improve the performance of the proposed algorithm.

2.2 Preliminary

Let $X \in \mathbb{R}^{N \times n}$ be a dataset of N examples with n real features and $Y \in \{-1, 1\}^N$ a set of N binary labels. The *logistic regression model* [61] for binary classification defines the probability for an example x of belonging to class $y = 1$ as

$$\mathbb{P}(y = 1 \mid x) = \frac{1}{1 + \exp(-w^\top x)}.$$

Substantially, a sigmoid nonlinearity is applied to the output of a linear regression model. Note that the intercept term is not explicitly present in the linear part of the model; in fact, it can be implicitly added, by considering it as a feature which is equal to 1 in all examples; we did so in the experimental part of this work. It is easy to see that

$$\mathbb{P}(y = -1 \mid x) = 1 - \mathbb{P}(y = 1 \mid x) = \frac{1}{1 + \exp(w^\top x)}.$$

Hence, the logistic regression model can be expressed by the single equation here below:

$$\mathbb{P}(y \mid x) = \frac{1}{1 + \exp(-yw^\top x)}. \tag{2.1}$$

Under the hypothesis that the distribution of $(y \mid x)$ follows a Bernoulli distribution, we get that model (2.1) is associated with the following *log-likelihood* function:

$$\ell(w) = -\sum_{i=1}^{N} \log\left(1 + \exp\left(-y^{(i)} w^\top x^{(i)}\right)\right). \tag{2.2}$$

A function $f(v) = \log(1 + \exp(-v))$ is referred to as logistic loss function and is a convex function. The maximum likelihood estimation of (2.1), which requires the maximization of $\ell(w)$, is thus a convex continuous optimization problem.

Identifying a subset of features that provides a good trade-off between fit quality and model sparsity is a recurrent task in applications. Indeed, a sparse model might offer a better explanation of the underlying generating model; moreover, sparsity is statistically proved to improve the generalization capabilities of the model [145]; finally, a sparse model will be computationally more efficient.

Many different approaches have been proposed in the literature for the best subset selection problem which, we recall, is a specific form of model selection. Every model selection procedure has advantages and disadvantages as it is difficult to think that there might exist a single, correct, model for a specific application. Among the many different proposals, those which base subset selection on *information criteria* [25, 26, 77] stand out as the most frequently used, both for their computational appeal as well as for their deep statistical theoretical support. Information criteria are statistical tools to compare the quality of different models in terms of quality of fit and sparsity simultaneously. The two currently most popular information criteria are:

- the *Akaike Information Criterion* (AIC) [2, 3, 20]:

  $$\mathrm{AIC}(w) = -2\ell(w) + 2\|w\|_0;$$

  Comparing a set of candidate models, the one with smallest AIC is considered closer to the truth than the others. Since the log-likelihood,

at its maximum point, is a biased upward estimator of the model selection target [25], the penalty term $2\|w\|_0$, i.e., the total number of parameters involved in the model, allows to correct this bias;

- the *Bayesian Information Criterion* (BIC) [130]:

$$\mathrm{BIC}(w) = -2\ell(w) + \log(N)\|w\|_0;$$

It has been shown [25, 77] that given a set of candidate models, the one which minimizes the BIC is optimal for the data, in the sense that it is the one that maximizes the marginal likelihood of the data under the Bayesian assumption that all candidate models have equal prior probabilities.

Although other models can be proposed for model selection, those based on the AIC and BIC, or their variant, are extremely popular thanks to their solid statistical properties.

In summary, when referred to logistic regression models, the problem of best subset selection based on information criteria like AIC or BIC has the form of the following optimization problem:

$$\min_{w\in\mathbb{R}^n} \mathcal{F}(w) + \lambda\|w\|_0, \tag{2.3}$$

where $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ is twice the negative log-likelihood of the logistic regression model ($\mathcal{F}(w) = -2\ell(w)$), which is a continuously differentiable convex function, $\lambda > 0$ is a constant depending on the choice of the information criterion and $\|\cdot\|_0$ denotes the ℓ_0 semi-norm of a vector. Given a solution $\bar{w}$, we will denote the set of its nonzero variables, also referred to as *support*, by $S(\bar{w}) \subseteq \{1, \ldots, n\}$, while $\bar{S}(\bar{w}) = \{1, \ldots, n\} \setminus S(\bar{w})$, denotes its complementary. In the following, we will also refer to the objective function as $\mathcal{L}(w) = \mathcal{F}(w) + \lambda\|w\|_0$.

Because of the discontinuous nature of the ℓ_0 semi-norm, solving problems of the form (2.3) is not an easy task. In fact, problems like (2.3) are well-known to be $\mathcal{NP}$-hard, hence, finding global minima is intrinsically difficult.

[98] have established necessary first-order optimality conditions for problem (2.3); in fact, they consider a more general, constrained version of the problem. In the unconstrained case we are interested in, such conditions reduce to the following.

Definition 1. A point $w^\star \in \mathbb{R}^n$ satisfies Lu-Zhang first order optimality conditions for problem (2.3) if $\nabla_j \mathcal{L}(w^\star) = 0$ for all $j \in \{1, \ldots, n\}$ such that $w_j^\star \neq 0$.

As proved by [98], if $\mathcal{L}(w)$ is a convex function, as in the case of logistic regression log-likelihood, there is an equivalence relation between Lu-Zhang optimality and local optimality, meaning there exists a neighborhood V of $w^\star$ such that $\mathcal{L}(w^\star) \leq \mathcal{L}(w)$ for all $w \in V$.

Proposition 1. Let $w^\star \in \mathbb{R}^n$. Then, $w^\star$ is a local minimizer for Problem (2.3) if and only if it satisfies Lu-Zhang first order optimality conditions.

Proof. Being $\mathcal{L}$ convex, a Lu-Zhang point is globally optimal w.r.t. the nonzero variables. As for the zero variables, since $\mathcal{L}$ is continuous, there exists a neighborhood such that the decrease in $\mathcal{L}$ is bounded by λ, which is the penalty term that is added to the overall objective function as soon as one of the zero variables is moved. □

Unfortunately, the number of Lu-Zhang local minima is in the order of 2^n. Indeed, for any subset of variables, minimizing w.r.t. those components, while keeping fixed the others to zero, allows to obtain a point which satisfies Lu-Zhang conditions. Hence, satisfying the necessary and sufficient conditions of local optimality is indeed a quite weak feature in practice. On the other hand, being the search of an optimal subset of variables a well-known $\mathcal{NP}$-hard problem, requiring theoretical guarantees of global optimality is unreasonable. In conclusion, it should be clear that the evaluation and comparison of algorithms designed to deal with problem (2.3) have to be based on the quality of the solutions empirically obtained in experiments.

However, we can further characterize candidates for optimality by means of the following notion, which adapts the concept of Component Wise optimality (CW-Optimality) for cardinality constrained problems defined by [11]. To this aim, we introduce the notation $w_{\neq i}$ to denote all the components of w except the i-th.

Definition 2. A point $w^\star \in \mathbb{R}^n$ is a CW-minimum for Problem (2.3) if

$$w_i^\star \in \arg\min_{w_i} \mathcal{L}(w_i; w^\star_{\neq i}) \tag{2.4}$$

for all $i = 1, \ldots, n$.

Equivalently, (2.4) could be expressed as

$$\begin{aligned} w^\star \in \arg\min_{w} \ & \mathcal{L}(w) \\ \text{s.t. } & \|w - w^\star\|_0 \leq 1 \end{aligned} \tag{2.5}$$

CW-optimality is a stronger property than Lu-Zhang stationarity. We outline this fact in the following proposition.

Proposition 2. Consider Problem (2.3) and let $w^\star \in \mathbb{R}^n$. The following statements hold:

1. If $w^\star$ is a CW-minimum for (2.3), then $w^\star$ satisfies Lu-Zhang optimality conditions, i.e., $w^\star$ is a local minimizer for $w^\star$.

2. If $w^\star$ is a global minimizer for (2.3), then $w^\star$ is a CW-minimum for (2.3).

Proof. We prove the statements one at a time.

1. Let $w^\star$ be a CW-minimum, i.e.,

$$w_i^\star \in \arg\min_{w_i} \mathcal{L}(w_i; w_{\neq i}^\star) \tag{2.6}$$

for all $i = 1, \dots, n$. Assume by contradiction that $w^\star$ does not satisfy Lu-Zhang conditions; then, there exists $h \in \{1, \dots, n\}$ such that $w_h^\star \neq 0$ and $\nabla_h \mathcal{L}(w^\star) > 0$. Hence, $-\nabla_h \mathcal{L}(w^\star)$ is a descent direction for $\mathcal{L}(w_h; w_{h\neq j}^\star)$ at $w_h^\star \neq 0$, which contradicts (2.6).

2. Let $w^\star$ be a globally optimal point for (2.3). Assume by contradiction that $w^\star$ is not a CW-minimum, i.e., there exists $h \in \{1\dots, n\}$ such that there exists $\hat{w}_h$ such that $\mathcal{L}(\hat{w}_h; w_{h\neq j}^\star) < \mathcal{L}(w^\star)$. This clearly contradicts that $w^\star$ is a global optimum.

□

Note that CW-optimality is a sufficient, yet not necessary, condition for local optimality. Indeed, Lu-Zhang conditions, and hence local optimality, certify that an improvement cannot be achieved without changing the set of nonzero variables. CW-optimality allows to also take into account possible changes in the support, although limited to one variable. We show this in the following examples, where, for the sake of simplicity, we consider a simpler convex function than $\mathcal{L}$.

Example 1. Consider the problem

$$\min_{w\in\mathbb{R}^2} \varphi(w) = (w_1 - 1)^2 + (w_2 - 2)^2 + 2\|w\|_0.$$

It is easy to see that Lu-Zhang conditions are satisfied by the points $w^a = (0,0)$, $w^b = (1,2)$, $w^c = (0,2)$ and $w^d = (1,0)$. We have $\varphi(w^a) = 5$, $\varphi(w^b) = 4$, $\varphi(w^c) = 3$, $\varphi(w^d) = 4$. We can then observe that w^c and w^d are CW-minima, as their objective value cannot be improved by changing only one of their components, while w^a and w^b are not CW-optima, as the solutions can be improved by zeroing a component or setting the first component to 1, respectively.

We can conclude by remarking that searching through the CW-points allows to filter out a number of local minima that are certainly not globally optimal.

2.3 Related work

A number of techniques has been proposed and considered in the literature to tackle problem (2.3). If the number of variables n in not exceedingly large, especially in the case of convex $\mathcal{L}$, heuristic and even exhaustive approaches are a viable way of proceeding.

The *exhaustive approach* consists of finding the global minimum for $\mathcal{L}$ for all possible combinations of non-zero variables. All the retrieved solutions are then compared, adding to $\mathcal{L}$ the penalty term on the ℓ_0-norm, to identify the optimal solution to the original problem. This approach is however clearly computationally intractable.

In applications, an heuristic relaxation of the exhaustive search is employed: the greedy *step-wise approach*, with both its variants, the *forward selection* strategy and the *backward elimination* strategy [44]. This method consists of adding (or removing, respectively) a variable to the support, in such a way that the variation of the objective function obtained by only changing that variable is optimal; the procedure typically stops as soon as the addition (removal) of a variable is not enough to improve the quality of the solution. This technique is clearly much cheaper, at the cost of a lower quality of the final solution retrieved.

One of the most prominent approaches (arguably the most popular one) to induce sparsity is Lasso [138]. Lasso consists of approximating the ℓ_0 penalty term by a continuous, convex surrogate, the ℓ_1-norm. When applied to (2.3), the resulting optimization problem is the widely used ℓ_1-regularized

formulation of logistic regression [75, 87, 149]:

$$\min_{w \in \mathbb{R}^n} \mathcal{L}(w) + \lambda \|w\|_1. \tag{2.7}$$

The ℓ_1-norm is well known to be sparsity-inducing [7]. Lasso often produces good solutions with a reasonable computational effort and is particularly suited for large scale problems, where methods directly tackling the ℓ_0 formulation are too expensive to be employed. However, equivalence relationships between problems (2.3) and (2.7) do not exist. Thus, problem (2.7) usually has to be solved for many different values of λ in order to find a satisfying solution of (2.3). Still, the solution is typically suboptimal for problem (2.3) and poor from the statistical point of view [105, 132, 153].

[98] proposed a *Penalty Decomposition* (PD) approach to solve problem (2.3). The classical variable splitting technique [72] can be applied to problem (2.3), duplicating the variables, adding a linear equality constraint and separating the two parts of the objective function, obtaining the following problem:

$$\begin{aligned} \min_{w,z \in \mathbb{R}^n} \ & \mathcal{L}(w) + \lambda \|z\|_0 \\ \text{s.t. } & w - z = 0. \end{aligned} \tag{2.8}$$

Problem (2.8) can then be solved by an alternate exact minimization of the quadratic penalty function

$$q_\tau(w, z) = \mathcal{L}(w) + \lambda \|z\|_0 + \frac{\tau}{2} \|w - z\|_2^2, \tag{2.9}$$

where the penalty parameter τ is increased every time a (approximate) stationary point, w.r.t. the w block of variables, of the current q_τ is attained. The algorithm is summarized in Algorithm 1. The z-update step can in fact be carried out in closed form by the following rule:

$$z_i^{k+1} = \begin{cases} 0 & \text{if } \frac{\tau}{2}(w_i^k)^2 < \lambda, \\ w_i^{k+1} & \text{otherwise.} \end{cases}$$

The algorithm is proved to asymptotically converge to Lu-Zhang stationary points, i.e., to local minima. The solution retrieved by the algorithm strongly depends on the choice of the initial value of the penalty parameter τ and of the increase factor σ_τ. Therefore, in order to find good quality solutions, the algorithm may be run in practice several times with different hyperparameters configurations.

Algorithm 1: Penalty Decomposition

Input: $\tau > 0,\ \sigma_\tau > 1,\ w^0, z^0 \in \mathbb{R}^n,\ \varepsilon > 0,\ \eta > 0,\ \sigma_\varepsilon \in (0,1)$.
$k = 0$
while $\|w^k - z^k\| > \eta$ **do**
  Set
$$w^{k+1} = \arg\min_w \mathcal{L}(w) + \frac{\tau}{2}||w - z^k||^2$$
  Set
$$z^{k+1} = \arg\min_z \frac{\tau}{2}||w^{k+1} - z||^2 + \lambda||z||_0$$
  if $\|\nabla_w q_\tau(w^k, z^k)\| \leq \varepsilon$ **then**
    Set $\tau = \sigma_\tau\, \tau$
    Set $\varepsilon = \sigma_\varepsilon\, \varepsilon$
  $k = k + 1$
return z^k

A different approach exploits the fact that the ℓ_0 semi-norm can be approximated by the sum of a finite sum of scalar terms, each one being a surrogate for the step function. In particular, the scalar step function can be approximated, for $t > 0$, by the continuously differentiable concave function $s(t) = 1 - e^{-\alpha t}$, as done by [124] or [95]. Problem (2.3) can hence be reformulated as

$$\min_{w \in \mathbb{R}^n}\ \mathcal{L}(w) + \lambda \sum_{i=1}^{n} (1 - e^{-\alpha |w_i|}). \tag{2.10}$$

A sequence of problems of the form (2.10), for increasing values of α, can then be solved, producing a sequence of solutions that are increasingly good approximations of those of the original problem. In fact, in the computational practice, problem (2.10) is solved for a suitable, fixed value of α.

In recent years, very effective algorithms have been proposed in the literature to tackle the sparse logistic regression in its cardinality-constrained formulation, i.e., to solve the problem

$$\begin{aligned} &\min_{w \in \mathbb{R}^n}\ \mathcal{L}(w) \\ &\text{s.t. } \|w\|_0 \leq s, \end{aligned} \tag{2.11}$$

for fixed $s < n$. Among these methods, the most remarkable one is arguably

the *Outer Approximation method* [19,43], which was proposed to be used for problem (2.11) by [17]. The algorithm, which is briefly reported in Algorithm 2, works in an alternating minimization fashion. First, it exactly solves, through a mixed-integer solver, a cutting-plane based approximation of the problem; then, it finds the exact global minimum w.r.t. the support of the newly obtained solution. If the objective function of the MIP problem is within some pre-specified tolerance ϵ of the true objective function at the new iterate, then the algorithm stops, otherwise the obtained point is used to perform a new cut.

Algorithm 2: Outer Approximation Method

Data: $M \gg 0$, $w^0 \in \mathbb{R}^n$, $\nu^0 = -\infty$, $\epsilon > 0$

$k = 0$

while $\nu^k - \mathcal{L}(w^k) < \epsilon$ **do**

Set

$$\hat{\beta}, \nu^{k+1} \in \arg\min_{\beta, w} \beta$$
$$\text{s.t.} \quad -Mz_i \leq w_i \leq Mz_i \quad \forall i = 1, \ldots, n,$$
$$z \in \{0,1\}^n,$$
$$\sum_{i=1}^{n} z_i \leq s,$$
$$\beta \geq \mathcal{L}(w^\ell) + \nabla\mathcal{L}(w^\ell)^T(w - w^\ell) \;\; \forall \ell = 0, \ldots, k,$$

Set

$$w^{\ell+1} \in \arg\min_{w} \mathcal{L}(w)$$
$$\text{s.t.} \quad w_i = 0 \text{ for all } i \in \bar{S}(\nu^{k+1})$$

$k = k + 1$

return z^k

Algorithm 2 can be employed to solve problem (2.3), by running it for every possible value of $s = 1, \ldots, n$ and choosing, among the n retrieved solutions, the one with lowest IC value.

In fact, the algorithm can straightforwardly be adapted to directly handle

problem (2.3). To this aim it is sufficient to remove from the MIP subproblem the cardinality constraint and add it as a penalty term in the objective function.

Recently, [74] proposed an alternative way of using the outer approximation method, which is however based on the ℓ_2-regularized formulation of the logistic regression problem with cardinality constraints

$$\begin{aligned} \min_{w \in \mathbb{R}^n} \ & \mathcal{L}(w) + \frac{1}{2\gamma}\|w\|_2^2 \\ \text{s.t.} \ & \|w\|_0 \leq s. \end{aligned}$$

Applying duality theory, the optimal value obtainable for a fixed configuration z of nonzero variables, $c(z)$, can be computed by solving the problem

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^N} \ & -\sum_{i=1}^{N} \left(\alpha_i \log(\alpha_i) + (1-\alpha_i)\log(1-\alpha_i)\right) - \frac{\gamma}{2}\sum_{j=1}^{n} z_j \left(\sum_{i=1}^{N} y_i \alpha_i X_{ij}\right)^2 \\ \text{s.t.} \ & \alpha \in [0,1]^N \end{aligned}$$

whereas cuts for the cutting-planes approximation can be added as

$$\beta \geq c(z^\ell) + \nabla c(z^\ell)^T (z - z^\ell),$$

where

$$\frac{\partial c(z)}{z_j} = -\frac{\gamma}{2} z_j \left(\sum_{i=1}^{N} y_i \alpha_i X_{ij}\right)^2.$$

They also show that the left hand side of the objective function in the dual problem can be approximated by a properly defined parabola, which makes the problem quadratic and thus much more efficiently solvable:

$$\alpha \log(\alpha) + (1-\alpha)\log(1-\alpha) \approx \frac{5}{2}\alpha^2 - \frac{5}{2}\alpha - \frac{1}{12}$$

This approximation, seen back in the primal space, is a good quadratic piecewise approximation of the logistic loss which should be more accurate than the piece-wise linear employed by [128].

Finally, we describe a particular approach that is relevant for the rest of the chapter. [128] proposed a mixed integer linear (MILO) reformulation for problem (2.3), which is, to the best of our knowledge, the top performing one, as long as the dimensions of the underlying classification problem are not

exceedingly large (thousand examples and/or hundreds of features). Such approach has two core ideas. The first one consists of the replacement of the ℓ_0 term by the sum of binary indicator variables.

The second key element is the approximation of the nonlinearity in $\mathcal{L}$, i.e., the logistic loss function, by a piecewise linear function, so that the resulting reformulated problem is a MILP problem. The approximating piecewise linear function is defined by the pointwise maximum of a family of tangent lines, that is,

$$\begin{aligned} f(v) = \log(1+\exp(-v)) \approx \hat{f}(v) &= \max\{f'(v^k)(v-v^k) + f(v^k) \mid k = 1, 2, \dots, K\} \\ &= \min\{t \mid t \geq f'(v^k)(v-v^k) + f(v^k),\ k = 1, \dots, K\} \end{aligned}$$

for some discrete set of points $\{v^1, \dots, v^K\}$. The function $\hat{f}$ is a linear underestimator to the true loss logistic function. The final MILP reformulation of problem (2.3) is given by

$$\begin{aligned} \min_{w,z,t}\ & 2\sum_{i=1}^{N} t_i + \lambda \sum_{i=1}^{n} z_i \\ \text{s.t.}\ & -Mz_i \leq w_i \leq Mz_i \quad \forall\, i = 1, \dots, n, \\ & z \in \{0,1\}^n, \\ & t_i \geq f'(v^k)(y^{(i)}(w^\top x^{(i)}) - v^k) + f(v^k)\ \ \forall\, k = 1, \dots, K,\ \ \forall\, i = 1, \dots, N, \end{aligned} \tag{2.12}$$

where M is a large enough positive constant.

The choice of the tangent lines is clearly crucial for this method. For large values of K, problem (2.12) becomes hard to solve. On the other hand, if the number of lines is small, the quality of the approximation will reasonably be low. Hence, points v^k should be selected carefully. [128] suggest to adopt a greedy algorithm that adds one tangent line at a time, minimizing the area of gap between the exact logistic loss and the linear piece-wise approximation. In their work, [128] show that the greedy algorithm provides, depending on the desired set size, the following sets of interpolation points:

$$V_1 = \{0, \pm 1.9, \pm\infty\}, \qquad V_2 = V_1 \cup \{\pm 0.89, \pm 3.55\}, \qquad V_3 = V_2 \cup \{\pm 0.44, \pm 1.37, \pm 2.63, \pm 5.16\}$$

As problem (2.12) employs an approximation of $\mathcal{L}$, the optimal solution $\hat{w}$ obtained by solving it is not necessarily optimal for (2.3). However, since the objective of (2.12) is an underestimator of the original objective function, it is possible to make a posteriori accuracy evaluations. In particular, letting

$w^\star$ be the optimal solution and

$$\hat{\mathcal{L}}(w) = 2\sum_{i=1}^{N} \max_k f'(v^k)(y^{(i)}(w^\top x^{(i)}) - v^k) + f(v^k),$$

we have

$$\hat{\mathcal{L}}(\hat{w}) + \lambda\|\hat{w}\|_0 \leq \mathcal{L}(w^\star) + \|w^\star\|_0 \leq \mathcal{L}(\hat{w}) + \lambda\|\hat{w}\|_0.$$

Hence, if $\mathcal{L}(\hat{w}) - \hat{\mathcal{L}}(\hat{w})$ is small, it is guaranteed that the value of the real objective function at $\hat{w}$ is close to the optimum.

2.4 Proposed method

The MILO approach from [128] is computationally very effective, but it suffers from a main drawback: it scales pretty badly as either the number of examples or the number of features in the dataset grows. This fact is also highlighted by the experimental results reported in the original MILO paper.

On the other hand, heuristic enumerative-like approaches present the limitation of performing moves with a limited horizon. This holds not only for the simple stepwise procedures, but also for other possible more complex and structured strategies that one may come up with. Indeed, selecting one move among all those involving the addition or removal from the current best subset of multiple variables at one time is unsustainable except for tiny datasets.

In this work, we describe a new approach that somehow employs the MILO formulation [128] to overcome the limitations of discrete enumeration methods, but also has better scalability features than the standard MILO approach itself, in particular w.r.t. the number of features. The core idea of our proposal consists of the application of a decomposition strategy to problem (2.3). The classical *Block Coordinate Descent* (BCD) [15,140] algorithm consists in performing, at each iteration, the optimization w.r.t. one block of variables, i.e., the iterations have the form

$$w_{B_\ell}^{\ell+1} \in \arg\min_{w_{B_\ell}} \mathcal{L}(w_{B_\ell}; w_{\bar{B}_\ell}^{\ell}), \tag{2.13}$$

$$w_{\bar{B}_\ell}^{\ell+1} = w_{\bar{B}_\ell}^{\ell}, \tag{2.14}$$

where $B_\ell \subset \{1,\ldots,n\}$ is referred to as *working set*, $\bar{B}_\ell = \{1,\ldots,n\} \setminus B_\ell$. Now, if the working set size $|B|$ is reasonably small, the subproblems can

be easily handled by means of a MILO model analogous to that from [128]. Carrying out such a strategy, the subproblems to be solved at each iteration have the form

$$\begin{aligned}
\min_{w_{B_\ell}, z, t} \; & 2\sum_{i=1}^{N} t_i + \lambda \sum_{i \in B_\ell} z_i \\
\text{s.t.} \; & -M z_i \leq w_i \leq M z_i \quad \forall\, i \in B_\ell, \\
& z_i \in \{0,1\} \quad \forall\, i \in B_\ell, \\
& t_i \geq f'(v^k)(y_i(w^\top x_i) - v^k) + f(v^k) \;\; \forall\, k = 1, \ldots, K, \;\; \forall\, i = 1, \ldots, N. \\
& w_{\bar{B}_\ell} = w^\ell_{\bar{B}_\ell}
\end{aligned} \tag{2.15}$$

At the end of each iteration, we can also introduce a minimization step of $\mathcal{L}$ w.r.t. the current nonzero variables. Since this is a convex minimization step, it allows to refine every iterate up to global optimality w.r.t. the support and to Lu-Zhang stationarity, i.e., local optimality, in terms of the original problem. This operation has low computational cost and a great practical utility, since it guarantees, as we will show in the following, finite termination of the algorithm.

2.4.1 The working set selection rule

Many different strategies could be designed for selecting, at each iteration ℓ, the variables constituting the working set B_ℓ, within the BCD framework. In this work, we propose a rule based on the violation of CW-optimality.

Given the current iterate x^ℓ, we can define a score function

$$p(w^\ell, i) = \begin{cases} \mathcal{L}(0, w^\ell_{j \neq i}) - \lambda + \lambda \|w^\ell\|_0 & \text{if } w^\ell_i \neq 0, \\ \min_{w_i} \mathcal{L}(w_i, w^\ell_{j \neq i}) + \lambda + \lambda \|w^\ell\|_0 & \text{if } w^\ell_i = 0. \end{cases} \tag{2.16}$$

The rational of this score is to estimate what the objective function would become if we forced the considered variable w_i alone to change its status, entering/leaving the support.

We finally select the working set B^ℓ, of size b, choosing, in a greedy way, the b lowest scoring variables, i.e., by solving the problem

$$\begin{aligned}
B^\ell \in \; & \arg\min_{B} \sum_{h \in B} p(w^\ell, h) \\
\text{s.t.} \; & B \subseteq \{1, \ldots, n\}, \\
& |B| = b.
\end{aligned} \tag{2.17}$$

2.4.2 The complete procedure

The whole proposed algorithm is formally summarized in Algorithm 3. Basically, it is a BCD where subproblems are (approximately) solved by the MILO reformulation and variables are selected by (2.17).

In addition, there are some technical steps aimed at making the algorithm work from both the theoretical and the practical point of view.

In the ideal case where the subproblems are solved exactly, thanks to our selection rule, we would be guaranteed to do at least as well as a greedy descent step along a single variable. However, subproblems are approximated and it happens that, solving the MILO, the true objective may sometimes not be decreased, even if the simple greedy step would. In such cases, we actually perform the greedy step to produce the next iterate.

Moreover, at the end of each iteration we perform the refinement step previously discussed. Note that this step cannot increase the value of $\mathcal{L}$, as we are lowering the value of $\mathcal{L}$ by only moving nonzero variables.

Last, we make the stopping criterion explicit; the algorithm stops as soon as an iteration is not able to produce a decrease in the objective value; we then return the point w^ℓ.

2.4.3 Theoretical analysis

In this section, we provide a theoretical characterization for Algorithm 3.

We begin by stating a nice property of the set of local minima of problem (2.3).

Lemma 1. Let $\Gamma = \{\mathcal{L}(w) \mid w \text{ is a local minimum point of (2.3)}\}$. Then $|\Gamma| \leq 2^n$.

Proof. For each support set $S \subseteq \{1, \dots, n\}$ let $L_S^\star$ be the optimal value of the problem

$$\min_{w: w_{\bar{S}}=0} \mathcal{L}(w).$$

Let $w^\star$ be a local minimizer for problem (2.3). Then, from Lu-Zhang conditions and the convexity of $\mathcal{L}$, it is a global minimizer of

$$\min_{w: w_{\bar{S}(w^\star)}=0} \mathcal{L}(w),$$

Algorithm 3: MILO-BCD

Input: $w^0 = 0$, $b < n$.
for $\ell = 0, 1, \ldots$ **do**
  Select the working set B^ℓ using rule (2.17)
  Compute $\nu_{B^\ell}^{\ell+1}$ by solving problem (2.15).
  Set $\nu_{\bar{B}^\ell}^{\ell+1} = w_{\bar{B}^\ell}^{\ell}$
  if $\mathcal{L}(\nu^{\ell+1}) \geq \mathcal{L}(w^\ell)$ **then**
    Set

$$\begin{aligned} \nu^{\ell+1} \in \arg\min_{w} \ & \mathcal{L}(w) \\ \text{s.t. } & \|w^\ell - w\|_0 \leq 1 \\ & w_{\bar{B}^\ell} = w_{\bar{B}^\ell}^{\ell} \end{aligned}$$

  Set

$$\begin{aligned} w^{\ell+1} \in \arg\min_{w} \ & \mathcal{L}(w) \\ \text{s.t. } & w_i = 0 \text{ for all } i \in \bar{S}(\nu^{\ell+1}) \end{aligned}$$

  if $\mathcal{L}(w^{\ell+1}) = \mathcal{L}(w^\ell)$ **then**
    return w^ℓ

and $\mathcal{L}(w^\star) = L^\star_{S(w^\star)} + \lambda |S(w^\star)|$. We hence have

$$\begin{aligned} \Gamma &= \{L^\star_{S(w^\star)} + \lambda |S(w^\star)| \mid w^\star \text{ is a local minimizer for (2.3)}\} \\ &\subseteq \{L^\star_S + \lambda |S| \mid S \subseteq \{1, \ldots, n\}\} \end{aligned}$$

and so

$$|\Gamma| \leq |\{L^\star_S + \lambda |S| \mid S \subseteq \{1, \ldots, n\}\}| \leq |\{S \mid S \subseteq \{1, \ldots, n\}\}| = 2^n.$$

□

We go on with a statement about the relationship between the objective function $\mathcal{L}(w)$ and the score function $p(w, i)$.

Lemma 2. Let p be the score function defined as in (2.16) and let $\bar{w} \in \mathbb{R}^n$. Moreover, for all $h = 1, \ldots, n$, let $\bar{w}^h \in \arg\min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h})$. Then the following statements hold

1. If $\mathcal{L}(\bar{w}^h) = \mathcal{L}(\bar{w})$ then $p(\bar{w}, h) \geq \mathcal{L}(\bar{w})$;
2. If $\mathcal{L}(\bar{w}^h) < \mathcal{L}(\bar{w})$ and $\bar{w}$ satisfies Lu-Zhang conditions, then $p(\bar{w}, h) = \mathcal{L}(\bar{w}^h)$.

Proof. We prove the two statements one at a time:

1. Let us assume that the thesis is false, i.e., $\mathcal{L}(\bar{w}^h) = \mathcal{L}(\bar{w})$ and $p(\bar{w}, h) < \mathcal{L}(\bar{w})$. We distinguish two cases: $\bar{w}_h = 0$ and $\bar{w}_h \neq 0$. In the former case we have

$$\begin{aligned}
\mathcal{L}(\bar{w}) > p(\bar{w}, h) &= \min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda + \lambda \|\bar{w}\|_0 \\
&= \min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda + \lambda \|\bar{w}_{\neq h}\|_0 \\
&\geq \min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda \|w_h\|_0 + \lambda \|\bar{w}_{\neq h}\|_0 \\
&= \min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda \|(w_h, \bar{w}_{\neq h})\|_0 \\
&= \mathcal{L}(\bar{w}^h) = \mathcal{L}(\bar{w}),
\end{aligned}$$

   which is absurd. In the latter case, we have

$$\begin{aligned}
\mathcal{L}(\bar{w}) > p(\bar{w}, h) &= \mathcal{L}(0, \bar{w}_{\neq h}) - \lambda + \lambda \|\bar{w}\|_0 \\
&= \mathcal{L}(0, \bar{w}_{\neq h}) + \lambda \|(0, \bar{w}_{\neq h})\|_0 \\
&\geq \mathcal{L}(\bar{w}^h) = \mathcal{L}(\bar{w}),
\end{aligned}$$

   which is again a contradiction; hence we get the thesis.

2. We again distinguish two cases: $\bar{w}_h = 0$ and $\bar{w}_h \neq 0$. In the first case we have

$$\begin{aligned}
\mathcal{L}(\bar{w}^h) &= \min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) \\
&= \min \left\{ \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}), \mathcal{L}(0, \bar{w}_{\neq h}) \right\} \\
&= \min \left\{ \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}), \mathcal{L}(\bar{w}) \right\}
\end{aligned}$$

   But since we know $\mathcal{L}(\bar{w}^h) < \mathcal{L}(\bar{w})$, we can imply that

$$\min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}) < \mathcal{L}(0, \bar{w}_{\neq h})$$

and we can also write

$$\begin{aligned}\mathcal{L}(\bar{w}^h) &= \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}) \\ &= \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda \|(w_h, \bar{w}_{\neq h})\|_0 \\ &= \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda + \lambda \|\bar{w}_{\neq h}\|_0 \\ &= \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda + \lambda \|\bar{w}\|_0 \\ &= \min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda + \lambda \|\bar{w}\|_0 \\ &= p(\bar{x}, h).\end{aligned}$$

In the second case, since $\bar{w}$ satisfies Lu-Zhang conditions, we have $\bar{w}_h \in \arg\min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h})$. Therefore

$$\bar{w}_h \in \arg\min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}) + \lambda \|(w_h, \bar{w}_{\neq h})\|_0 = \arg\min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h}).$$

Since $\mathcal{L}(\bar{w}^h) < \mathcal{L}(\bar{w}) = \min_{w_h \neq 0} \mathcal{L}(w_h, \bar{w}_{\neq h})$, we get $\bar{w}^h = (0, \bar{w}_{\neq h})$. We finally obtain

$$\begin{aligned}\mathcal{L}(\bar{w}^h) &= \mathcal{L}(\bar{w}^h) + \lambda \|\bar{w}^h\|_0 \\ &= \mathcal{L}(0, \bar{w}_{\neq h}) + \lambda \|(0, \bar{w}_{\neq h})\|_0 \\ &= \mathcal{L}(0, \bar{w}_{\neq h}) + \lambda \|\bar{w}_{\neq h}\|_0 \\ &= \mathcal{L}(0, \bar{w}_{\neq h}) + \lambda \|\bar{w}\|_0 - \lambda \\ &= p(\bar{w}, h).\end{aligned}$$

□

We are finally able to state finite termination and optimality properties of the returned solution of the MILO-BCD procedure.

Proposition 3. Let $\{w^\ell\}$ be the sequence generated by Algorithm 3. Then $\{w^\ell\}$ is a finite sequence and the last element $\bar{w}$ is a CW-minimum for problem (2.3).

Proof. From the instructions of the algorithm, for all $\ell = 1, 2, \ldots$, we have that

$$\begin{aligned}w^\ell \in \arg\min_{w} \ & \mathcal{L}(w) \\ \text{s.t.} \quad & w_i = 0 \text{ for all } i \in \bar{S}(w^\ell),\end{aligned}$$

hence $\nabla_i \mathcal{L}(w^\ell) = 0$ for all $i \in S(w^\ell)$, i.e., w^ℓ satisfies Lu-Zhang conditions and is therefore a local minimum point for problem (2.3). From Lemma 1, we thus know that there exist finite possible values for $\mathcal{L}(w^\ell)$. Moreover, $\{\mathcal{L}(w^\ell)\}$ is a nonincreasing sequence. We can conclude that in a finite number of iterations we get $\mathcal{L}(w^\ell) = \mathcal{L}(w^{\ell+1})$, activating the stopping criterion.

We now prove that the returned point, $\bar{w} = w^{\bar{\ell}}$ for some $\bar{\ell} \in \mathbb{N}$, is CW-optimal. Assume by contradiction that $\bar{w}$ is not CW-optimal. Then, there exists $h \in \{1, \ldots, n\}$ such that $\min_{w_h} \mathcal{L}(w_h, \bar{w}_{\neq h}) < \mathcal{L}(\bar{w})$.

We show that this implies that there exists $t \in \{1, \ldots, n\}$ such that $t \in B^{\bar{\ell}}$ and $\min_{w_t} \mathcal{L}(w_t, \bar{w}_{\neq t}) < \mathcal{L}(\bar{w})$. Assume by contradiction that for all $j \in B^{\bar{\ell}}$ it holds $\min_{w_j} \mathcal{L}(w_j, \bar{w}_{\neq j}) = \mathcal{L}(\bar{w})$. Letting i any index in the working set $B^{\bar{\ell}}$ and recalling Lemma 2, we have

$$\begin{aligned}
\sum_{j \in B^{\bar{\ell}}} p(w^{\bar{\ell}}, j) &= \sum_{j \in B^{\bar{\ell}} \setminus \{i\}} p(w^{\bar{\ell}}, j) + p(w^{\bar{\ell}}, i) \\
&\geq \sum_{j \in B^{\bar{\ell}} \setminus \{i\}} p(w^{\bar{\ell}}, j) + \mathcal{L}(w^{\bar{\ell}}) \\
&> \sum_{j \in B^{\bar{\ell}} \setminus \{i\}} p(w^{\bar{\ell}}, j) + p(w^{\bar{\ell}}, h) \\
&= \sum_{j \in B^{\bar{\ell}} \cup \{h\} \setminus \{i\}} p(w^{\bar{\ell}}, j),
\end{aligned}$$

which contradicts the working set selection rule (2.17).

Now, either $\mathcal{L}(\nu^{\ell+1}) < \mathcal{L}(w^{\bar{\ell}})$ after steps 4-5 of the algorithm, or, after step 7, we get

$$\mathcal{L}(\nu^{\ell+1}) \leq \min_{w_t} \mathcal{L}(w_t, w^{\bar{\ell}}_{\neq t}) < \mathcal{L}(w^{\bar{\ell}}).$$

Therefore, since step 8 cannot increase the value of $\mathcal{L}$, we get $\mathcal{L}(w^{\bar{\ell}+1}) < \mathcal{F}^{\bar{\ell}}$, but this contradicts the fact that the stopping criterion at line 9 is satisfied at iteration $\bar{\ell}$. □

2.4.4 Finding good CW-optima

We have shown in the previous section that Algorithm 3 always returns a CW-optimal solution. Although this allows us to cut off a lot of local minima, there are in practice many low-quality CW-minima. For this reason, we

introduce in our algorithm an heuristic aimed at leaving bad CW-optima where it may get stuck.

In detail, we do as follows. Instead of stopping the algorithm as soon as the objective value does not decrease, we try to repeat the iteration with a different working set. In doing this, we obviously have to change the working set selection rule. This operation is repeated up to a maximum number of times. If after testing a suitable amount of different working sets a decrease in the objective function cannot be achieved, the algorithm stops.

Specifically, we define a modified score function

$$\hat{p}(w^\ell, i) = p(w^\ell, i) + 2^{r_i} - 1, \tag{2.18}$$

where r_i is the number of times the i-th variable was in the working set in the previous attempts.

The idea of this working set selection rule is to first try a greedy selection. Then, if that first attempt failed, we penalize (exponentially) variables that were tried more times and could not provide improvements in the end. This penalty is heuristic. In fact, we may end up with repeating the search over the same working set from the same starting point. However, we can keep track of the working set used throughout the outer iteration, in order to avoid duplicate computations.

Note that such a modification does not alter the theoretical properties of the algorithm; on the other hand, it has a massive impact on the empirical performance.

2.5 Experiments

This section is dedicated to a computational comparison between the approach proposed in this chapter and the state-of-the-art algorithms described in Section 2.3. In our experiments we took into account 11 datasets for binary classification tasks, listed in Table 2.1, from the UCI Machine Learning Repository [42]. In fact, the `digits` dataset is inherently for multi-class classification; we followed the same binarization strategy as [128], assigning a positive label to the examples from the largest class and a negative one to all the others. Moreover, we removed data points with missing variables, encoded the categorical variables with a one-hot vector and normalized the other ones to zero mean and unit variance. In Table 2.1 we also reported

the number n of data points and the number p of features of each dataset, after the aforementioned preprocessing operations.

Dataset	n	p	**Abbreviation**
Parkinsons	195	22	`parkinsons`
Heart (Statlog)	270	25	`heart`
Breast Cancer Wisconsin (Prognostic)	194	33	`breast`
QSAR Biodegradation	1055	41	`biodeg`
SPECTF Heart	267	44	`spectf`
Spambase	4601	57	`spam`
Optical Recognition of Handwritten Digits	3823	62	`digits`
Libras Movement	360	90	`libras`
A2A	2265	123	`a2a`
W2A	2470	300	`w2a`
Madelon	2000	500	`madelon`

Table 2.1: List of datasets used for the experiments on best subset selection in logistic regression.

These datasets constitute a benchmark to evaluate the performance of the algorithms under examination, namely: Forward Selection and Backward Elimination Stepwise heuristics, LASSO, Penalty Decomposition, Concave approximation, the Outer Approximation method in its original form, in the adapted version for cardinality-penalized problems and also in the variant exploiting the approximated dual problems, MILO and our proposed method MILO-BCD. All of these algorithms are described in Section 2.3.

All the experiments described in this section were performed on a machine with Ubuntu Server 18.04 LTS OS, Intel Xeon E5-2430 v2 @ 2.50GHz CPU and 16GB RAM. The algorithms were implemented in Python 3.7.4, exploiting Gurobi 9.0.0 [58] for the outer approximation method, MILO and MILO-BCD. The `scipy` [141] implementation of the L-BFGS algorithm defined in [91] was employed for local optimization steps of all methods. A time limit of 10000 seconds was set for each method.

Both the stepwise methods (forward and backward) exploit L-BFGS [91] as internal optimizer. The forward selection version uses L-BFGS to optimize the logistic with respect to one variable, whereas backward elimination defines his starting point exploiting L-BFGS to optimize the model w.r.t. all the variables.

Concerning LASSO, we solved Problem (2.7) using the `scikit-learn`

implementation [117], with LIBLINEAR library [46] as internal optimizer, for each value of the hyperparameter λ. Each λ value was chosen so that two different hyperparameters, $\lambda_1 \neq \lambda_2$, would not produce the same level of sparsity and to avoid the zero solution. More specifically, we defined our set of hyperparameters by computing the LASSO path, exploiting to the `scikit-learn` function `l1_min_c`. All the obtained solutions were refined by further optimizing w.r.t. the nonzero components only by means of L-BFGS. At the end of this grid search we selected the solution, among these one, providing the best information criterion value.

Penalty Decomposition requires to set a large number of hyperparameters: in our experiments we set $\varepsilon = 10^{-1}$, $\eta = 10^{-3}$ and $\sigma_\varepsilon = 1$ for all the datasets. We ran the algorithm multiple times for values of τ and σ_τ taken from a small grid. L-BFGS was again used as internal solver. The best solution obtained, in terms of information criterion, was retained at the end of the process.

Concave approximation, theoretically, requires the solution of a sequence of problem. However, as outlined in Section 2.3, a single problem with fixed approximation hyperparameter μ can be solved in practice [124]. In our experiments, Problem (2.10) was solved by using L-BFGS. Again, we retain as optimal solution the one that, after an L-BFGS refining step w.r.t. the nonzero variables, minimizes the information criterion among a set of resulting points obtained for different values of μ.

It is important to highlight that the refining optimization step is crucial for methods like the Concave Approximation or LASSO; as a matter of fact, without this precaution, the computed solutions don't even necessarily satisfy the Lu-Zhang conditions.

All variants of the Outer Approximation method expoit Gurobi to handle the MILP subproblems and L-BFGS for the continuous ones. As suggested by [17], a single branch and bound tree is constructed to solve all the MILP subproblems, adding cutting-type constraints dynamically as lazy constraints. Moreover, the starting cut is decided by means of the first-order heuristic described in the referenced work. For the cardinality-constrained version of the algorithm, we set a time limit of 1000 seconds for the solution of any individual problem of the form (2.11) with a fixed value of s. As for the dual formulation, we set $\gamma = 10^4$ to make the considered problem as close as possible to the formulation tackled by all other algorithms. The approximated version of the dual problem, which is quadratic, is efficiently

Dataset	Method	AIC	ℓ_0	Time (s)
	Forward Stepwise	129.2567	5	0.280
	Backward Stepwise	126.6948	17	0.172
	LASSO	129.7412	16	6.290
	Penalty Decomposition	134.1499	2	22.486
	Concave Approximation	129.6589	17	2.899
`parkinsons`	Outer Approximation CC	115.8998	9	$\geq$ 10000
	Outer Approximation CP	120.6478	11	$\geq$ 10000
	Outer Approximation Dual	128.1812	17	3.278
	MILO	113.5371	8	12.531
	MILO-BCD	**113.5005**	8	93.708
	Forward Stepwise	197.6972	11	0.577
	Backward Stepwise	216.6682	23	0.038
	LASSO	202.4335	15	2.282
	Penalty Decomposition	226.1013	4	49.500
	Concave Approximation	206.4321	17	1.384
`heart`	Outer Approximation CC	206.8911	12	$\geq$ 10000
	Outer Approximation CP	263.2117	5	$\geq$ 10000
	Outer Approximation Dual	207.2493	19	4.087
	MILO	195.7757	11	41.593
	MILO-BCD	**195.6242**	11	95.399
	Forward Stepwise	180.4932	6	0.470
	Backward Stepwise	163.2610	33	0.413
	LASSO	156.6797	24	21.321
	Penalty Decomposition	189.2942	2	8.044
	Concave Approximation	158.11729	24	3.398
`breast`	Outer Approximation CC	166.1055	34	$\geq$ 10000
	Outer Approximation CP	202.8904	9	$\geq$ 10000
	Outer Approximation Dual	161.6405	31	6.675
	MILO	**147.5119**	19	86.250
	MILO-BCD	147.6781	17	236.126
	Forward Stepwise	703.9588	20	3.582
	Backward Stepwise	661.6047	32	0.417
	LASSO	665.1640	32	65.344
	Penalty Decomposition	671.8854	18	232.120
	Concave Approximation	663.5171	24	5.789
`biodeg`	Outer Approximation CC	678.4316	42	$\geq$ 10000
	Outer Approximation CP	1263.0706	6	$\geq$ 10000
	Outer Approximation Dual	681.6687	31	29.329
	MILO	**653.4768**	23	6885.277
	MILO-BCD	654.4053	25	707.356
	Forward Stepwise	178.9840	6	0.797
	Backward Stepwise	180.0595	28	0.214
	LASSO	181.4678	13	8.966
	Penalty Decomposition	222.8672	2	55.287
	Concave Approximation	181.8271	17	3.788
`spectf`	Outer Approximation CC	178.8349	12	$\geq$ 10000
	Outer Approximation CP	222.3555	5	$\geq$ 10000
	Outer Approximation Dual	206.1484	38	10.766
	MILO	168.5162	15	1293.650
	MILO-BCD	**168.3443**	15	205.6255
	Forward Stepwise	53.3215	11	2.558
	Backward Stepwise	152.0723	76	0.470
	LASSO	28.0720	14	5.413
	Penalty Decomposition	142.4580	2	530.951
	Concave Approximation	70.0072	35	12.532
`libras`	Outer Approximation CC	33.4904	11	$\geq$ 10000
	Outer Approximation CP	72.4350	6	$\geq$ 10000
	Outer Approximation Dual	64.0018	32	58.061
	MILO	14.2040	7	$\geq$ 10000
	MILO-BCD	**14.1557**	7	654.227

Table 2.2: Results of AIC minimization in logistic regression with different optimization methods on small datasets.

Dataset	Method	AIC	ℓ_0	Time (s)
	Forward Stepwise	1906.5143	45	36.136
	Backward Stepwise	1901.9650	46	1.468
	LASSO	**1892.6580**	53	1209.108
	Penalty Decomposition	5244.4292	3	$\geq$ 10000
	Concave Approximation	1916.1159	51	13.654
`spam`	Outer Approximation CC	1963.0467	52	$\geq$ 10000
	Outer Approximation CP	2138.2306	36	$\geq$ 10000
	Outer Approximation Dual	1931.7670	58	161.062
	MILO	1909.0709	44	$\geq$ 10000
	MILO-BCD	1904.2989	44	8442.004
	Forward Stepwise	378.6893	25	13.139
	Backward Stepwise	341.8344	42	1.894
	LASSO	346.9967	43	2154.283
	Penalty Decomposition	7168.3316	1	$\geq$ 10000
	Concave Approximation	338.1436	31	24.398
`digits`	Outer Approximation CC	386.4583	64	$\geq$ 10000
	Outer Approximation CP	686.1014	12	$\geq$ 10000
	Outer Approximation Dual	372.3541	44	125.282
	MILO	323.6231	26	$\geq$ 10000
	MILO-BCD	**322.7531**	25	6557.441
	Forward Stepwise	1605.9851	34	20.864
	Backward Stepwise	1659.0279	87	4.038
	LASSO	1615.6245	60	394.008
	Penalty Decomposition	1676.8714	16	605.042
	Concave Approximation	1647.3086	84	17.422
`a2a`	Outer Approximation CC	1710.0609	120	$\geq$ 10000
	Outer Approximation CP	2581.0224	2	$\geq$ 10000
	Outer Approximation Dual	1663.0632	53	$\geq$ 10000
	MILO	1607.3254	52	$\geq$ 10000
	MILO-BCD	**1589.5884**	37	8553.430
	Forward Stepwise	395.0422	51	283.327
	Backward Stepwise	479.2162	169	137.361
	LASSO	721.0487	294	$\geq$ 10000
	Penalty Decomposition	1973.6854	1	$\geq$ 10000
	Concave Approximation	534.2647	166	144.470
`w2a`	Outer Approximation CC	760.3417	301	$\geq$ 10000
	Outer Approximation CP	833.2059	7	$\geq$ 10000
	Outer Approximation Dual	722.9003	294	207.279
	MILO	358.5662	82	$\geq$ 10000
	MILO-BCD	**339.7765**	55	$\geq$ 10000
	Forward Stepwise	2506.9165	91	461.957
	Backward Stepwise	2528.5802	156	431.609
	LASSO	2523.0742	103	1795.424
	Penalty Decomposition	2638.5021	4	833.624
	Concave Approximation	2769.9642	340	47.042
`madelon`	Outer Approximation CC	2652.4555	9	$\geq$ 10000
	Outer Approximation CP	2765.2852	2	$\geq$ 10000
	Outer Approximation Dual	2657.4810	15	$\geq$ 10000
	MILO	2616.5531	16	$\geq$ 10000
	MILO-BCD	**2504.0655**	102	$\geq$ 10000

Table 2.3: Results of AIC minimization in logistic regression with different optimization methods on large datasets.

solved with Gurobi instead of L-BFGS.

As concerns MILO and MILO-BCD, we employed the V_2 set of interpolation points for both methods, in order to have a good trade-off between accuracy and computational burden. Moreover, for MILO-BCD we set the cardinality of the working set b to 20 for all the problems. We report in Section 2.5 the results of preliminar computational experiments that appear to support our choice. All the subproblems were solved with Gurobi. For MILO-BCD we employ the heuristic discussed in Section 2.4.4. For each problem, the maximum number of consecutive attempts with no improvement, before stopping the algorithm, is set to n. Note that, in order to improve the algorithm efficiency, we instantiate a single MILP problem with n variables and dynamically change the box constraints based on the current working set. The continuous optimization steps needed to perform steps 7 and 8 of Algorithm 3 are performed by using L-BFGS.

In Tables 2.2, 2.3, 2.4 and 2.5 the computational results of minimizing AIC and BIC respectively on the 11 datasets are shown. For each algorithm and problem, we can see the information criterion value at the returned solution, its zero norm and the total runtime. We can observe the effectiveness of the MILO-BCD approach w.r.t. the other methods. In particular in 8 out of 11 test problems MILO-BCD found the best AIC value, while in the remaining three cases it attains a very close second-best result. The results of minimizing BIC are very similar: for 9 out of 11 datasets MILO-BCD returns the best solution and in the remaining two it ranks at the second place. We can also note that, in cases where p is large such as `spam`, `digits`, `a2a`, `w2a` and `madelon` datasets, our method, within the established time limit, is able to find a much better quality solution with respect to the other algorithms (with the only exception of `spam` for the AIC), and in particular compared to MILO.

As for the efficiency, Tables 2.2, 2.3, 2.4 and 2.5 also allow to evaluate the computational burden of MILO-BCD. As expected, our method is slower than the approaches that are not based on Mixed Integer Optimization, which on the other hand provide lower quality solutions. However, compared to standard MILO, we can see a considerable improvement in terms of computational time with both the small and the large datasets.

In Figure 2.1 we plot the cumulative distribution of absolute distance from the optimum attained by each solver, computed upon the 22 subset selection problems. The x-axis values represent the difference in absolute

Dataset	Method	BIC	ℓ_0	Time (s)
	Forward Stepwise	142.4486	3	0.198
	Backward Stepwise	166.7417	12	0.165
	LASSO	140.6959	2	6.391
	Penalty Decomposition	277.1788	1	25.056
	Concave Approximation	147.2337	4	2.922
`parkinsons`	Outer Approximation CC	139.1962	6	$\geq$ 10000
	Outer Approximation CP	139.1962	6	5533.104
	Outer Approximation Dual	145.6313	3	3.412
	MILO	137.6446	6	16.276
	MILO-BCD	**137.6011**	6	93.572
	Forward Stepwise	225.7059	5	0.337
	Backward Stepwise	279.0788	19	0.065
	LASSO	227.2449	5	2.350
	Penalty Decomposition	317.3171	1	61.155
	Concave Approximation	251.8435	12	2.156
`heart`	Outer Approximation CC	236.8324	3	$\geq$ 10000
	Outer Approximation CP	288.0285	5	$\geq$ 10000
	Outer Approximation Dual	234.3866	7	4.045
	MILO	223.7984	6	22.865
	MILO-BCD	**223.6797**	6	116.618
	Forward Stepwise	195.8299	2	0.216
	Backward Stepwise	265.9623	32	0.354
	LASSO	195.8299	2	21.225
	Penalty Decomposition	195.8299	2	7.972
	Concave Approximation	203.5140	6	3.201
`breast`	Outer Approximation CC	194.2193	4	$\geq$ 10000
	Outer Approximation CP	231.7141	8	$\geq$ 10000
	Outer Approximation Dual	195.8300	2	6.664
	MILO	**192.4211**	10	653.077
	MILO-BCD	193.5695	5	124.760
	Forward Stepwise	782.2522	13	2.560
	Backward Stepwise	792.7384	26	0.400
	LASSO	785.1660	22	65.498
	Penalty Decomposition	950.2008	4	420.761
	Concave Approximation	772.6349	22	6.917
`biodeg`	Outer Approximation CC	880.5540	12	$\geq$ 10000
	Outer Approximation CP	1154.3736	5	$\geq$ 10000
	Outer Approximation Dual	835.4689	31	29.016
	MILO	746.8531	14	$\geq$ 10000
	MILO-BCD	**745.1778**	13	2305.093
	Forward Stepwise	203.9442	4	0.573
	Backward Stepwise	237.7547	17	0.252
	LASSO	208.8296	7	8.949
	Penalty Decomposition	277.1788	1	25.056
	Concave Approximation	228.256224	12	3.867
`spectf`	Outer Approximation CC	214.4389	3	$\geq$ 10000
	Outer Approximation CP	224.2627	5	$\geq$ 10000
	Outer Approximation Dual	251.6325	7	10.649
	MILO	196.8356	5	231.938
	MILO-BCD	**196.8238**	5	115.597
	Forward Stepwise	101.5028	4	1.145
	Backward Stepwise	270.8327	46	0.970
	LASSO	71.4674	9	5.453
	Penalty Decomposition	182.2357	1	42.429
	Concave Approximation	131.7340	17	14.591
`libras`	Outer Approximation CC	75.3065	9	$\geq$ 10000
	Outer Approximation CP	153.6910	5	$\geq$ 10000
	Outer Approximation Dual	132.1737	10	58.047
	MILO	**41.3979**	7	$\geq$ 10000
	MILO-BCD	53.0895	9	642.618

Table 2.4: Results of BIC minimization in logistic regression with different optimization methods on small datasets.

Dataset	Method	BIC	ℓ_0	Time (s)
	Forward Stepwise	2361.1337	27	28.446
	Backward Stepwise	2140.7302	32	2.0124
	LASSO	2177.5219	40	1214.969
	Penalty Decomposition	6184.8715	1	$\geq$ 10000
	Concave Approximation	2196.5090	38	16.464
`spam`	Outer Approximation CC	2336.2203	58	$\geq$ 10000
	Outer Approximation CP	3894.7351	10	$\geq$ 10000
	Outer Approximation Dual	2275.2687	51	157.805
	MILO	2150.2450	30	$\geq$ 10000
	MILO-BCD	**2137.9834**	31	$\geq$ 10000
	Forward Stepwise	552.1658	13	8.110
	Backward Stepwise	468.9395	20	2.615
	LASSO	529.0165	24	2160.987
	Penalty Decomposition	5299.8033	0	$\geq$ 10000
	Concave Approximation	516.442699	28	32.288
`digits`	Outer Approximation CC	640.2697	10	$\geq$ 10000
	Outer Approximation CP	1696.2871	5	$\geq$ 10000
	Outer Approximation Dual	596.1621	28	128.011
	MILO	448.3050	14	$\geq$ 10000
	MILO-BCD	**441.0145**	15	9949.433
	Forward Stepwise	1741.3958	15	10.727
	Backward Stepwise	2016.2528	64	5.798
	LASSO	1764.5871	15	397.503
	Penalty Decomposition	1860.2444	5	607.869
	Concave Approximation	1873.3706	44	21.709
`a2a`	Outer Approximation CC	2028.2982	11	$\geq$ 10000
	Outer Approximation CP	2268.7472	4	$\geq$ 10000
	Outer Approximation Dual	1829.5696	14	$\geq$ 10000
	MILO	1754.9999	16	$\geq$ 10000
	MILO-BCD	**1733.8513**	17	2933.3452
	Forward Stepwise	614.3182	18	107.705
	Backward Stepwise	1320.8765	143	147.459
	LASSO	2524.0133	293	$\geq$ 10000
	Penalty Decomposition	1979.8373	1	$\geq$ 10000
	Concave Approximation	919.0693	70	166.238
`w2a`	Outer Approximation CC	879.0359	2	$\geq$ 10000
	Outer Approximation CP	931.5931	3	$\geq$ 10000
	Outer Approximation Dual	2531.5618	294	205.223
	MILO	671.9868	20	$\geq$ 10000
	MILO-BCD	**579.0229**	26	8842.6791
	Forward Stepwise	**2660.6283**	3	24.179
	Backward Stepwise	2732.3224	15	488.801
	LASSO	2661.9344	6	1852.270
	Penalty Decomposition	2772.5887	0	75.713
	Concave Approximation	3030.0118	86	152.799
`madelon`	Outer Approximation CC	2677.8156	4	$\geq$ 10000
	Outer Approximation CP	2781.6611	2	$\geq$ 10000
	Outer Approximation Dual	2689.3907	2	$\geq$ 10000
	MILO	2681.9310	1	$\geq$ 10000
	MILO-BCD	**2660.6283**	3	$\geq$ 10000

Table 2.5: Results of BIC minimization in logistic regression with different optimization methods on large datasets.

value between the information criterion obtained and the best one found, while y-axis reports the fraction of solved problems within a certain distance from the best. As it is possible to see, MILO-BCD clearly outperforms the other methods. As a matter of fact, MILO-BCD always found a solution that is distant less than 15 from the optimal one and in around the 80% of the problems it attained the optimal solution. We can also see that for all the other methods there is a number of bad cases where the obtained value is very far from the optimal one. Note that we consider the absolute distance from the best, instead of a relative distance, since it is usually the difference in IC values which is considered in practice to assess the quality of a model w.r.t. another one [25].

Finally, we highlight that MILO-BCD manages to greatly increase the performance of MILO, without making its interface more complex. As a matter of fact, we have only added a hyperparameter that controls the cardinality of the working set and experimentally appears to be extremely easy to tune. Indeed, note that all the experiments were carried out using the same working set size for each dataset and, despite this choice, MILO-BCD shown impressive performances in all the considered datasets.

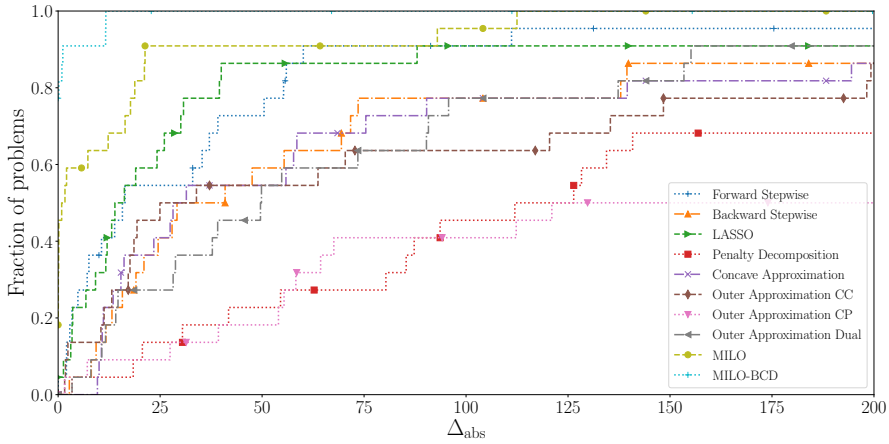


Figure 2.1: Each curve represents the fraction of the 22 classification problems for which the corresponding solver obtains an absolute error less or equal than Δ_{abs} w.r.t. the optimal value.

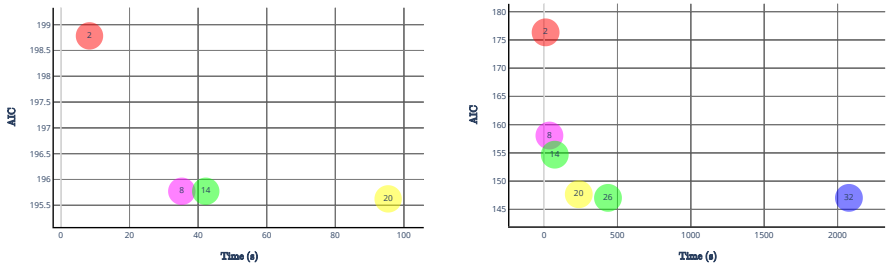
Varying the working set size

The value of the working set size b may greatly affect the performance of the MILO-BCD procedure, in terms of both quality of solutions and running time. For this reason, we performed a study to evaluate the behavior of the algorithm as the value of b changes. We ran MILO-BCD on the problems obtained from datasets at different scales: `heart`, `breast`, `spectf`, and `a2a`. AIC is used as GOF measure.

Dataset	b	AIC	ℓ_0	Time (s)
`heart`	2	198.7826	11	8.326
	8	195.7715	10	35.240
	14	195.7715	10	42.222
	20	195.6242	11	95.399
	26	-	-	-
	32	-	-	-
`breast`	2	176.3391	7	10.079
	8	158.0725	13	36.012
	14	154.6846	17	72.4643
	20	147.6781	17	236.126
	26	147.0381	19	435.3751
	32	147.0381	19	2077.4473
`spectf`	2	171.9253	12	19.333
	8	175.4713	7	43.999
	14	169.4771	18	118.313
	20	168.3443	15	205.6255
	26	168.3443	15	485.486
	32	168.3443	15	1245.422
`a2a`	2	1591.7767	35	430.714
	8	1595.2172	37	1333.7368
	14	1590.7749	35	1984.113
	20	1589.5884	37	8553.430
	26	1586.7499	39	$\geq$ 10000
	32	1592.8824	37	$\geq$ 10000

Table 2.6: Results obtained by the MILO-BCD procedure on the best subset selection problem based on AIC with four datasets for different values of working set size b.

The results are reported in Table 2.6 and Figure 2.2. We can see that a working set size of 20, as employed in the experiments of the previous section, provides a good trade-off. Indeed, the running time seems to grow in general with the working set size, whereas the optimal solution is approached only when large working sets are employed. We can see that in some cases a slightly larger value of b allows to retrieve even better solutions than those obtained in the experiments done, but the computational cost significantly increases. In the end, as can be observed in this Section, the choice $b = 20$ experimentally led to excellent results on the entirety of our benchmark.



(a) `heart` (b) `breast`

(c) `spectf` (d) `a2a`

Figure 2.2: Trade-off between runtime and solution quality for different values of the working set size in MILO-BCD, on the best subset selection problem based on AIC for the four considered problems.

2.6 Final considerations

In this Chapter, we considered the problem of best subset selection in logistic regression, with particular emphasis on the IC-based formulation. We introduced an algorithm combining mixed-integer programming models and decomposition techniques like the block coordinate descent. The aim of the algorithm is to find high quality solutions even on larger scale problems, where other existing MIP-based methods are unreasonably expensive, while heuristic and local-optimization-based methods produce very poor solutions.

We theoretically characterized the features and the behavior of the proposed method. Then, we showed the results of wide computational experiments, proving that the proposed approach indeed is able to find, in a reasonable running time, much better solutions than a set of other state-of-the-art solvers; this fact appears particularly evident on the problems with higher dimension.

Future research will be focused on the definition of possibly more effective and efficient working set selection rules for our algorithm. Upcoming work may also be aimed at adapting the proposed algorithm to deal with different or more general problems.

Chapter 3

Pruning of Fully Connected Layer's Nodes

Training of neural networks can be reformulated in spectral space, by allowing eigenvalues and eigenvectors of the network to act as target of the optimization instead of the individual weights. Working in this setting, we show that the eigenvalues can be used to rank the nodes' importance within the ensemble. Indeed, we will prove that sorting the nodes based on their associated eigenvalues, enables effective pre- and post-processing pruning strategies to yield massively compacted networks (in terms of the number of composing neurons) with virtually unchanged performance. The proposed methods are tested for different architectures, with just a single or multiple hidden layers, and against distinct classification tasks of general interest.

3.1 Preamble

In this Chapter we will discuss a relevant byproduct of the spectral learning scheme [30, 52] (discussed in Section 3.2). More specifically, we will argue that the eigenvalues do provide a reliable ranking of the nodes, in terms of their associated contribution to the overall performance of the trained network. Working along these lines, we will empirically prove that the absolute value of the eigenvalues is an excellent marker of the node's significance in carrying out the assigned discrimination task. This observation can be effectively exploited, downstream of training, to filter the nodes in terms of their relative importance and prune the unessential units so as to yield a more compact model, with almost identical classification abilities. For the sake of completeness, let us emphasize a substantial difference between [30, 52] and the one proposed in this Chapter: in Giambagli et al. [52] and Chicchi et al. [30] the focus is on designing a training algorithm in the spectral domain while, in this work, we propose a novel idea to effectively prune fully connected layers by exploiting the spectral training.

The effectiveness of the proposed method has been tested for different feed-forward architectures, with just a single or multiple hidden layers, by invoking several activation functions, and against distinct datasets for image recognition, with various levels of inherent complexity. Building on these findings, we will also propose a two stages training protocol to generate minimal networks (in terms of allowed computing neurons) which outperform those obtained by hacking off dispensable units from a large, fully trained, apparatus. This strategy can be seen as an effective way to discover subnetworks (a.k.a. "winning tickets" [50]) with recorded performance comparable to those displayed by their unaltered homologous, after a proper round of training [50]. More specifically, after a first round of training which solely acts on the eigenvalues, one can identify the most relevant nodes, as follows the magnitude of the associated eigenvalues. Since the first training stage is just targeted to eigenvalues, the eigenvectors obtained after pruning are still bearing reflexes of the random initialization and thus represent a sort of "winning ticket" [50]. In this respect, according to the above reasoning, the proposed two stages strategy can be seen as a novel and efficient way to discover optimal sub-networks. Finally, both the proposed approaches address the same underlying problem of fitting the best model with an ℓ_0 constraint on the eigenvalues and, as a consequence, on the number neurons.

3.2 Preliminary

In standard neural network training one seeks to optimize the weights that link pairs of neurons belonging to adjacent layers of the selected architecture [55]. This is achieved by computing the gradient of the loss with respect to the sought weights, a procedure which amounts to operate in the so called direct space of the network [52]. Alternatively, the learning can be carried out in reciprocal space: the spectral attributes (eigenvalues and eigenvectors) of the fully-connected layer define the actual target of the optimization.

In the following, we follow mainly the results of [30], where an extension of the method originally introduced in [52] is handed over. In a nutshell, the procedure introduced in [52] and further refined in [30], enables a substantial compression of the space of trainable parameters. The spectral method leverages on a limited subset of key parameters which impact on the whole set of weights in direct space. Particularly relevant, in this respect, is the setting where the eigenmodes of the inter-layer transfer operators align along random directions. In this case, the associated eigenvalues constitute the sole trainable parameters. When employed for classifications tasks, the accuracy displayed by the spectral scheme restricted to operate with eigenvalues is slightly worse than that reported when the learning is carried in direct space, for an identical architecture and by employing the full set of trainable parameters. To bridge the gap between conventional and spectral methods in terms of measured performances, one can also train the elements that populate the non trivial block of the eigenvectors matrix [52]. By resorting to apt decomposition schemes, it is still possible to contain the total number of trainable parameters, while reaching stunning performances in terms of classification outcomes [30].

Consider a deep feed-forward network made of ℓ distinct layers. Each layer is labelled with a discrete index i $(= 1, ..., \ell)$. Denote by N_i the number of the neurons, the individual computing units, that pertain to layer i. Then, we posit $N = \sum_{i=1}^{\ell} N_i$ and introduce a column vector $\vec{x}^{(1)}$, of size N, the first N_1 entries referring to the supplied input signal. As anticipated, we will be mainly concerned with datasets for image recognition, so we will use this specific case to illustrate the more general approach of spectral learning. This means that, the first N_1 elements of $\vec{x}^{(1)}$ are the intensities (from the top-left to the bottom-right, moving horizontally) as displayed on the pixels of the image presented as an input. All other entries of $\vec{x}^{(1)}$ are identically equal to zero.

The aim of the procedure is to map $\vec{x}^{(1)}$ into an output vector $\vec{x}^{(\ell)}$, still of size N: the last N_ℓ elements are the intensities displayed at the output nodes, where reading is eventually performed. The applied transformation is composed by a suite of linear operations, interposed to non linear filters. To exemplify the overall strategy, consider the generic vector $\vec{x}^{(k)}$, with $k = 1, ..., \ell - 1$, as obtained after k execution of the above procedure. At the successive iteration, one gets $\vec{x}^{(k+1)} = \boldsymbol{A}^{(k)}\vec{x}_{(k)}$, where $\boldsymbol{A}^{(k)}$ is a $N \times N$ matrix with a rather specific structure (schematically depicted in Fig. 3.2). Further, a suitably defined non-linear function $f(\cdot, \beta_k)$ is applied to $\vec{x}^{(k+1)}$, where β_k identifies an optional bias. Eq. (3.1) summarizes the chain of operation described above.

To proceed in the analysis, we cast $\boldsymbol{A}^{(k)} = \boldsymbol{\Phi}^{(k)}\boldsymbol{\Lambda}^{(k)}\left(\boldsymbol{\Phi}^{(k)}\right)^{-1}$ by invoking spectral decomposition. Here, $\boldsymbol{\Lambda}^{(k)}$ denotes the diagonal matrix of the eigenvalues of $\boldsymbol{A}^{(k)}$. Following [30], we set $\left(\boldsymbol{\Lambda}^{(k)}\right)_{jj} = 1$ for $j < \sum_{i=1}^{k-1} N_i$ and $j > \sum_{i=1}^{k+1} N_i$. The remaining $N_k + N_{k+1}$ elements are initially assigned to random entries, as e.g. extracted from a uniform distribution, and define a first basin of target variables for the spectral learning scheme. Then, $\boldsymbol{\Phi}^{(k)}$ is the identity matrix $\mathbb{I}_{N\times N}$, with the inclusion of a sub-diagonal $N_{k+1} \times N_k$ block, denoted by $\boldsymbol{\phi}^{(k)}$, see Fig. 3.3. This choice amounts to assume a feed-forward architecture. It can be easily shown that $\left(\boldsymbol{\Phi}^{(k)}\right)^{-1} = 2\mathbb{I}_{N\times N} - \boldsymbol{\Phi}^{(k)}$, which readily yields $\boldsymbol{A}^{(k)} = \boldsymbol{\Phi}^{(k)}\boldsymbol{\Lambda}^{(k)}\left(2\mathbb{I}_{N\times N} - \boldsymbol{\Phi}^{(k)}\right)$. The off-diagonal elements of $\boldsymbol{\Phi}^{(k)}$ define a second set of adjustable parameters to be self-consistently modulated during active training.

To implement the learning scheme on these basis, we consider $\vec{x}^{(\ell)}$, the image on the output layer of the input vector $\vec{x}^{(1)}$ (see Figure 3.1 for a graphical illustration):

$$\vec{x}^{(\ell)} = f\left(\boldsymbol{A}^{(\ell-1)}...f\left(\boldsymbol{A}^{(1)}\vec{x}^{(1)}, \beta_1\right), \beta_{\ell-1}\right) \tag{3.1}$$

Since we are dealing with image classification, we can calculate $\vec{z} = softmax(\vec{x}^{(\ell)})$. We will then use $\vec{z}$ to compute the categorical cross-entropy loss function $\mathrm{CCE}(l(\vec{x}^{(1)}), \vec{z})$, where $l(\vec{x}^{(1)})$ is the label which identifies the category to which $\vec{x}^{(1)}$ belongs, via one-hot encoding [1].

The loss function can thus be minimized by acting on the spectral parameters, i.e. the ensemble made of non trivial eigenvalues and/or the associated eigendirections. Moreover, it is possible to derive a closed analytical expression for $w_{ij}^{(k)}$, the weights of the edges linking nodes i (belonging to layer $k + 1$) and j (sitting on layer k) in direct space, as a function of the

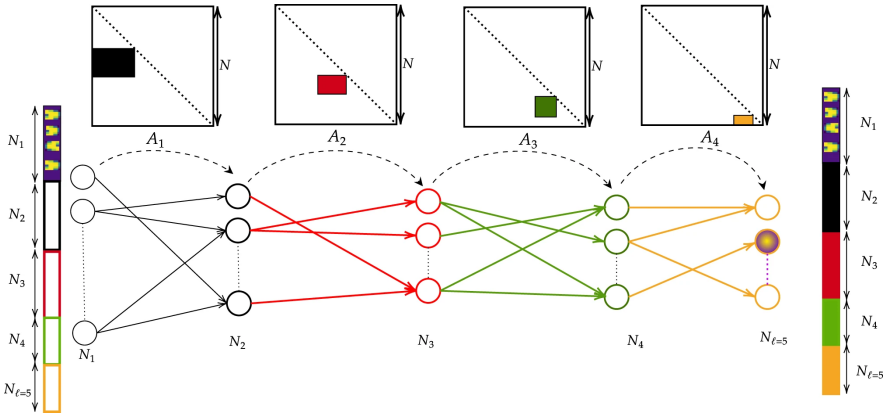


Figure 3.1: A sketch of the spectral learning framework. Image taken from [52].

underlying spectral quantities.

We begin by recalling that $\boldsymbol{A}^{(k)}$ is a $N \times N$ matrix. From $\boldsymbol{A}^{(k)}$ we select a square sub-block of size $(N_k + N_{k+1}) \times (N_k + N_{k+1})$, formed by the elements $\boldsymbol{A}^{(k)}_{i',j'}$ with $i' = \sum_{s=1}^{k-1} N_s + i$ and $j' = \sum_{s=1}^{k-1} N_s + j$, with $i = 1, ..., N_k + N_{k+1}$, $j = 1, ..., N_k + N_{k+1}$. We use $\boldsymbol{A}^{(k)}$ to identify the obtained matrix and proceed in analogy for $\boldsymbol{\Lambda}^{(k)}$ and $\boldsymbol{\Phi}^{(k)}$. Then:

$$\begin{aligned} A^{(k)}_{ij} &= \left[\boldsymbol{\Phi}^{(k)}\boldsymbol{\Lambda}^{(k)}\left(2I - \boldsymbol{\Phi}^{(k)}\right)\right]_{ij} \\ &= \left[2\boldsymbol{\Phi}^{(k)}\boldsymbol{\Lambda}^{(k)}\right]_{ij} - \left[\boldsymbol{\Phi}^{(k)}\boldsymbol{\Lambda}^{(k)}\boldsymbol{\Phi}^{(k)}\right]_{ij} \\ &= \alpha^{(k)}_{ij} - \beta^{(k)}_{ij} \end{aligned} \tag{3.2}$$

From hereon, we will omit the apex (k). Assume $\lambda_1 \dots \lambda_{N_k+N_{k+1}}$ to identify the eigenvalues of the transfer operator $\boldsymbol{A}$, namely the diagonal entries of Λ. Hence, $\Lambda_{ij} = \sum_{j=1}^{N_k+N_{k+1}} \delta_{ij}\lambda_j$.

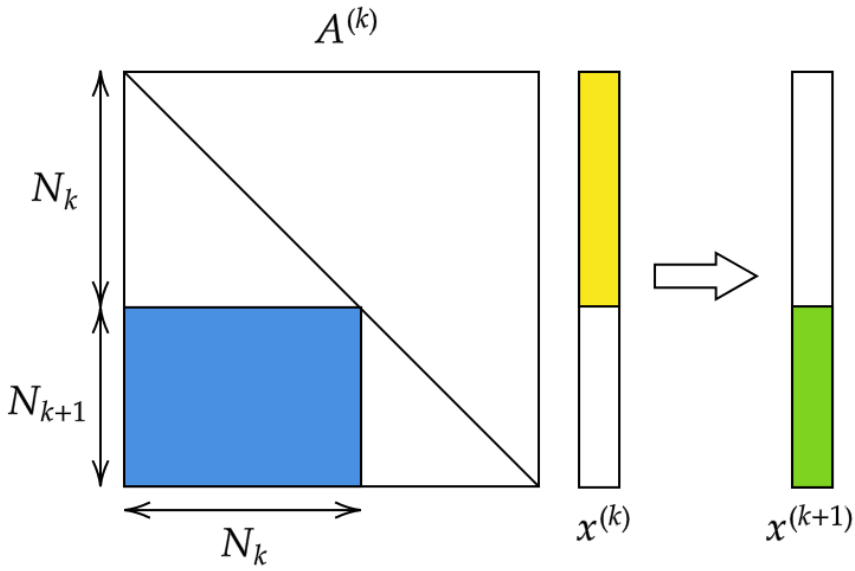


Figure 3.2: A schematic outline of the structure of transfer matrix $\boldsymbol{A}^{(k)}$, bridging layer k to layer $k+1$. The action of $\boldsymbol{A}^{(k)}$ on $\vec{x}^{(k)}$ is also graphically illustrated.

The quantities α_{ij} and β_{ij} read:

$$\begin{aligned}\alpha_{ij} &= 2\sum_{k=1}^{N_k+N_{k+1}} \Phi_{ik}\lambda_k\delta_{kj} = 2\Phi_{ij}\lambda_j \\ \beta_{ij} &= \sum_{k,m=1}^{N_k+N_{k+1}} \Phi_{ik}\lambda_k\delta_{km}\Phi_{mj} \\ &= \sum_{m\in\mathcal{I}\cup\mathcal{J}} \delta_{im}\lambda_m\Phi_{mj}\end{aligned}$$

where $j \in \mathcal{J} = (1, ..., N_k)$ refer to the nodes at the departure layer (k), whereas $i \in \mathcal{I} = (N_k + 1, ..., N_k + N_{k+1})$ stand for those at arrival. Hence, $\mathcal{I} \cup \mathcal{J} = [1, ..., N_k + N_{k+1}]$. The above expression for β_{ij} can be further

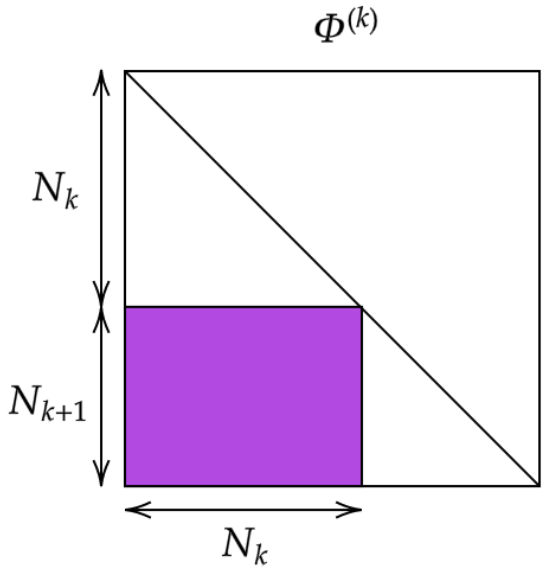


Figure 3.3: The structure of matrix $\mathbf{\Phi}^{(k)}$ is schematically displayed.

manipulated to eventually yield

$$\begin{aligned}\beta_{ij} &= \sum_{m\in J} \Phi_{im}\lambda_m\Phi_{mj} + \sum_{m\in I} \Phi_{im}\lambda_m\Phi_{mj} \\ &= \Phi_{ij}\lambda_j + \lambda_i\Phi_{ij}\end{aligned}$$

and therefore Equation (3.2) as

$$\begin{aligned}\alpha_{ij} - \beta_{ij} &= 2\Phi_{ij}\lambda_j - \Phi_{ij}\lambda_j - \lambda_i\Phi_{ij} \\ &= (\lambda_j - \lambda_i)\phi_{ij}\end{aligned} \tag{3.3}$$

Therefore, rewriting everything in a more convenient notation, we obtain:

$$A_{ij}^{(k)} = w_{ij}^{(k)} = \left(\lambda_{m(j)}^{(k)} - \lambda_{l(i)}^{(k)}\right) \Phi_{l(i),m(j)}^{(k)} \tag{3.4}$$

where $l(i) = \sum_{s=1}^{k} N_s + i$ and $m(j) = \sum_{s=1}^{k-1} N_s + j$, with $i \in (1, ..., N_{k+1})$ and $j \in (1, ..., N_k)$. In the above expression, $\lambda_{m(j)}^{(k)}$ stand for the first N_k eigenvalues of $\mathbf{\Lambda}^{(k)}$. The remaining N_{k+1} eigenvalues are labelled $\lambda_{l(i)}^{(k)}$. In

the above expression, $\lambda^{(k)}_{m(j)}$ stand for the first N_k eigenvalues of $\mathbf{\Lambda}^{(k)}$. The remaining N_{k+1} eigenvalues are labelled $\lambda^{(k)}_{l(i)}$.

To help comprehension denote by $x^{(k)}_j$ the activity on nodes j. Then, the activity $x^{(k)}_i$ on node i reads:

$$x^{(k+1)}_i = \sum_{j=1}^{N_k} \left(\lambda^{(k)}_{m(j)} \Phi^{(k)}_{l(i),m(j)} x^{(k)}_j \right) - \lambda^{(k)}_{l(i)} \sum_{j=1}^{N_k} \left(\Phi^{(k)}_{l(i),m(j)} x^{(k)}_j \right) \tag{3.5}$$

The eigenvalues $\lambda^{(k)}_{m(j)}$ modulate the density at the origin, while $\lambda^{(k)}_{l(i)}$ set the excitability of the receiver nodes, weighting the network activity in its immediate neighbourhood. As remarked in [30], this can be rationalized as the artificial analogue of the *homeostatic plasticity*, the strategy used by living neurons to maintain the synaptic basis for learning, respiration, and locomotion [135].

Starting from this background, we shall hereafter operate within a simplified setting which is obtained by imposing $\lambda^{(k)}_{m(j)} = 0$. This implies that $\lambda^{(k)}_{l(i)}$ are the sole eigenvalues to be actively involved in the training. As we shall prove, these latter eigenvalues provide an effective criterion to rank a posteriori, i.e. upon training being completed, the relative importance of the nodes belonging to the examined network. This motivates us to introduce, and thoroughly test, an effective spectral pruning strategy which seeks at removing the nodes deemed unessential, while preserving the overall network classification score. The Methods Section is entirely devoted to explain in detail the proposed strategy, that we shall contextualize with reference to other existing methodologies.

3.3 Related work

Generally speaking, it is possible to ideally group various approaches for network compression into five different categories: Weights Sharing, Network Pruning, Knowledge Distillation, Matrix Decomposition and Quantization [29, 113].

Weights Sharing defines one of the simplest strategies to reduce the number of parameters, while allowing for a robust feature detection. The key idea is to have a shared set of model parameters between layers, a choice which reflects back in an effective model compression. An immediate example of this methodology are the convolutional neural networks [84]. A refined

approach is proposed in Bat et al. [9] where a virtual infinitely deep neural network is considered. Further, in Zhang et al. [151] an ℓ_1 group regularizer is exploited to induce sparsity and, simultaneously, identify the subset of weights which can share the same features.

Network Pruning is arguably one of the most common technique to compress Neural Network: in a nutshell it aims at removing a set of weights according to a certain criterion (magnitude, importance, etc). Chang et al. [28] proposed an iterative pruning algorithm that exploits a continuously differentiable version of the $\ell_{\frac{1}{2}}$ norm, as a penalty term. Molchanov et al. [109] focused on pruning convolutional filters, so as to achieve better inference performances (with a modest impact on the recorded accuracy) in a transfer leaning scenario. Starting from a network fine-tuned on the target task, they proposed an iterative algorithm made up of three main parts: (i) assessing the importance of each convolutional filter on the final performance via a Taylor expansion, (ii) removing the less informative filters and (iii) re-training the remaining filters, on the target task. Inspired by the pioneering work in [50], Pau de Jorge et al. [37] proved that pruning at initialization leads to a significant performance degradation, after a certain pruning threshold. In order to overcome this limitation they proposed two different methods that enable an initially trimmed weight to be reconsidered during the subsequent training stages.

Knowledge Distillation is yet another technique, firstly proposed by Hinton et al. [66]. In its simplest version Knowledge Distillation is implemented by combining two objective functions. The first accounts for the discrepancy between the predicted and true labels. The second is the cross-entropy between the output produced by the examined network and that obtained by running a (generally more powerful) trained model. In [119] Polino et al. proposed two approaches to mix distillation and quantization (see below): the first method uses the distillation during the training of the so called student network under a fixed quantization scheme while the second exploits a network (termed the teacher network) to directly optimize the quantization. Mirzadeh et al. [106] analyzed the regime in which knowledge distillation can be properly leveraged. They discovered that the representation power gap of the two networks (teacher and student) should be bounded for the method to yield beneficial effects. To resolve this problem, they inserted an intermediate network (the assistant), which sits in between the teacher and the student, when their associated gap is too large.

Matrix Decomposition is a technique that remove redundancies in the parameters by the means of a tensor/matrix decomposition. Masana et al. [102] proposed a matrix decomposition method for transfer learning scenario. They showed that decomposing a matrix taking into account the activation outperforms the approaches that solely rely on the weights. In [114], Novikov et al. proposed to replace the dense layer with its Tensor-Train representation [115]. Yu et al. [148] introduced a unified framework, integrating the low-rank and sparse decomposition of weight matrices with the feature map reconstructions.

Quantization, as also mentioned above, aims at lowering the number of bits used to represent any given parameter of the network. Stock et al. [134] defined an algorithm that quantize the model by minimizing the reconstruction error for inputs sampled from the training set distribution. The same authors also claimed that their proposed method is particularly suited for compressing residual network architectures and that the compressed model proves very efficient when run on CPU. In Banner et al. [10] a practical 4-bit post-training quantization approach was introduced and tested. Moreover, a method to reduce network complexity based on node-pruning was presented by He et al. in [65]. Once the network has been trained, nodes are classified by means of a node importance function and then removed or retained depending on their score. The authors proposed three different node ranking functions: entropy, output-weights norm (onorm) and input-weights norm (inorm). In particular, the input-weights norm function is defined as the sum of the absolute values of the incoming connections weights. As we will see this latter defines the benchmark model that we shall employ to challenge the performance of the trimming strategy here proposed. Finally, it is worth mentioning the Conditional Computation methods [13, 143, 144]: the aim is to dynamically skip part of the network according to the provided input so as to reduce the computational burden.

Summing up, in contrast with the pruning techniques which primarily pursue the goal of enforcing a sparsification by cutting connections from the trained neural network, the idea of our method is to a posteriori identify the nodes of the trained network which prove unessential. This yields a more compact neural network, in terms of composing neurons, with unaltered classification performance. The method relies on the spectral learning [30, 52] and exploits the fact that eigenvalues are credible parameters to gauge the importance of a given node among those composing the destination layer.

In short, our aim is to make the network more compact by removing nodes classified as unimportant, according to a suitable spectral rating.

3.4 Proposed method

We detail here the spectral procedure to make a trained network smaller, while preserving its ability to perform classification.

To introduce the main idea of the proposed method, we make reference to formula (3.4) and assume the setting where $\lambda^{(k)}_{m(j)} = 0$. The information travelling from layer k to layer $k+1$ gets hence processed as follows: first, the activity on the departure node j is modulated by a multiplicative scaling factor $\Phi^{(k)}_{l(i),m(j)}$, specifically linked to the selected (i,j) pair. Then, all incoming (and rescaled) activities reaching the destination node i are summed together and further weighted via the scalar quantity $\lambda^{(k)}_{l(i)}$. This latter eigenvalue, downstream of the training, can be hence conceived as a distinguishing feature of node i of layer $k+1$. Assume for the moment that $\Phi^{(k)}_{l(i),m(j)}$ are drawn from a given distribution and stay put during optimization. Then, every individual neuron bound to layer $k+1$ is statistically equivalent (in terms of incoming weights) to all other nodes, belonging to the very same layer. The eigenvalues $\lambda^{(k)}_{l(i)}$ gauge therefore the relative importance of the nodes, within a given stack, and as reflecting the (randomly generated) web of local inter-layer connections (though statistically comparable). Large values of $|\lambda^{(k)}_{l(i)}|$ suggest that node i on layer $k+1$ plays a central role in the economy of the neural network functioning. This is opposed to the setting when $|\lambda^{(k)}_{l(i)}|$ is found to be small. Stated differently, the subset of trained eigenvalues provide a viable tool to rank the nodes according to their degree of importance. As such, they can be used as reference labels to make decision on the nodes that should be retained in a compressed analogue of the trained neural network, with unaltered classification performance. As empirically shown in the Results section with reference to a variegated set of applications, the sorting of the nodes based on the optimized eigenvalues turns out effective also when the eigenvectors get simultaneously trained, thus breaking, at least in principle, statistical invariance across nodes.

As we will clarify, the latter setting translates in a post-training spectral pruning strategy, whereas the former materializes in a rather efficient pre-training procedure. The non linear activation function as employed in the training scheme leaves a non trivial imprint, which has to be critically

assessed.

More specifically, in carrying out the numerical experiments here reported we considered two distinct settings, as listed below:

1. As a first step, we will begin by considering a deep neural network made of N neurons organized in ℓ layers. The network will be initially trained by solely leveraging on the set of tunable eigenvalues. Then, we will proceed by progressively removing the neurons depending on their associated eigenvalues (as in the spirit discussed above). The trimmed network, composed by a total of $M < N$ units, still distributed in ℓ distinct layers, can be again trained acting now on the eigenvectors, while keeping the eigenvalues frozen to the earlier determined values. This combination of steps, which we categorize as pre-training, yields a rather compact neural network (M can be very small) which performs equally well than its fully trained analogue made of N computing nodes.

2. We begin by constructing a deep neural network made of N neurons organized in ℓ layers. This latter undergoes a full spectral training, which optimizes simultaneously eigenvectors and the eigenvalues. The trained network can be compressed, by pruning the nodes which are associated to eigenvalues (see above) with magnitude smaller that a given threshold. This is indeed a post-training pruning strategy, as it acts *ex post* on a fully trained device.

To evaluate the performance of the proposed spectral pruning strategies (schematically represented in the flowchart of Figure 3.4), we also introduced a reference benchmark model. This latter can be conceptualized as an immediate overturning of the methods in direct space. Simply stated, we train the neural network in the space of the nodes, by using standard approaches to the learning. Then, we classify the nodes in terms of their relevance using a proper metric to which shall make reference below, and consequently trim the nodes identified as less important. When adopting the spectral viewpoint, one can rely on the eigenvalues to rank the nodes importance. As remarked above, in fact, the eigenvalues at the receiver nodes set a local scale for the incoming activity, the larger the eigenvalue (in terms of magnitude) the more important the role played by the processing unit. As a surrogate of the eigenvalues, when anchoring the train in direct space, we can consider the quantity $\sum_{j=1}^{N_k} |w_{ij}|$, for each neuron i belonging to layer

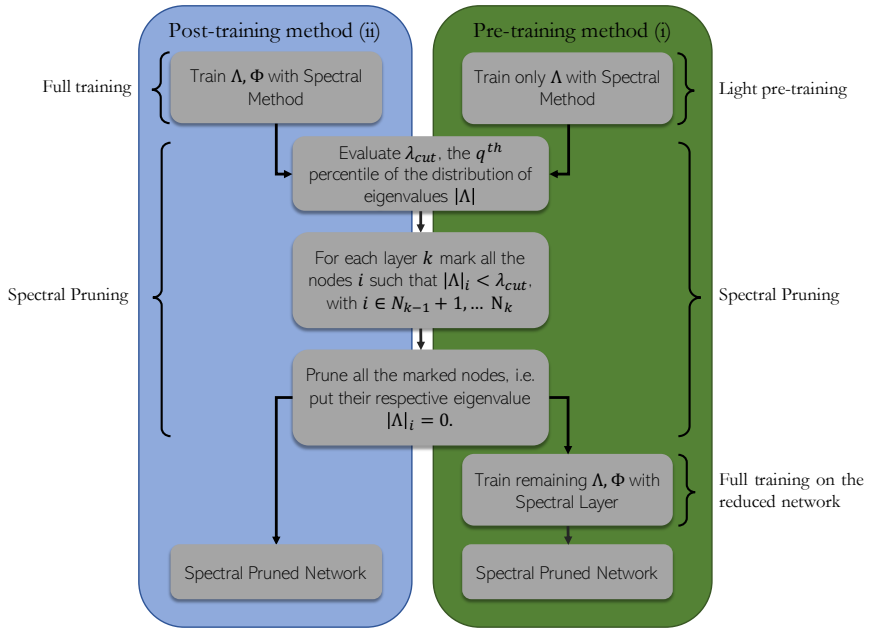


Figure 3.4: Flowchart of the pre- and post- speactral training pruning strategies as presented in section 3.4.

$k+1$, see also [65]. The absolute value prevents mutual cancellations of sensible contributions bearing opposite signs, which could incidentally hide the actual importance of the examined node.

In all explored cases, the pruning is realized by imposing a threshold on the reference indicator (be it the magnitude of the eigenvalues or the cumulated flux of incoming –and made positive– weights). Pointedly, the respective indicator is extracted for every node in the arrival layer. Then a percentile q is chosen and the threshold fixed to the q-th percentile. Nodes displaying an indicator below the chosen threshold are removed and the accuracy of the obtained (trimmed) neural network assessed on the test-set. The codes employed, as well as a notebook to reproduce our results, can be found in the public repository of this project [1].

[1] `https://github.com/Jamba15/SpectralTools`

3.5 Experiments

In order to assess the effectiveness of the eigenvalues as a marker of the node's importance (and hence as a potential target for a cogent pruning procedure) we will consider a fully connected feed-forward architecture. Applications of the explored methods will be reported for $\ell = 3$ and $\ell > 3$ configurations. The nodes that compose the hidden layers are the target of the implemented pruning strategies. As we shall prove, it is possible to get rid of the vast majority of nodes without reflecting in a sensible decrease in the test accuracy, if the filter, either in its pre- or post-training versions, relies on the eigenvalues ranking. Moreover, it is also important to stress that, in general terms, the pruning of unessential nodes improves the computational efficiency of the network. As a matter of fact, reducing the number of output nodes leads a compression in terms of both memory and inference time which is directly proportional to the number of removed elements. As an example, by pruning a fraction α (< 1) of the total nodes, we obtain a new layer with $\alpha \cdot N$ less neurons and a memory reduction of $\alpha \cdot N$ times the number of input features.

For our test, we used three different datasets of images. The first is the renowned MNIST database of handwritten digits [83], composed by greyscale images of dimension 28×28 pixels. The second is Fashion-MNIST (F-MNIST) [146] (an image dataset of Zalando's items) which are still depicted with a greyscale with dimension 28×28 but display an enhanced degree of inherent complexity for what concerns the type of classification requiredas compared to the basic MNIST benchmark model (more complex shapes, patterns on items). The last one is CIFAR-10 [78] a richer dataset composed by 32×32 RGB images of complex real-world objects divided in 10 classes. In the main text we report our findings for Fashion-MNIST. Analogous investigations carried out for MNIST and CIFAR10 are reported. Further, different activation functions have been employed to evaluate the performance of the methods. In the following we will report into two separate sub-sections the results pertaining to either the single or multiple hidden layers settings.

3.5.1 Single hidden layer

In Figure 3.7 the performance of the inspected methods are compared for the minimal case study of a three layers network. The intermediate layer, the sole hidden layer in this configuration, is set to $N_2 = 500$ neurons. The

accuracy of the different methods are compared, upon cutting at different percentile, following the strategies discussed in the Methods and compared with the benchmark model (the orange profile). In the benchmark model, the neural network is trained in direct space, by adjusting the weights of each individual inter-nodes connection. Then, the absolute value of the incoming connectivity is computed and used as an importance rank of the nodes' influence on the test accuracy (analogous to the way in which we use the eigenvalues). Such a model has been presented and discussed by He et al. in [65]. Following this assessment, nodes are progressively removed from the trained network, depending on the imposed percentile, and the ability of the trimmed network to perform the sought classification (with no further training) tested. We choose this particular type of trimming as a benchmark to our spectral pruning technique for the following reasons. First, it also amount to removing nodes from the collection, and not just sparsify the weight of the associated transfer matrices. Then, both approaches build on the concept of nodes ranking, as obtained from a suitable metric, which is respectively bound to direct vs. spectral domains. The abovementioned procedure is repeated 5 times and the mean value of the accuracy plotted in the orange curve of Figure 3.7. The shaded region stands for the semi dispersion of the measurements. A significant drop of the network performance is found when removing a fraction of nodes larger than 60 % from the second layer.

The blue curve Figure 3.7 refers instead to the post-processing spectral pruning based on the eigenvalues and identified, as method (ii), in the Methods Section. More precisely, the three layers network is trained by simultaneously acting on the eigenvectors and the eigenvalues of the associated transfer operators, as illustrated above. The accuracy displayed by the network trained according to this procedure is virtually identical to that reported when the learning is carried out in direct space, as one can clearly appreciate by eye inspection of Figure 3.7. Removing the nodes based on the magnitude their associated eigenvalues, allows one to keep stable (practically unchanged) classification performance for an intermediate layer that is compressed of about 70% of its original size. In this case the spectral pruning is operated as a post-processing filter, meaning that the neural network is only trained once, before the nodes' removal takes eventually place.

At variance, the green curve in Figure 3.7 is obtained following method (i) from the Methods Section, which can be conceptualized as a pre-training

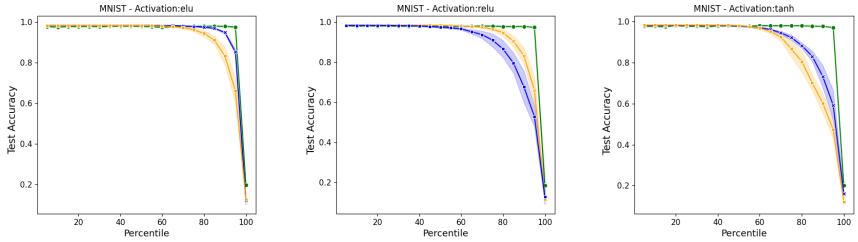
manipulation. Based on this strategy, we first train the network on the set of tunable eigenvalues, than reduce its size by performing a compression that reflects the ranking of the optimized eigenvalues and then train again the obtained network by acting uniquely on the ensemble of residual eigenvectors. The results reported in Figure 3.7 indicate that, following this procedure, it is indeed possible to attain astoundingly compact networks with unaltered classification abilities. Moreover, the total number of parameters that need to be tuned following this latter procedure is considerably smaller than that on which the other methods rely. This is due to the fact that only the random directions (the eigenvectors) that prove relevant for discrimination purposes (as signaled by the magnitude of their associated eigenvalues) undergoes the second step of the optimization. This method can also be seen as a similar kind of [50]. As a matter of fact, the initial training of the eigenvalues uncovers a sub-network that, once trained, obtains performances comparable to the original model. More specifically, the uncovered network can be seen as a *winning ticket* [50]. That is, a sub-network with an initialization particularly suitable for carrying out a successful training.

We report (see Figures 3.5a, 3.5b, 3.5c, 3.6a and 3.6b) on the performance of the proposed trimming strategies, as applied to MNIST and Fashion-MNIST, for a single hidden layer architecture and using the same setting as before. In particular, we will show ELU, tanh and ReLU for MNIST (ii) tanh, and ReLU as additional activation functions for Fashion-MNIST.

Next, we shall generalize the analysis to the a multi-layer setting ($\ell > 3$), reaching analogous conclusions.

3.5.2 Multiple hidden layers

The results achieved in the simplified context of a single hidden layer network also apply within the framework of a multi-layers setting.
To prove this statement we set to consider a $\ell = 5$ feedforward neural network with ELU activation. Here, $N_1 = 784$ and $N_5 = 10$ as reflecting the specificity of the employed dataset. The performed tests follows closely those reported above, with the notable difference that now the ranking of the eigenvalues is operated on the pool of $N_2 + N_3 + N_4$ neurons that compose the hidden bulk of the trained network. In other words, the selection of the neuron to be removed is operated after a global assessment, i.e. scanning across the full set of nodes, without any specific reference to an a priori chosen layer.

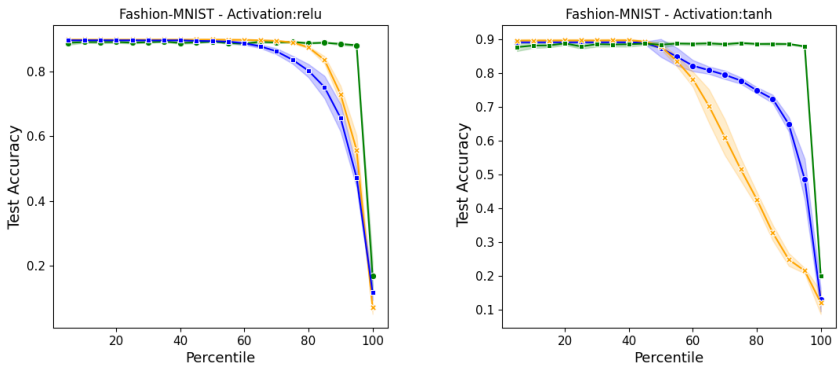


(a) (b) (c)

Figure 3.5: Accuracy on the MNIST database with respect to the percentage of trimmed nodes (selected from the 500 neurons that compose the sole hidden layer), in a three layers feedforward architecture. The results reported in each panel refer to a different selection of the nonlinear activation functions, respectively ELU (a), ReLU (b) and tanh (c). In orange, the results obtained by using the trimming procedure based on the absolute value of the incoming connectivity. In blue, the results obtained when filtering the nodes after a full spectral training (post-training). The curve in green displays the accuracy of the trimmed networks generated upon application of the pre-training filter. In this case, the examined network is initially trained on the set of eigenvalues, while keeping the eigenvectors frozen. After having removed unessential nodes, based on their associated eigenvalues, the network undergoes another training phase that is solely targeted to adjusting the entries of the residual eigenvectors. The shadowed region represents the semi-dispersion over 5 independent realizations. When using the ReLU function, trimming on the absolute value of the incoming connectivity yields slightly better results than what found when using the post-training spectral filter. The two stages spectral trimming proves always more effective.

In Figure 3.8 the results of the analysis are reported, assuming $N_2 = N_3 = N_4 = 500$. The conclusions are perfectly in line with those reported above for the one layer setting, except for the fact that now the improvement of the spectral pruning over the benchmark reference are even superior. The orange curve drops at percentile 20, while the blue begins its descent at about 60 %. The green curve, relative to the sequential two steps training, stays stably horizontal up to about 90 %.

Finally, see Figures 3.9a, 3.9b, 3.9c, 3.10a and 3.10b we report the result for multiple activation functions (ELU, ReLU, and tanh) for both MNIST



(a) (b)

Figure 3.6: Accuracy on the Fashion-MNIST database with respect to the percentage of trimmed nodes (selected from the 500 neurons that compose the sole hidden layer), in a three layers feedforward architecture. The results reported in each panel refer to a different selection of the nonlinear activation functions, respectively ReLU (b) and tanh (c). Symbols and conclusions are in line with those reported for the case of MNIST.

and Fashion-MNIST. In line with the setting choose above, we will assume a five layered deep neural network with $N_2 = N_3 = N_4 = 500$.

3.5.3 CIFAR-10

To assess the flexibility of the schemes outlined in Section III-B we here consider the CIFAR10 dataset and assume a modified MobileNetV2 [126] adding two dense layer at the end of the network. During training we freeze all the layers (pretrained on ImageNet), except for the two appended dense layers. These latter are trained in the spectral domain from scratch. Working in this setting, the pruning is performed on the first dense layer by using both strategies (i) and (ii), as introduced in the main body of the chapter. Here again the results are compared to those obtained when using the absolute value of the incoming connectivity as an alternative trimming criterion (see Figures 3.11a, 3.11b and 3.11c). As a further step in the analysis, we also introduce and test a ℓ_1-norm regularization acting on the eigenvalues, so as to induce a sparse solution [7]. All experiments are performed by using a

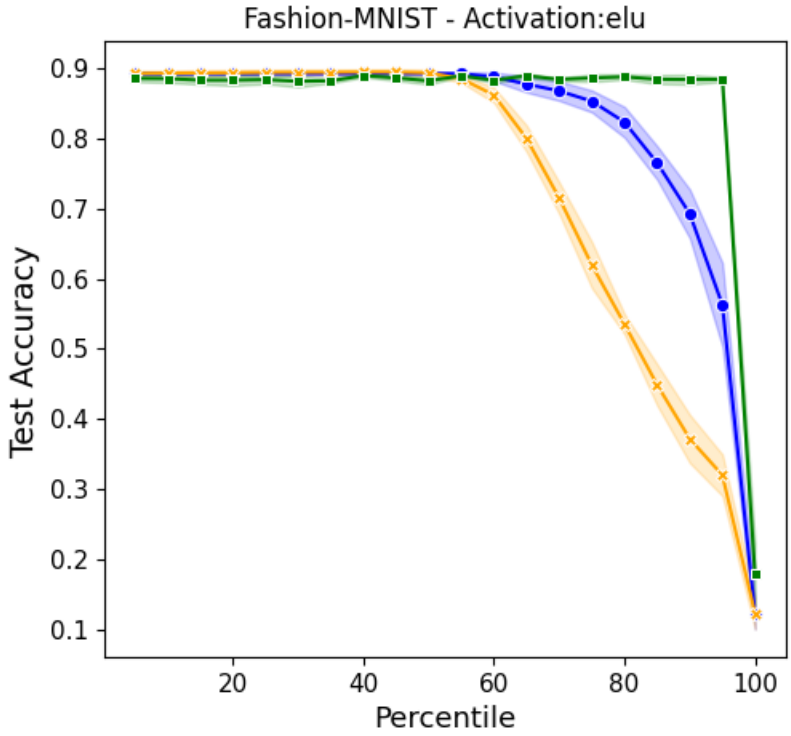


Figure 3.7: Accuracy on the Fashion-MNIST database with respect to the percentage of trimmed nodes (from the hidden layer), in a three layers feed-forward architecture. Here, $N_2 = 500$, while $N_1 = 784$ and $N_3 = 10$, as reflecting the structural characteristics of the data. In orange the results obtained by pruning the network trained in direct space, based on the absolute value of the incoming connectivity (see main text). In blue, the results obtained when filtering the nodes after a full spectral training (post-training). The curve in green reports the accuracy of the trimmed networks generated upon application of the pre-training filter. Symbols stand for the averaged accuracy computed over 5 independent realizations. The shadowed region is traced after the associated semi-dispersion.

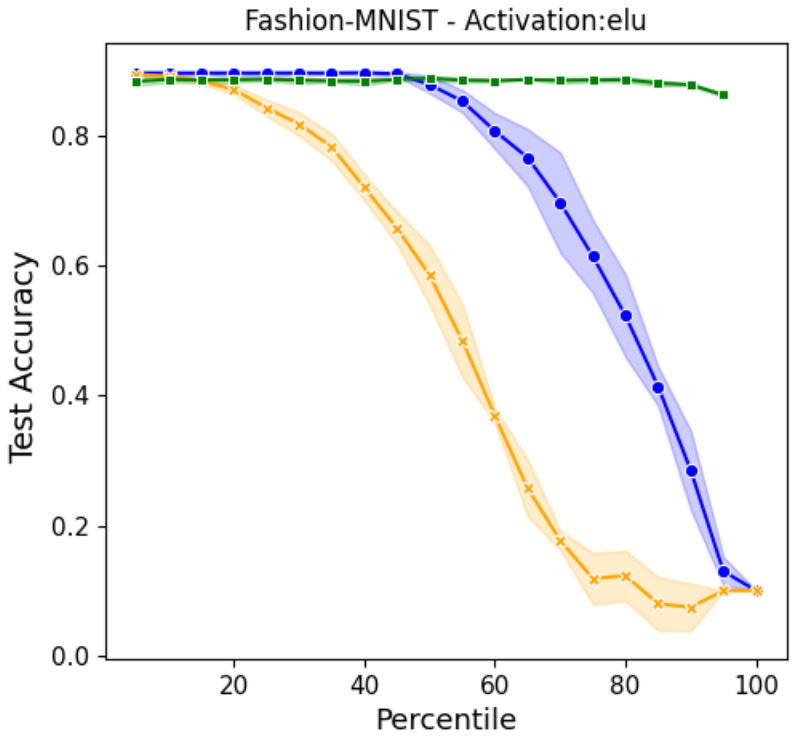
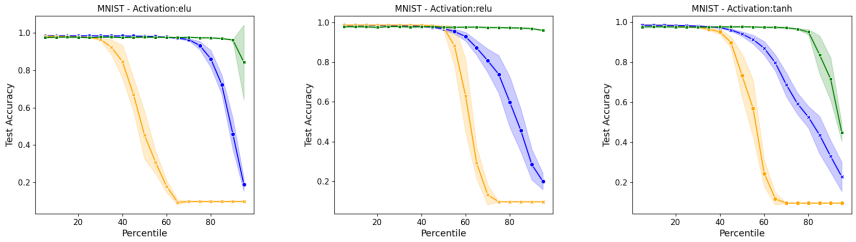


Figure 3.8: Accuracy on the Fashion-MNIST database with respect to the percentage of pruned nodes (from the hidden layers), in a five layers feed-forward architecture. Here, $N_2 = N_3 = N_4 = 500$, while $N_1 = 784$ and $N_5 = 10$, as reflecting the structural characteristics of the data. Symbols and colors are chosen as in Figure 3.7.

MobileNetV2 based architecture. The first dense layer is made of 512 nodes with an ELU activation function (others activation functions yield analogous results). The following regularization loss functions are considered depending on whether the training takes place in the reciprocal (spectral layer) or direct space:



(a) (b) (c)

Figure 3.9: Accuracy on the MNIST database with respect to the percentage of trimmed nodes (from the set of $N_2+N_3+N_4$ neurons). The results in each panel refer to different choices of the non linear function, ELU (a), ReLU (b) and tanh (c). Symbols are chosen as for the case of the single hidden layer setting. It should be remarked that the spectral trimming strategies proves definitely more effective than the benchmark model anchored to direct space, also when the ReLU function is employed, in the case of multiple hidden layers.

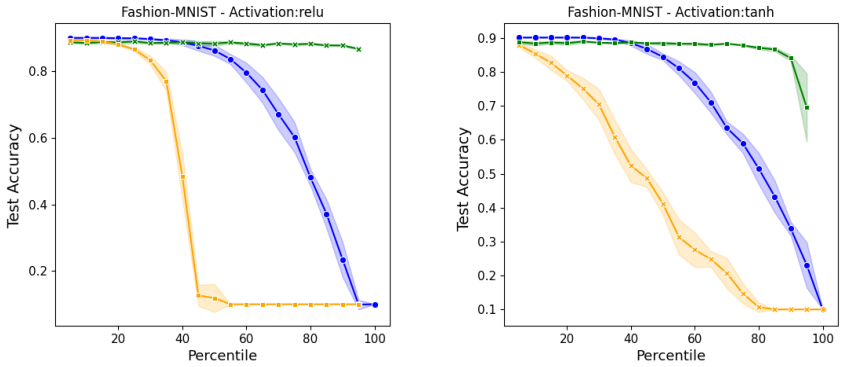
- Spectral regularization

$$L_r^{\text{spec}} = \gamma * \sum_{i=1}^{N_{\ell-1}} |\lambda_i^{(\ell-1)}|$$

- Connectivity regularization

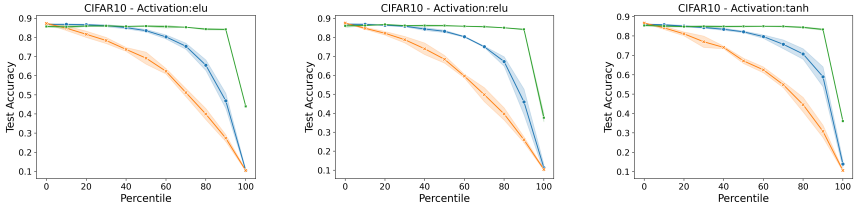
$$L_r^{\text{conn}} = \gamma * \sum_{i,j} |w_{ij}^{(\ell-1)}|$$

where γ stands for a suitable regularizer weight.
Clearly L_r^{conn} is equivalent to a regularization which acts on the incoming absolute connectivity. In fact, $|\sum_i |x_i|| = \sum_i |x_i|$.
The ℓ_1 regularization impacts significantly on the classification accuracy, as it can be clearly appreciated by direct inspection of Figure 3.12.
Choosing the correct regularizer weight (γ), the performance of the network are stable across various range of pruning thresholds, even at the highest percentile.



(a) (b)

Figure 3.10: Accuracy on the Fashion-MNIST database with respect to the percentage of trimmed nodes (from the set of $N_2 + N_3 + N_4$ neurons). The results in each panel refer to different choices of the non linear activation function, ReLU (a) and tanh (b). For the symbols, see the caption of the Figures above. Also in this case the spectral filters prove always superior.

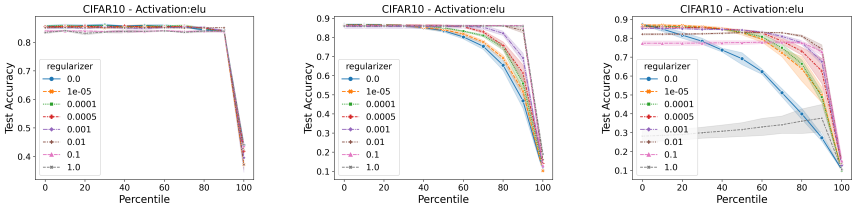


(a) (b) (c)

Figure 3.11: Accuracy on the CIFAR10 database with respect to the percentage of trimmed nodes (from the $\ell - 1$ layer). The results in each panel refer to different non linear functions, respectively ELU (a), ReLU (b) and tanh (c). Symbols are chosen in analogy with the above (the result drawn in green are based on two different runs).

3.6 Final considerations

In this Chapter we have discussed a relevant byproduct of a spectral approach to the learning of deep neural networks. The eigenvalues of the



(a) (b) (c)

Figure 3.12: Computed accuracy on the CIFAR10 dataset against the percentage of trimmed nodes (from the first of the two dense layers appended to the MobileNet-like architecture). The panels displays the performance of the network as according to each trimming procedure, and using weights (W) for the ℓ_1 regularizer. In panel (a) and (b) pre-training (based on two runs) and post-spectral filter, respectively; in panel (c) the reduction schem based on the absolute connectivity.

transfer operator that connects adjacent stacks in a multi-layered architecture provide an effective measure of the nodes importance in handling the information processing. By exploiting this fact we have introduced and successfully tested two distinct procedures to yield compact networks –in terms of number of computing neurons– which perform equally well than their untrimmed original homologous. One procedure (referred as (ii) in the description) is acknowledged as a post processing method, in that it acts on a multi-layered network downstream of training. The other (referred as (i)) is based on a sequence of two nested operations. First the eigenvalues are solely trained. After the spectral pruning took place, a second step in the optimization path seeks to adjust the entries of the eigenvectors that populate a trimmed space of reduced dimensionality. The total number of trained parameters is small as compared to that involved when the pruning acts as a post processing filter. Despite that, the two steps pre-processing protocol yields compact devices which outperform those obtained with a single post-processing removal of the unessential nodes.

As a benchmark model, and for a neural network trained in direct space, we decided to rank the nodes importance based on the absolute value of the incoming connectivity. This latter appeared as the obvious choice, when aiming at gauging the local information flow in the space of the nodes, see also [65]. In principle, one could consider to diagonalizing the transfer op-

erators as obtained after a standard approach to the training and make use of the computed eigenvalues to a posteriori sort the nodes relevance. This is however not possible as the transfer operator that links a generic layer k to its adjacent counterpart $k+1$, as follows the training performed in direct space, is populated only below the diagonal, with all diagonal entries identically equal zero. All associated eigenvalues are hence are zero and they provide no information on the relative importance of the nodes of layer $k+1$, at variance with what happens when the learning is carried out in the reciprocal domain.

Summing up, by reformulating the training of neural networks in spectral space, we identified a set of sensible scalars, the eigenvalues of suitable operators, that unequivocally correlate with the influence of the nodes within the collection. This observation translates in straightforward procedures to generate efficient networks that exploit a reduced number of computing units. Tests performed on different settings corroborate this conclusion.

Chapter 4

Pruning of Convolutional Layer's Filters

Identifying, possibly in an automated and efficient manner, the proper structure for a deep network to tackle a specific task is one of the most relevant problems in modern machine learning. In some cases, starting from a given architecture, the final neural architectures must be subject to constraints on the network's size. In this particular setting, we usually refer to Neural Compression. Taking inspiration from the recently proposed differentiable architectural search framework, we consider generic bilevel programming problems with sparsity constraints to solve the network compression problem. These problems underlie a combinatorial aspect since they imply a strong sparsity requirement hidden in the training problem. We provide the definition and a thorough theoretical analysis of a tailored penalty decomposition approach for this class of problems.

4.1 Preamble

One of the most relevant problems in modern machine learning is that of identifying, possibly in an automated and efficient manner, the right structure for a deep network to tackle a specific task; this is the well-known problem of *Neural Architecture Search* (NAS). The required neural architectures may be subject to constraints; for example, bounds on the size of the network may be imposed in order for the model to be employable with limited hardware. In this particular setting, we usually refer to *Neural Network Compression*.

In a popular work of 2018, Liu et al. [92] proposed DARTS, a methodology for NAS which is based on differentiable optimization tools and that has subsequently been improved in recent years. Moreover, differentiable optimization techniques have also been proposed for neural compression problems, e.g., [27, 111]. In this Chapter we propose a compression techniques based on a bilevel optimization problem. In such approaches of neural network training, not only weights are optimized so as to minimize training loss, but some hyperparameters are also optimized, trying to improve the performance on the validation set and, in parallel, selecting only the relevant part of the network. Similar bilevel setting can also be found in hyperparameters optimization [48, 100, 118], data denoising [94, 123], meta-learning [49, 122], data poisoning [76, 104].

As we will detail later in this Chapter, both in DARTS [92] and in compression algorithms decisions of combinatorial nature are implicitly made. As a matter of fact, we are basically requiring a sparse solution after the model's training.

Sparsity is a recurrent requirement in real-world optimization problems [139] and different classes of methods have been proposed to deal with sparsity-constrained problems, even in the nonlinear nonconvex case [11, 24, 80]. Among these algorithms, the class of Penalty Decomposition (PD) methods received attention in recent years [82, 98]. Indeed, [27] showed that a penalty decomposition approach is effective with neural compression problems: a sequential penalty method allows the weights to progressively adapt to the sparse structure of the network and avoids performance drops that are typical of methods that carry out abrupt thresholding operations. An analogous reasoning can be made for DARTS, where sparsity is induced by simplex constraint and involves a final discretization step that often undermines the predictive performance of the obtained network.

Motivated by the above remarks, we consider generic bilevel programming problems with sparsity constraints. For this class of problems, we provide the definition and a thorough theoretical analysis of a tailored penalty decomposition approach.

We outline that, similarly to the sequential penalty method proposed by [103] for smooth deep bilevel problems, we are able to prove theoretical results of convergence under suitable assumptions. This is in contrast with the approaches born from [92]: as we will discuss in detail, [92] is not designed to solve the considered bilevel problem. In its faster, but effective, formulation is rather an algorithm for a Nash equilibrium problems that, as we see later in Section 4.2.1, lack of convergence guarantees.

4.2 Related work

Liu et al. [92] propose a method to solve problems of the form

$$\begin{aligned} &\min_{\alpha,\bar{w}} f(\bar{w},\alpha) \\ &\text{s.t. } \bar{w} \in \arg\min_{w} g(w,\alpha), \end{aligned} \tag{4.1}$$

where f usually denote the validation loss, g the training loss, α are hyperparameters and w are the weights of the network (this bilevel formulation also appear on hyperparameters optimization [49]). More specifically, they propose a method where the following two operations are (approximately) carried out repeatedly:

$$w^{k+1} \in \arg\min_{w} g(w,\alpha^k) \qquad \alpha^{k+1} \in \arg\min_{\alpha} f(w^{k+1},\alpha). \tag{4.2}$$

For the sake of completeness, we also report that they propose a slightly more complex (but slower) method were another gradient step is considered inside 4.2.

4.2.1 Alternate Minimization Approaches

Convergence properties for the above-mentioned algorithm 4.2 are not known. As a matter of fact, even Liu et al. [92] state that convergence properties of DARTS are not clear and theoretical analysis concerning this aspect was not provided. In this Section, we argue that convergence results for Liu et al. [92] proposal are not known because, basically, there is not any.

Carefully looking at the algorithmic structure of this simplified approach, we realize that this is in fact not designed to directly tackle bilevel formulation (4.1). The two optimization steps are carried out optimizing two different objective functions with respect to two disjoint subset of variables. Specifically, this process can be interpreted as a non-cooperative game between two players that independently try to improve their own reward. In other words, the optimization problem underlying this well-known approach is in fact a Nash equilibrium problem, defined by the pair of problems:

$$\arg\min_{w} g(w, \alpha) \qquad \arg\min_{\alpha} f(w, \alpha).$$

Recalling that a solution $(\bar{w}, \bar{\alpha})$ is a Nash equilibrium if

$$f(\bar{w}, \bar{\alpha}) \leq f(\bar{w}, \alpha) \; \forall \, \alpha, \qquad g(\bar{w}, \bar{\alpha}) \leq g(w, \bar{\alpha}) \; \forall \, w.$$

It should be easy to see (see Example 2) that such a Nash equilibrium is not necessarily a solution of the bilevel problem (4.1) where we basically seek for optimality of f with respect to the entire pair of variables (w, α).

Gauss-Seidel type alternate minimization schemes like (4.2) are known in the literature of equilibrium problems as Best-response algorithms [45]. Unfortunately, these methods are not guaranteed to converge to Nash equilibria: convergence results can be stated if potential games are taken into account. Moreover, in this latter setting, convergence can be guaranteed if there are only two players, or under convexity assumptions on the potential function (i.e., a function that both players are implicitly optimizing at each step) and introducing a proximal-point regularizer [45].

However, if f and g denote training and validation losses, a potential function [110] is unlikely to exist.

In conclusion, although methods of the form (4.2) have empirically proven to perform quite well in deep bilevel optimization tasks, it is not possible to support these good results with theoretical results: the methods are indeed designed to tackle a weaker formulation of the problem and not necessarily convergent even under strong regularity assumptions.

Hereafter, we provide a counterexample to illustrate the issues discussed in this Section.

Example 2. Let us consider the following Nash equilibrium problem:

$$\min_{x} \; f(x, y) = x^2 - 4xy, \qquad \min_{y} \; g(x, y) = (1 - y)^2 + 4xy.$$

The above game is defined by two unconstrained, continuously differentiable strictly convex optimization problems, i.e., strong regularity assumptions hold. However, a potential function does not exist for this game. Forcing the first order optimality conditions for the two problems, it is easy to see that the optimal solution for a player, for a given strategy of the other player, is given by

$$x^*(y) = 2y \qquad\qquad y^*(x) = 1 - 2x.$$

The (unique) Nash equilibrium is thus attained at $(x, y) = (2/5, 1/5)$. However, consider the Best Response algorithm starting at $(0, 0)$; a sequence of solutions is produced as follows:

$$(0,0) \xrightarrow{y=1} (0,1) \xrightarrow{x=2} (2,1) \xrightarrow{y=-3} (2,-3) \xrightarrow{x=-6} (-6,-3) \xrightarrow{\cdots}$$

which is clearly a divergent sequence; in fact,

$$y^{k+1} = 1 - 2x^k = 1 - 4y^k = \ldots = \sum_{i=0}^{k} (-4)^{k-i} = \sum_{j=0}^{k} (-4)^j,$$

which is a classical divergent geometric series.

Now, let us consider the "corresponding" bilevel problem

$$\begin{aligned} \min_x \; & f(x,y) = x^2 - 4xy \\ \text{s.t. } & y \in \arg\min_y g(x,y) = (1-y)^2 + 4xy. \end{aligned}$$

The lower-level problem is convex for all x and the corresponding solution is always unique; thus Assumption 1 holds and we can reformulate the problem as

$$\begin{aligned} \min_{x,y} \; & f(x,y) = x^2 - 4xy \\ \text{s.t. } & y = 1 - 2x. \end{aligned}$$

Substituting the linear constraint in the objective function, we get

$$\min_x \; x^2 - 4x + 8x^2 = 9x^2 - 4x,$$

whose minimizer is at $x = 2/9$. The feasible pair $(2/9, 5/9)$ is thus the optimal solution of the bilevel problem, and has indeed an objective value of $-4/9$ which is better that the value $(-4/25)$ attained at the Nash equilibrium point $(2/5, 1/5)$.

Thus, applying the best response algorithm in this case we might obtain a divergent sequence, and even in fortunate cases we would only converge to a Nash equilibrium which is suboptimal for the bilevel problem. On the other hand, standard algorithms for constrained optimization are in general guaranteed to converge to a feasible stationary point under constraints qualification; in the considered case, we have linear constraints and a convex objective function, thus we would be guaranteed to converge to the global minimizer of the problem [14].

4.3 Proposed method

Let $f(w,\, \alpha)$ and $g(w,\, \alpha)$ the validation and the training losses (w are the network weights and α are the hyper-parameters). Moreover, we modify the standard 2D convolutional layer (Conv2d) with weights $w \in \mathbb{R}^{C_{out} \times C_{in} \times K_0 \times K_0}$ where: C_{out} is the number of output channels, C_{in} the number of input channels and, K_0, K_1 are the kernel dimensions. For the sake of simplicity and without loss of generality, in the following, we will assume $K_0 = K_1 = K$. More specifically, considering an input tensor $x \in \mathbb{R}^{C_{in} \times H \times W}$, the j-th output channel of Conv2d(x) is defined as

$$\text{Conv2d}(x)[j,\, :,\, :] = \sum_{i=0}^{C_{in}} w[j,\, i,\, :,\, :] \circledast x[i,\, :,\, :]$$

where $\circledast$ denotes the cross-correlation operator. We propose to add a set of hyper-parameters $\alpha \in \mathbb{R}^{C_{out}}$ to control the scale of each output filter (see Figure 4.1). Therefore, we define our GatedConv2d (see Figure 4.1 to see how the hyper-parameters control the network behavior) as

$$\text{GatedConv2d}(x)[j,\, :,\, :] = \sum_{i=0}^{C_{in}} (w[j,\, i,\, :,\, :] \cdot \alpha[j]) \circledast x[i,\, :,\, :] \tag{4.3}$$

Finally, taking inspiration from the bilevel problem proposed in Liu et al. [92], we optimize the hyper-parameters α with respect to the validation set and adding a constraint on the maximum number of α different from zero. Spefically, we wanto to solve the following bilevel problem:

$$\begin{aligned} &\min_{\alpha, \bar{w}} f(\bar{w}, \alpha) \\ &\text{s.t. } \bar{w} \in \arg\min_{w} g(w, \alpha) \\ &\quad \|\alpha\|_0 \leq C \end{aligned} \tag{4.4}$$

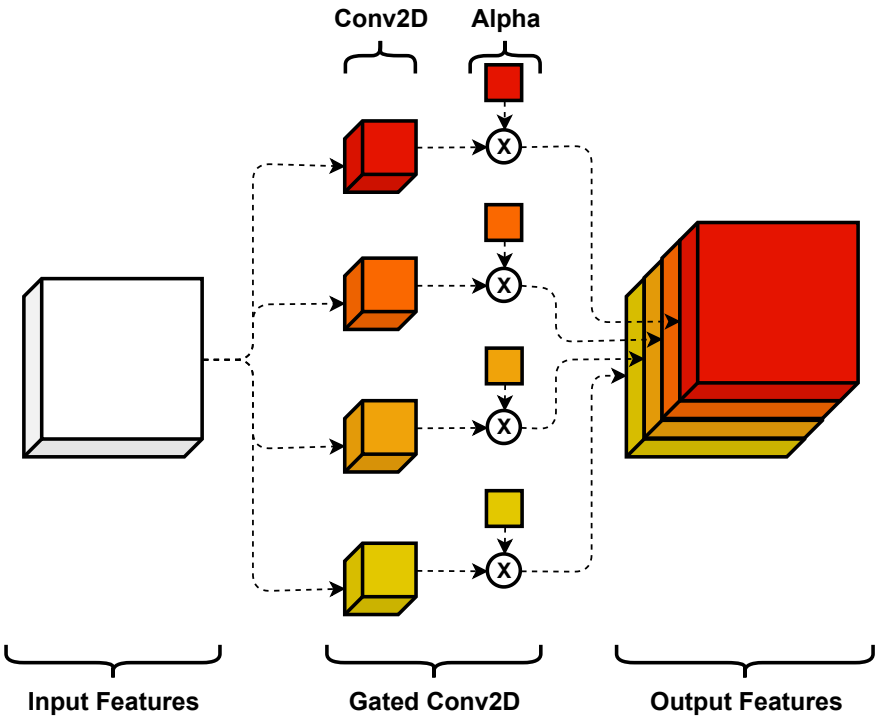


Figure 4.1: Our proposed Conv Layer

where C is the maximum number of channels that can be used.

4.3.1 Theoretical analysis

For this theoretical analysis, we can abstract from the particular realization of f and g being two loss functions. As a matter of fact, the theory developed in this part of the Chapter is more general and can be applied to more general problems. Specifically, let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ two continuously differentiable functions. We are interested in solving bilevel

optimization problem of the form

$$\begin{aligned} \min_{\alpha} \quad & f(\bar{w}, \alpha) \\ \text{s.t.} \quad & \bar{w} \in \arg\min_{w} \; g(w, \alpha), \\ & \|\alpha\|_0 \leq C, \end{aligned} \tag{4.5}$$

where the peculiarity of the problem lies in the sparsity constraint required for the upper-level variables α.

Under suitable assumptions (e.g., the convexity of $g(\cdot, \alpha)$ for all α), the lower level problem can straightforwardly be substituted by a constraint $\nabla_w g(\bar{w}, \alpha) = 0$. Moreover, under the assumption that the optimal set of the lower-level problem is a singleton for all α, the optimistic and pessimistic variants of problem (4.5) coincide [129]; under these hypotheses, we are allowed to reformulate problem (4.5) as

$$\begin{aligned} \min_{w,\alpha} \quad & f(w, \alpha) \\ \text{s.t.} \quad & h(w, \alpha) = \nabla_w g(w, \alpha) = 0, \\ & \|\alpha\|_0 \leq C. \end{aligned} \tag{4.6}$$

In the following discussions, we will always assume that the aforementioned assumptions are satisfied. More explicitly, we assume that the following statement holds.

Assumption 1. A pair $(\bar{w}, \bar{\alpha})$ is a feasible solution for problem (4.5) if and only if it is feasible for problem (4.6). Moreover, $(\bar{w}, \bar{\alpha})$ is an optimal solution of problem (4.5) if and only if it is optimal for problem (4.6).

Applying the classical variable splitting technique [57, 72], the problem can be equivalently rewritten as

$$\begin{aligned} \min_{w,\alpha,\beta} \quad & f(w, \alpha) \\ \text{s.t.} \quad & h(w, \alpha) = 0, \\ & \|\beta\|_0 \leq C, \\ & \alpha = \beta. \end{aligned} \tag{4.7}$$

Basically, a new vector of variables β has been introduced, to decouple the objective function and the sparsity constraint. Then, variables α and β are linked by a simple equality constraint. The latter can thus be handled

in a quadratic penalty fashion; since, as also done in [103], the constraint $\nabla_w g(w, \alpha) = 0$ can be seen as a generic differentiable equality constraint and moved into the penalty function, we can associate problem (4.7) with the overall penalty function

$$\phi_\gamma(w, \alpha, \beta) = f(w,\ \alpha) + \frac{\gamma}{2}\|h(w,\ \alpha)\|^2 + \frac{\gamma}{2}\|\alpha - \beta\|^2. \tag{4.8}$$

Indeed, function $\phi_\gamma(w, \alpha, \beta)$ can be used in a sequential penalty type framework (see Algorithm 4). In particular, given a sequence of penalty parameters $\{\gamma_k\}$ such that $\gamma_k \to \infty$ and a sequence $\{\epsilon_k\}$ such that $\epsilon_k \to 0$, a sequence $\{w_k, \alpha_k, \beta_k\}$ can be constructed such that, for all k, the optimization problem

$$\begin{aligned} &\min_{w,\alpha,\beta}\ \phi_{\gamma_k}(w, \alpha, \beta) \\ &\text{s.t. } \|\beta\|_0 \leq C \end{aligned} \tag{4.9}$$

is approximately solved up to stationarity, i.e., the following conditions hold:

$$\|\nabla_w \phi_{\gamma_k}(w_k, \alpha_k, \beta_k)\| + \|\nabla_\alpha \phi_{\gamma_k}(w_k, \alpha_k, \beta_k)\| \leq \epsilon_k \tag{4.10}$$

$$\beta^k \in \underset{\|\beta\|_0 \leq C}{\arg\min}\ \ \phi_{\gamma_k}(w^k, \alpha^k, \beta) \tag{4.11}$$

We shall note that, by the definition of $\phi_\gamma(w, \alpha, \beta)$, condition (4.11) is clearly equivalent to

$$\beta^k \in \underset{\|\beta\|_0 \leq C}{\arg\min}\ \ \|\beta - \alpha^k\|^2. \tag{4.12}$$

Moreover, it is important to highlight that the solution of Problem (4.12) can be computed in closed form [98], setting

$$\beta_i^k = \begin{cases} \alpha_i^k & \text{if } i \in G, \\ 0 & \text{otherwise,} \end{cases} \tag{4.13}$$

where G denotes a set of C indices corresponding to the greatest elements of α^k in absolute value. We denote by $\mathcal{G}(\alpha^k)$ the set of all the possible such sets.

Now, in order to obtain a sequence of points satisfying conditions (4.10) - (4.11), we can resort to an alternate minimization scheme, as done in Penalty Decomposition approaches [98]. The instructions for this inner solver for subproblems of the form (4.9) are reported in Algorithm 5. Note that, in order to obtain convergence guarantees without convexity assumptions, a

Algorithm 4: Penalty Decomposition approach for problem (4.7)

Data: $f,\ g,\ \alpha_0,\ w_0,\ \beta_0,\ C > 0,\ N > 0,\ \gamma_0,\ \Delta > 1,\ \epsilon_0,\ \delta < 1$
Result: $w_N,\ \alpha_N$
$\phi_{\gamma_k}(w,\ \alpha,\ \beta) \leftarrow f(w,\ \alpha) + \frac{\gamma_k}{2}\|\nabla_w g(w,\ \alpha)\|^2 + \frac{\gamma_k}{2}\|\alpha - \beta\|^2$
for $k = 0,\ \dots\ N$ **do**
  Find $(\alpha_{k+1},\ w_{k+1},\ \beta_{k+1})$ to satisfy (4.10) and (4.11)
  $\gamma_{k+1} \leftarrow \gamma_k \cdot \Delta$
  $\epsilon_{k+1} \leftarrow \epsilon_k \cdot \delta$

proximal point term [56] is added in the update subproblem for the w and α variables,. Interestingly, we do not need to modify the β-update subproblem to obtain a convergent algorithm, as we will show later in this work, so that we are still allowed to use the closed form formula (4.13) for the projection onto the sparse set.

Algorithm 5: Alternate minimization for Problem (4.9)

Data: $\phi_{\gamma_k},\ \alpha_k,\ w_k,\ \beta_k,\ \epsilon_k > 0, \rho > 0$
Result: $w_{k+1},\ \alpha_{k+1},\ \beta_{k+1}$
$\ell \leftarrow 0$
$\hat{\alpha}_\ell,\ \hat{w}_\ell,\ \hat{\beta}_\ell \leftarrow \alpha_k,\ w_k,\ \beta_k$
while $\|\nabla_w \phi_{\gamma_k}(\hat{\alpha}_\ell, \hat{w}_\ell, \hat{\beta}_\ell)\|^2 + \|\nabla_\alpha \phi_{\gamma_k}(\hat{\alpha}_\ell, \hat{w}_\ell, \hat{\beta}_\ell)\|^2 > \epsilon_k$ **do**
  $\hat{w}_{\ell+1} \leftarrow \arg\min_w\ \phi_{\gamma_k}(w,\ \hat{\alpha}_\ell,\ \hat{\beta}_\ell) + \rho\|w - \hat{w}_\ell\|^2$
  $\hat{\alpha}_{\ell+1} \leftarrow \arg\min_\alpha\ \phi_{\gamma_k}(\hat{w}_{\ell+1},\ \alpha,\ \hat{\beta}_\ell) + \rho\|\alpha - \hat{\alpha}_\ell\|^2$
  $\hat{\beta}_{\ell+1} \leftarrow \arg\min_{\beta\,:\,\|\beta\|_0 \le C}\ \|\hat{\alpha}_{\ell+1} - \beta\|^2$
  $\ell \leftarrow \ell + 1$
$\alpha_{k+1},\ w_{k+1},\ \beta_{k+1} \leftarrow \hat{\alpha}_\ell,\ \hat{w}_\ell,\ \hat{\beta}_\ell$

The combination of Algorithms 4 and 5 provides an effective method to tackle problem (4.5). In the next section, we provide a theoretical analysis justifying this claim under suitable assumptions.

4.3.2 Convergence analysis

In this section, we carry out the convergence analysis of the proposed approach. Firstly, we state a reasonable assumption that we will make throughout the rest of the chapter.

Assumption 2. The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ has bounded level sets upon its domain, i.e., $\{(w, \alpha) \mid f(w, \alpha) \leq \eta\}$ is bounded for any $\eta \in \mathbb{R}$.

Next, we begin the discussion recalling some basic concepts in sparse optimization.

The *support* of variable α associated with the cardinality constraint is the index set of nonzero elements of α and is denoted by

$$I_1(\alpha) = \{i \mid \alpha_i \neq 0\}.$$

Moreover, we shall introduce a further concept.

Definition 3 ([12])**.** Let $\alpha \in \mathbb{R}^m$ such that $\|\alpha\|_0 \leq C$. A *super support set* J at α is an index set such that

- $J \supseteq I_1(\alpha)$;
- $|J| = C$.

The set of all support sets at α is denoted by $\mathcal{J}(\alpha)$.

Now, we are able to introduce the necessary optimality conditions for problem (4.5). Firstly, we recall the *Lu-Zhang conditions* for the single level problem (4.6).

Definition 4 ([81,82,98])**.** A point $(\bar{w}, \bar{\alpha})$ satisfies Lu-Zhang conditions for problem (4.6) if there exist multipliers $\lambda \in \mathbb{R}^n$, $\xi \in \mathbb{R}^m$ and $J \in \mathcal{J}(\bar{\alpha})$ such that

$$\nabla_w f(\bar{w}, \bar{\alpha}) + \sum_{i=1}^{n} \lambda_i \nabla_w h_i(\bar{w}, \bar{\alpha}) = 0, \tag{4.14}$$

$$\nabla_\alpha f(\bar{w}, \bar{\alpha}) + \sum_{i=1}^{n} \lambda_i \nabla_\alpha h_i(\bar{w}, \bar{\alpha}) + \xi = 0, \tag{4.15}$$

$$\xi_i = 0 \ \text{ for all } i \in J. \tag{4.16}$$

Next, we introduce a suitable constraint qualification for problem (4.6).

Definition 5 ([98]). Let $(\bar{w}, \bar{\alpha}) \in \mathbb{R}^n \times \mathbb{R}^m$. Let $\nabla h_1(\bar{w}, \bar{\alpha}), \ldots, \nabla h_n(\bar{w}, \bar{\alpha})$ denote the gradients (w.r.t. both w and α) of the components of $h(\bar{w}, \bar{\alpha})$, i.e.,

$$\nabla h_i(\bar{w}, \bar{\alpha}) = \left(\frac{\partial g(\bar{w}, \bar{\alpha})}{\partial w_1 \partial w_i}, \ldots, \frac{\partial g(\bar{w}, \bar{\alpha})}{\partial w_n \partial w_i}, \frac{\partial g(\bar{w}, \bar{\alpha})}{\partial \alpha_1 \partial w_i}, \ldots, \frac{\partial g(\bar{w}, \bar{\alpha})}{\partial \alpha_m \partial w_i} \right)^T.$$

Moreover, let $e_j \in \mathbb{R}^{n+m}$, $j = 1, \ldots, m$, be the element of the canonical bases in $\mathbb{R}^{n+m}$ corresponding to variable α_j. Then, $(\bar{w}, \bar{\alpha})$ satisfies the *extended Robinson condition* for a set $G \in \mathcal{G}(\bar{\alpha})$, with $\{1 \ldots, m\} \setminus G = \{j_1, \ldots, j_{m-C}\}$, if the vectors $\nabla h_1(\bar{w}, \bar{\alpha}), \ldots, \nabla h_n(\bar{w}, \bar{\alpha}), e_{j_1}, \ldots, e_{j_{m-C}}$ are linearly independent.

Basically, the extended Robinson condition for a set $G \in \mathcal{G}(\bar{\alpha})$ is the classical linear independence constraint qualification (LICQ) for the problem where the constraints $\alpha_i = 0$ for all $i \notin G$ have been added. Also note that, at a feasible point, $\mathcal{G}(\bar{\alpha}) = \mathcal{J}(\bar{\alpha})$, i.e., G is a super support set for $\bar{\alpha}$.

We are finally able to state the necessary optimality condition for the bilevel problem (4.5).

Proposition 4. Let $(\bar{w}, \bar{\alpha})$ be an optimal solution of problem (4.5). If $(\bar{w}, \bar{\alpha})$ satisfies the (extended) Robinson condition for some $J \in \mathcal{J}(\bar{\alpha})$, then it also satisfies Lu-Zhang conditions for problem (4.6).

Proof. An optimal solution of the bilevel problem (4.5) is certainly a solution of the single level problem (4.6) by Assumption 1. Now, from [98, Theorem 2.1], since $(\bar{w}, \bar{\alpha})$ satisfies the Robinson condition for some super support set $J \in \mathcal{J}(\bar{\alpha})$, we know that $(\bar{w}, \bar{\alpha})$ satisfies the Lu-Zhang conditions for problem (4.6) with the super support set J. □

We are now ready to turn to the properties of the proposed algorithm. We begin the analysis proving that the three-blocks alternate minimization scheme (Algorithm 5) indeed produces an approximate solution to the penalty subproblem in finite time. In order to do that, however, we preliminarily have to prove a nice property of ϕ_{γ_k}.

Lemma 3. The penalty function $\phi_\gamma(w, \alpha, \beta)$ has bounded level sets for any $\gamma > 0$.

Proof. Consider any $\eta \in \mathbb{R}$. From Assumption 2, the level set $L_f(\eta) = \{(w, \alpha) | f(w, \alpha) \leq \eta\}$ is bounded. Let us consider $L_{\phi_\gamma}(\eta) = \{(w, \alpha, \beta) \mid \phi_\gamma(w, \alpha, \beta) \leq \eta\}$ for any $\gamma \geq 0$.

Assume by contradiction that $L_{\phi_\gamma}(\eta)$ is not bounded, i.e., there exists $\{w_t, \alpha_t, \beta_t\}$ such that $(w_t, \alpha_t, \beta_t) \in L_{\phi_\gamma}(\eta)$ for all t and $\|(w_t, \alpha_t, \beta_t)\| \to \infty$. Then, either $\|(w_t, \alpha_t)\| \to \infty$ or $\|\beta_t\| \to \infty$.

If $\|(w_t, \alpha_t)\| \to \infty$, we have $f(w_t, \alpha_t) > \eta$ for t sufficiently large, being $L_f(\eta)$ bounded. But then, from the definition of $\phi_\gamma(w, \alpha, \beta)$, we have for t sufficiently large $\phi_\gamma(w_t, \alpha_t, \beta_t) \geq f(w_t, \alpha_t) > \eta$, which contradicts $\{w_t, \alpha_t, \beta_t\} \subseteq L_{\phi_\gamma}(\eta)$.

Thus, $\|\beta_t\| \to \infty$ while $\|(w_t, \alpha_t)\|$ stays bounded. However,

$$\phi_\gamma(w_t, \alpha_t, \beta_t) = f(w_t, \alpha_t) + \frac{\gamma}{2}\left(\|\alpha_t - \beta_t\|^2 + \|h(w_t, \alpha_t)\|^2\right) > \eta$$

for t sufficiently large, as $\|\alpha_t - \beta_t\|^2 \to \infty$ and $\|h(w_t, \alpha_t)\|^2 \geq 0$ and f is bounded having compact level sets. This again is a contradiction, which completes the proof. □

Proposition 5. Algorithm 5 cannot infinitely cycle and produces in a finite number of iterations a point $(w^{k+1}, \alpha^{k+1}, \beta^{k+1})$ such that conditions (4.10) and (4.11) are satisfied.

Proof. Condition (4.11) is trivially satisfied for every ℓ from the instructions of the algorithm. Now, assume by contradiction that the sequence $\{\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell\}$ is infinite, i.e., condition (4.10) is never satisfied by $(\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell)$. From the instructions of the algorithm we have

$$\begin{aligned}
\phi_{\gamma_k}(\hat{w}_{\ell+1}, \hat{\alpha}_{\ell+1}, \hat{\beta}_{\ell+1}) &\leq \phi_{\gamma_k}(\hat{w}_{\ell+1}, \hat{\alpha}_{\ell+1}, \hat{\beta}_\ell) \\
&\leq \phi_{\gamma_k}(\hat{w}_{\ell+1}, \hat{\alpha}_\ell, \hat{\beta}_\ell) - \rho\|\hat{\alpha}_{\ell+1} - \hat{\alpha}_\ell\|^2 \\
&\leq \phi_{\gamma_k}(\hat{w}_{\ell+1}, \hat{\alpha}_\ell, \hat{\beta}_\ell) \\
&\leq \phi_{\gamma_k}(\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell) - \rho\|\hat{w}_{\ell+1} - \hat{w}_\ell\|^2 \\
&\leq \ldots \leq \phi_{\gamma_k}(\hat{w}_0, \hat{\alpha}_0, \hat{\beta}_0),
\end{aligned} \tag{4.17}$$

hence the sequence $\{\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell\}$ belongs to the level set $L_{\phi_{\gamma_k}}(\phi_{\gamma_k}(\hat{w}_0, \hat{\alpha}_0, \hat{\beta}_0))$, which is bounded by Lemma 3. Thus, there exists $T \subseteq \{0, 1, \ldots\}$ such that

$$\lim_{\substack{\ell \to \infty \\ \ell \in T}} (\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell) = (\hat{w}, \hat{\alpha}, \hat{\beta}).$$

Moreover, since the sequence $\{\phi_{\gamma_k}(\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell)\}$ is monotone decreasing, it has limit $\Phi^\star$, which is finite being ϕ_{γ_k} bounded below, i.e., recalling the

continuity of ϕ_{γ_k}, we have

$$\Phi^\star = \lim_{\ell\to\infty} \phi_{\gamma_k}(\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell) = \lim_{\substack{\ell\to\infty \\ \ell\in T}} \phi_{\gamma_k}(\hat{w}_\ell, \hat{\alpha}_\ell, \hat{\beta}_\ell) = \phi_{\gamma_k}(\hat{w}, \hat{\alpha}, \hat{\beta}).$$

Thus, taking the limits in (4.17) and by the squeeze theorem we have

$$\lim_{\ell\to\infty} \|(\hat{w}_{\ell+1} - \hat{w}_\ell, \hat{\alpha}_{\ell+1} - \hat{\alpha}_\ell)\|^2 = 0.$$

From the instructions of the algorithm, recalling the first order optimality conditions, we also know that, for all $\ell \in T$, it holds

$$\nabla_w \phi_{\gamma_k}(\hat{w}_{\ell+1}, \hat{\alpha}_\ell, \hat{\beta}_\ell) = 0, \qquad \nabla_\alpha \phi_{\gamma_k}(\hat{w}_{\ell+1}, \hat{\alpha}_{\ell+1}, \hat{\beta}_\ell) = 0,$$

therefore, since $\|\hat{w}_{\ell+1} - \hat{w}_\ell\| \to 0$ and $\|\hat{\alpha}_{\ell+1} - \hat{\alpha}_\ell\| \to 0$, taking again the limits for $\ell \to \infty$, $\ell \in T$, we get by the continuity of the gradient

$$\nabla_w \phi_{\gamma_k}(\hat{w}, \hat{\alpha}, \hat{\beta}) = 0, \qquad \nabla_\alpha \phi_{\gamma_k}(\hat{w}, \hat{\alpha}, \hat{\beta}) = 0.$$

The last result implies that, for $\ell \in T$ sufficiently large, we have

$$\|\nabla_w \phi_{\gamma_k}(\hat{w}, \hat{\alpha}, \hat{\beta})\| + \|\nabla_\alpha \phi_{\gamma_k}(\hat{w}, \hat{\alpha}, \hat{\beta})\| \leq \epsilon_k,$$

which is a contradiction. □

The above proposition guarantees that, for any $\gamma_k > 0$, the Alternate Minimization scheme actually provides a point satisfying the conditions required at each iteration of Algorithm 4.

We are finally able to state the main convergence result of this chapter. The proof partly follows the the reasoning of [98, Theorem 4.3]. However, we still report it here not only for the sake of completeness, but also because of differences in Algorithm 4 w.r.t. the original PD scheme that lead to changes in the proof. In fact, in [98] the knowledge of a feasible point is (unreasonably) assumed, which enables to design a convergent scheme under weaker regularity assumptions on the constraints.

Proposition 6. Let $\{\gamma_k\}$, $\{\epsilon_k\}$ and $\{w_k, \alpha_k, \beta_k\}$ be sequences such that $\gamma_k \to \infty$, $\epsilon_k \to 0$ and for (w_k, α_k, β_k) satisfies conditions (4.11) and (4.10). If $(\bar{w}, \bar{\alpha}, \bar{\beta})$ is an accumulation point of $\{w_k, \alpha_k, \beta_k\}$, then one of the two following conditions hold:

1. $(\bar{w}, \bar{\alpha}, \bar{\beta})$ is feasible for problem (4.7), $(\bar{w}, \bar{\alpha})$ is a feasible solution of Problem (4.6) and satisfies Lu-Zhang conditions;

2. there exists $G \in \mathcal{G}(\bar{\alpha})$ such that the extended Robinson conditions is not satisfied by $(\bar{w}, \bar{\alpha})$ for G.

Proof. Let $K \subseteq \{0, 1, \ldots\}$ be a infinite subsequence of iterations such that

$$\lim_{\substack{k \to \infty \\ k \in K}} (w_k, \alpha_k, \beta_k) = (\bar{w}, \bar{\alpha}, \bar{\beta}).$$

If there exists $G \in \mathcal{G}(\bar{\alpha})$ such that the Robinson condition is not satisfied, we are in case 2 and our assertion is true. Thus, let us now assume that the Robinson condition is satisfied for all $G \in \mathcal{G}(\bar{\alpha})$. From the instructions of the algorithm, we have that for any k

$$\begin{aligned} &\left\| \nabla_w f(w_{k+1}, \alpha_{k+1}) + \gamma_k \sum_{i=1}^{n} h_i(w_{k+1}, \alpha_{k+1}) \nabla_w h_i(w_{k+1}, \alpha_{k+1}) \right\| \\ + &\left\| \nabla_\alpha f(w_{k+1}, \alpha_{k+1}) + \gamma_k(\alpha_{k+1} - \beta_{k+1}) + \gamma_k \sum_{i=1}^{n} h_i(w_{k+1}, \alpha_{k+1}) \nabla_\alpha h_i(w_{k+1}, \alpha_{k+1}) \right\| \leq \epsilon_k. \end{aligned} \tag{4.18}$$

Dividing both sides by γ_k we get

$$\begin{aligned} &\left\| \frac{\nabla_w f(w_{k+1}, \alpha_{k+1})}{\gamma_k} + \sum_{i=1}^{n} h_i(w_{k+1}, \alpha_{k+1}) \nabla_w h_i(w_{k+1}, \alpha_{k+1}) \right\| \\ + &\left\| \frac{\nabla_\alpha f(w_{k+1}, \alpha_{k+1})}{\gamma_k} + (\alpha_{k+1} - \beta_{k+1}) + \sum_{i=1}^{n} h_i(w_{k+1}, \alpha_{k+1}) \nabla_\alpha h_i(w_{k+1}, \alpha_{k+1}) \right\| \leq \frac{\epsilon_k}{\gamma_k}. \end{aligned}$$

Let $K_1 \subseteq K$ be a further subsequence such that $I_1(\beta_{k+1}) = \mathcal{G} \in \mathcal{G}(\alpha_{k+1})$ for all $k \in K_1$; such a subsequence exists since the number of possible index sets is finite. Then we can define $\tilde{d}_{k+1} = \alpha^{k+1} - \beta^{k+1}$, where, by (4.13), $\tilde{d}_j^{k+1} = 0$ for all $j \in \mathcal{G}$ and $\tilde{d}_{k+1,j} = \alpha_{k+1,j}$ if $j \notin \mathcal{G}$. We thus have, for all $k \in K_1$,

$$\begin{aligned} &\left\| \frac{\nabla_w f(w_{k+1}, \alpha_{k+1})}{\gamma_k} + \sum_{i=1}^{n} h_i(w_{k+1}, \alpha_{k+1}) \nabla_w h_i(w_{k+1}, \alpha_{k+1}) \right\| \\ + &\left\| \frac{\nabla_\alpha f(w_{k+1}, \alpha_{k+1})}{\gamma_k} + \tilde{d}^{k+1} + \sum_{i=1}^{n} h_i(w_{k+1}, \alpha_{k+1}) \nabla_\alpha h_i(w_{k+1}, \alpha_{k+1}) \right\| \leq \frac{\epsilon_k}{\gamma_k}. \end{aligned}$$

Recalling that $\{\epsilon_k\}$ is bounded, $\gamma_k \to \infty$, $f, h, \nabla f$, and ∇h are continuous functions, we have taking the limits for $k \in K_1$, $k \to \infty$:

$$\sum_{i=1}^{n} h_i(\bar{w}, \bar{\alpha}) \nabla_w h_i(\bar{w}, \bar{\alpha}) = 0$$

and

$$\tilde{d} + \sum_{i=1}^{n} h_i(\bar{w}, \bar{\alpha}) \nabla_\alpha h_i(\bar{w}, \bar{\alpha}) = 0,$$

where $\tilde{d} = \bar{\alpha} - \bar{\beta}$, i.e., $\tilde{d}_j = \bar{\alpha}_j$ if $j \notin \mathcal{G}$ and $\tilde{d}_j = 0$ if $j \in \mathcal{G}$. Putting everything together, we obtain

$$\sum_{i \in \{1,\dots,m\} \setminus \mathcal{G}} e_i \bar{\alpha}_i + \sum_{i=1}^{n} h_i(\bar{w}, \bar{\alpha}) \nabla h_i(\bar{w}, \bar{\alpha}) = 0,$$

which is only possible if $h(\bar{w}, \bar{\alpha}) = 0$ and $\bar{\alpha}_i = 0$ for all $i \in \{1, \dots, m\} \setminus \mathcal{G}$ (i.e., $\bar{\alpha} - \bar{\beta} = 0$), since the extended Robinson condition holds for $\mathcal{G}$ and thus $\nabla h_1(\bar{w}, \bar{\alpha}), \dots, \nabla h_n(\bar{w}, \bar{\alpha})$ and e_i for $i \in \{1, \dots, m\} \setminus \mathcal{G}$ are linearly independent.

The point $(\bar{w}, \bar{\alpha}, \bar{\beta})$ is thus feasible for problem (4.7) and thus $(\bar{w}, \bar{\alpha})$ is feasible for problem (4.6).

In order to prove stationarity condition for the limit points, let us define $\lambda_i^k = \gamma_k h_i(w_{k+1}, \alpha_{k+1})$ for all $i = 1, \dots, n$ and $\xi_k = \gamma_k(\alpha_{k+1} - \beta_{k+1})$.

We show that $\{\lambda^k, \xi^k\}_K$ is a bounded sequence. By contradiction, assume $\|\lambda^k, \xi^k\| \to \infty$; let us define $(\hat{\lambda}^k, \hat{\xi}^k) = (\lambda^k, \xi^k)/\|\lambda^k, \xi^k\|$ for all $k \in K$. The sequence $\{\hat{\lambda}^k, \hat{\xi}^k\}$ is bounded by definition, since $\|\hat{\lambda}^k, \hat{\xi}^k\| = 1$ for all $k \in K$.

Dividing by $\|\lambda^k, \xi^k\|$ both sides of equation (4.18) and taking the limit for $k \in K$ (along a suitable subsequence where $\hat{\lambda}^k \to \bar{\lambda}$ and $\hat{\xi}^k \to \bar{\xi}$ if needed), $k \to \infty$, recalling $\|\lambda^k, \xi^k\| \to \infty$ and the continuity of ∇f, and ∇h, we get

$$\sum_{i=1}^{n} \bar{\lambda}_i \nabla_w h_i(\bar{w}, \bar{\alpha}) = 0$$

and

$$\bar{\xi} + \sum_{i=1}^{n} \bar{\lambda}_i \nabla_\alpha h_i(\bar{w}, \bar{\alpha}) = 0.$$

Now, taking again if needed a subsequence such that $I_1(\beta_{k+1}) = \mathcal{G} \in \mathcal{G}(\alpha_{k+1})$ for all k, we have $\xi_i^{k+1} = \gamma_k(\alpha_{k+1,i} - \beta_{k+1,i}) = 0$ for all k for all $i \in \mathcal{G}$ and thus $\bar{\xi}_i = 0$ for all $i \in \mathcal{G}$.

Since the extended Robinson condition holds at $(\bar{w}, \bar{\alpha})$ for $\mathcal{G}$ and $\|\bar{\lambda}, \bar{\xi}\| = 1$, this is absurd, by similar reasonings as above.

Thus $\{\lambda^{k+1}, \xi^{k+1}\}_K$ is a bounded sequence. Taking a subsequence if necessary, let $(\lambda^\star, \xi^\star)$ be an accumulation point of $\{\lambda^{k+1}, \xi^{k+1}\}_K$; taking the limits in (4.18), we get

$$\nabla_w f(\bar{w}, \bar{\alpha}) + \sum_{i=1}^{n} \lambda_i^\star \nabla_w h_i(\bar{x}) = 0$$

and

$$\nabla_\alpha f(\bar{w}, \bar{\alpha}) + \xi^\star + \sum_{i=1}^{n} \lambda_i^\star \nabla_\alpha h_i(\bar{x}) = 0$$

with, by analogous reasoning as before, $\xi_i^\star = 0$ for all $i \in \mathcal{G} \in \mathcal{G}(\bar{\alpha}) = \mathcal{J}(\bar{\alpha})$, where the last inequality comes from the feasibility of the limit point. This completes the proof. □

4.4 Experiments

This section is devoted to reporting and discussing the proposed methods' proof of concept results. Specifically, in the following, we report: results based on quadratic functions that satisfy all the hypotheses done in theory and results obtained training a ResNet20 on CIFAR10.

4.4.1 Convex case

For the convex case, as case of study, we consider problems of the form

$$\begin{aligned} \min_{x,\bar{y}} \; & f(x, \bar{y}) = \frac{1}{2}(1-x)^T Q_1 (1-x) + \lambda_1 x^T \bar{y} \\ \text{s.t. } & \bar{y} \in \arg\min_y g(x, y) = \frac{1}{2} y^T Q_2 y - \lambda_2 x^T y \\ & \|x\|_0 \leq C \end{aligned} \tag{4.19}$$

where $Q \in \mathbb{R}^{N \times N}$ are positive semi-definite matrices, $\lambda \in \mathbb{R}$, x and y are vectors in $\mathbb{R}^N$.

To solve Problem (4.19) we use both the alternating minimization approach and our penalty decomposition algorithm. Specifically, in Table 4.1, since the matrices Q and the parameters λ are sampled, respectively, from an uniform distribution in $[-1, 1]$ and a random integer from 0 to 15, we report the number of times the algorithm converge to a solution (out of five attempts).

Even if a more extended set of experiments is needed, Table 4.1 confirms the intuition reported in the first part of this Chapter: our proposed algorithm is more stable than the Alternating Minimization approach proposed in DARTS [92].

Finally, to strengthen the soundness of our approach we report the value of the function f (lower is better) for the case $N = 5$ and $C = 5$ in Table 4.2.

Parameter	Alternating Minimization	Penalty Decomposition
$N = 10$, $C = 1$	2	4
$N = 10$, $C = 3$	0	4
$N = 10$, $C = 5$	1	4
$N = 5$, $C = 3$	1	4

Table 4.1: Number of convergent experiments (out of five attempts) for the tested algorithms. Higher number means a more stable algorithm.

Alternating Minimization	Penalty Decomposition
122.9	21.0
62.6	9
318.8	12.8
0.3	3×10^{-5}
35.2	7.4

Table 4.2: Comparison between the value of the outer objective functions f (lower is better) for both the methods. We run these experiments using $N = 5$ and $C = 5$.

We use this setting because, from our experiments, removing the cardinality constraint helps the Alternating Minimization's stability.

4.4.2 Non-convex case

We focus this part on preliminary experiments on ResNet architecture [63] trained on CIFAR-10 [79]. Moreover, we compare three different approaches against our proposal: Alternating Minimization, Random Pruning, and Channel pruning based on the norm of the filter.

We implement Random pruning as a post-training pruning. More specifically, after a full training of the model, we randomly remove a certain percentage of filters and refine the pruned architecture. It is important to highlight that, as reported in [89, 93], random pruning is an effective and robust baseline for network compression tasks.

For the Channel pruning approach [88, 142], we use a strategy similar to the one followed for Random pruning: after a full network training, we

Method	**Pruned Channels**	**Test Accuracy** (↑)
Baseline	0 %	90.44
Our proposal	75 %	**82.16** ± 0.68
Alternating Minimization		78.86 ± 0.82
Random Pruning		81.40 ± 0.16
Norm Pruning		81.41 ± 0.24

Table 4.3: ResNet20 channel pruning on CIFAR-10, we run each experiments three times with different seed. The first row refers to the full ResNet-20's accuracy while the others refer to the accuracy reached removing 75% of the channel filters.

remove a percentage of filters following a ranking based on the filter's weight norm.

Finally, we follow the Alternating Minimization approach proposed in [92] for DARTS.

In Table 4.3 we report the results obtained pruning 75% of ResNet-20's convolutional filters. Specifically, we run each algorithm with different seeds three times and report the mean and standard deviation of the obtained results. As it is possible to see, our proposal obtains better results compared to the other considered approaches.

4.5 Final considerations

Bilevel optimization formulation, as emerged from a recent stream of work, can be used to model the problem of neural network compression. In this Chapter, we discussed a general penalty decomposition approach suitable to solve such problems with an additional ℓ_0-norm constraint. In contrast to our proposed algorithm, we prove that the popular alternate minimization approach followed by Liu et al. [92] does not have theoretical convergence results for such bilevel problems. In addition to this theoretical analysis, we also show some preliminary empirical results to prove our algorithm's soundness.

Chapter 5

Normalization Free ResNet-like models

Batch Normalization is an essential component of all state-of-the-art neural networks architectures. However, since it introduces many practical issues, much recent research has been devoted to designing normalization-free architectures. In this chapter, we show that weight initialization is key to train ResNet-like normalization-free networks. In particular, we propose a slight modification to the summation operation of a block output to the skip-connection branch, so that the whole network is correctly initialized. We show that this modified architecture achieves competitive results on CIFAR-10, CIFAR-100, and ImageNet without further regularization nor algorithmic modifications.

5.1 Preamble

Batch normalization [71], in conjunction with skip connections [63, 64], has allowed the training of significantly deeper networks. Nowadays, most state-of-the-art architectures use batch normalization since it yields well behaved gradients (removing *mean-shift*, avoiding *vanishing* or *exploding* gradients) and, moreover, it introduces a regularizing effect [67,99]. Anyhow, while skip connections can be easily implemented and integrated in any network architecture without major drawbacks, batch normalization poses a few practical challenges. As already observed and discussed by [21, 22], batch normalization adds a significant memory overhead, introduces a discrepancy between training and inference time, has a tricky implementation in distributed training, performs poorly with small batch sizes [147] and breaks the independence between training examples in a minibatch, which can be extremely harmful for some learning tasks [86, 96].

In this Chapter, we propose a simple modification of the residual block that, together with a careful initialization, allows to train deep residual networks without any normalization layer. With such scheme, standardization layers or algorithmic modifications are not required.

The main contributions described in this Chapter are:

- We show that while the proposals of [21, 22] enjoy a perfect forward variance (as already noted by [22]), it puts the network in a regime of *exploding gradients*. This is shown by looking at the variance of the derivatives of the loss with respect to the feature maps at different depths;

- We propose a simple modification of the residual layer and then develop a suitable initialization scheme building on the work of [62];

- We show that the proposed architecture achieves competitive results on CIFAR-10 [78], CIFAR-100 [79] and ImageNet [38] which we consider evidence supporting our theoretical claims.

5.2 Preliminary

In this part of the Chapter, we report the operations performed inside a Batch Normalization module [71]. Specifically, let $x \in \mathbb{R}^{N \times C \times H \times W}$ a tensor

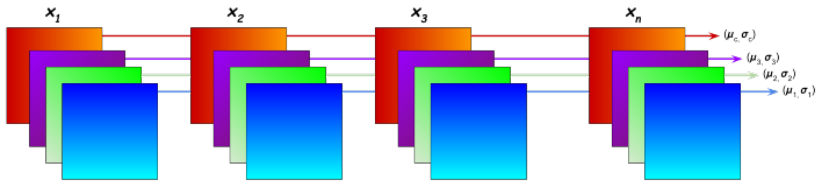


Figure 5.1: A sketch of how Batch Normalization computes mean and variance for each batch. Image adapted from "becominghuman.ai".

modeling the input of Batch Normalization where N is the number of examples inside the batch, C is the number of feature maps, and H, W are the feature map spatial dimensions. Batch Normalization performs the following two consecutive operations

$$\hat{x} = \frac{x - \mu}{\sigma^2}$$

and

$$y = \alpha\hat{x} + \beta.$$

In these operations, it is important to highlight two parts. On the one hand, the coefficients α and β model a linear transformation on the normalized (zero mean and unitary variance) input tensor $\hat{x}$. These two parameters are learned during optimization, like standard network weights. While in the other hand, the vectors μ and σ^2 in $\mathbb{R}^C$ represent the channel-wise mean and the variance across the whole batch (Figure 5.1 highlights how the vectors are computed). More specifically, we have

$$\mu_c = \frac{1}{NHW}\sum_{i=0}^{N}\sum_{j=0}^{H}\sum_{k=0}^{W} x[i,c,j,k] \quad \forall \ c \in [0,\dots C]$$

and

$$\sigma_c^2 = \frac{1}{NHW}\sum_{i=0}^{N}\sum_{j=0}^{H}\sum_{k=0}^{W}(x[i,c,j,k] - \mu_c)^2 \quad \forall \ c \in [0,\dots C].$$

To conclude, since the training of a neural network is performed using batches of data, the actual mean and variance used to normalize the input tensor is defined averaging multiple batch statistics.

5.3 Related work

As highlighted in a number of recent studies [5, 35, 59], weights initialization is crucial to make deep networks work in absence of batch normalization. In particular, the weights at the beginning of the training process should be set so as to correctly propagate the forward activation and the backward gradients signal in terms of mean and variance.

This kind of analysis was first proposed by [53] and later extended by [62]. These seminal studies considered architectures composed by a sequence of convolutions and Rectified Linear Units (ReLU), which mainly differ from modern ResNet architectures for the absence of skip-connections.

The analysis in [62] investigates the variance of each response layer ℓ (*forward variance*). More specifically, considering the response of each layer ℓ:

$$z_\ell = g(x_{\ell-1}), \qquad x_\ell = W_\ell z_\ell,$$

where x is a $k^2c \times 1$ vector that represents co-located $k \times k$ pixels in c input channels, W_ℓ is a $d \times n$ matrix where d is the number of filters and $g(\cdot)$ is a nonlinear activation function. In the following, we will consider the classical ReLU, $g(z) = \max(0, z)$.

Formally, let us consider normally distributed input data

$$x_{\ell-1} \sim \mathcal{N}(0, \sigma^2).$$

It is well known that with this particular activation we get a Rectified Normal Distribution [6, 133] with central moments:

$$\mu_g = \mathbb{E}[g(z_l)] = \frac{\sigma}{\sqrt{2\pi}}, \qquad \sigma_g^2 = \mathrm{Var}[g(z_\ell)] = \frac{\sigma^2}{2} - \frac{\sigma^2}{2\pi}.$$

From the basic properties of variance we also have

$$\mathbb{E}[g(z_\ell)^2] = \mu_g^2 + \sigma_g^2 = \frac{\sigma^2}{2}. \tag{5.1}$$

Making the assumption that the weights W (we omit for simplicity the dependency on layer ℓ) at a network's layer ℓ are i.i.d. with zero mean ($\mu_W = 0$) and that have zero correlation with the input (hence $\mathrm{Var}[Wg(x_{\ell-1})] = \sigma_W^2\sigma_g^2$, putting $Var[W] = \sigma_W^2$), we obtain for the output elements that

$$\mathbb{E}[x_\ell] = n_{\text{in}}\mu_g\mu_W = 0$$

and

$$\begin{aligned}\mathrm{Var}[x_\ell] &= n_{\mathrm{in}}[\mathbb{E}[W^2 z_\ell^2] - (\mathbb{E}[W z_\ell])^2] \\ &= n_{\mathrm{in}}[\mathbb{E}[W^2]\mathbb{E}[z_\ell^2] - (\mathbb{E}[W]\mathbb{E}[z_\ell])^2] \\ &= n_{\mathrm{in}}[(\mu_W^2 + \sigma_W^2)(\mu_g^2 + \sigma_g^2) - \mu_W^2 \mu_g^2] \\ &= n_{\mathrm{in}}[\sigma_g^2(\mu_W^2 + \sigma_W^2) + \sigma_W^2 \mu_g^2] \\ &= n_{\mathrm{in}}[\sigma_W^2(\mu_g^2 + \sigma_g^2)].\end{aligned} \tag{5.2}$$

where $n_{\mathrm{in}} = k^2 c$ with k the filter dimension and c the number of input channels (*fan in*). Hence if input has unit variance ($\sigma^2 = 1$) we obtain output unit variance by initializing W in such a way that

$$\mathrm{Var}[W] = \frac{2}{n_{\mathrm{in}}}. \tag{5.3}$$

Similarly we can perform the analysis w.r.t. the gradients signal.

For back-propagation, we can also write

$$\frac{\partial \mathcal{L}}{\partial z_\ell} = \hat{W} \frac{\partial \mathcal{L}}{\partial x_\ell},$$

where $\mathcal{L}$ is the loss function and $\hat{W}$ is a suitable rearrangement of W. If weights W are initialized with zero mean from a symmetric distribution, $\frac{\partial \mathcal{L}}{\partial z_\ell}$ will also have zero mean. We can assume $\frac{\partial \mathcal{L}}{\partial x_\ell}$ and $\hat{W}$ to be uncorrelated.

In addition,

$$\frac{\partial \mathcal{L}}{\partial x_{\ell-1}} = \frac{\partial \mathcal{L}}{\partial x_\ell} g'(x_{\ell-1});$$

being g the ReLU, $g'(x_{\ell-1})$ is either 0 or 1 with equal probability, hence, assuming $g'(x_{\ell-1})$ and $\frac{\partial \mathcal{L}}{\partial x_\ell}$ uncorrelated, we get

$$\begin{aligned}\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= \frac{1}{2}\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right] = \frac{1}{2}\mathbb{E}\left[\hat{W}\right]\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] = 0, \\ \mathbb{E}\left[\left(\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right)^2\right] &= \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] = \frac{1}{2}\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right].\end{aligned}$$

Therefore, we can conclude that

$$\begin{aligned}\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= \frac{1}{2}\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right] \\ &= \frac{1}{2}\mathrm{Var}\left[\hat{W}\frac{\partial \mathcal{L}}{\partial x_\ell}\right] \\ &= \frac{n_{\mathrm{out}}}{2}\sigma_W^2 \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right].\end{aligned}$$

Thus, the initialization

$$\text{Var}[W] = \frac{2}{n_{\text{out}}}, \tag{5.4}$$

where $n_{\text{out}} = k^2 d$ with k the filter dimension and d the number of output channels (*fan out*), allows to preserve the variance of gradients.

Note that equations (5.3) and (5.4) only differ for a factor which, in most common network architectures, is in fact equal to 1 in the vast majority of layers. Therefore, the initialization proposed by [62] should generally lead to the conservation of both *forward* and *backward* signals.

In a recent work [22] argued that initial weights should not be considered as random variables, but are rather the realization of a random process. Thus, weights mean and variance are empirical values different from those of the generating random process. Hence, normalization of the weights matrix should be performed after sampling to obtain the desired moments. Moreover, they argue that channel-wise responses should be analyzed. The derivations in (5.2) should be revised in order to consider expected value and the variance of any single channel i of the output x_ℓ and to take into account constant $\sigma^2_{W_i}$ and $\mu_{W_i} = 0$; specifically, we obtain

$$\begin{aligned}
\text{Var}[x_{\ell i}] &= \sum_{j=1}^{n_{\text{in}}} \text{Var}[W_{ij} z_{\ell j}] = \sum_{j=1}^{n_{\text{in}}} W_{ij}^2 \text{Var}[z_{\ell j}] \\
&= n_{\text{in}} \left(\sigma_g^2 \cdot \frac{1}{n_{\text{in}}} \sum_{j=1}^{n_{\text{in}}} W_{ij}^2 \right) \\
&= N \sigma_g^2 (\mu_{W_i}^2 + \sigma_{W_i}^2) \\
&= n_{\text{in}} \sigma_g^2 \sigma_{W_i}^2,
\end{aligned}$$

so that we retrieve the following initialization rule to preserve an activation signal with unit variance:

$$\text{Var}[W_i] = \frac{\gamma_g^2}{n_{in}}, \tag{5.5}$$

where $\gamma_g^2 = 2/(1 - \frac{1}{\pi})$ for the ReLU activation. Note that if mean and variance are preserved channel-wise, then they are also preserved if the whole layer is taken into account.

The authors do not take into account the *backward variance*. The authors [22] show that the latter initialization scheme allows to experimentally preserve the channel-wise activation variance, whereas He et al. technique only works at the full-layer level.

In the ResNet setting, initialization alone is not sufficient to make the training properly work without batch normalization, if the commonly employed architecture with Identity Shortcuts (see Figure 5.2a) is considered.

In particular, the skip-branch summation

$$x_\ell = x_{\ell-1} + f_\ell(x_{\ell-1}), \tag{5.6}$$

at the end of each block does not preserve variance, causing the phenomenon known as *internal covariate shift* [71].

In order to overcome this issue, Batch Normalization has been devised. More recently, effort has been put into designing other architectural and algorithmic modifications that dot not rely on batch statistics.

Specifically, [8,36,152] modified the skip-identity summation as to downscale the variance at the beginning of training, biasing, in other words, the network towards the identity function, i.e., computing

$$x_{\ell+1} = x_{\ell-1} + \alpha f_\ell(x_{\ell-1}).$$

This has the downside that α must be tuned and is dependent on the number of layers. More specifically, α is often initialized to zero so that the gradient is dominated, early on in the training, by the skip path. While these approaches have been shown to allow the training of very deep networks, they still struggle to obtain state-of-the-art test results on challenging benchmarks. As a matter of fact, as outlined in [22], while these solutions enjoy good convergence on the training set, they appear not to be sufficient to make deep ResNets reach state-of-the-art test accuracies.

Similarly, [131] suggest to compute the output of the residual branch as a weighted sum between the identity and the non-linear branch. Formally, the residual layer becomes

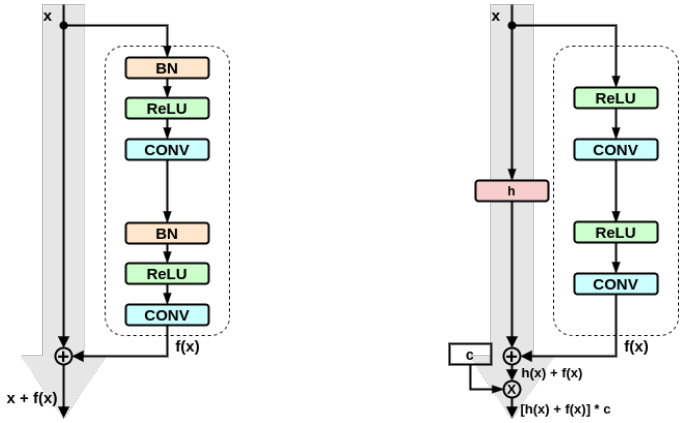
$$x_\ell = \alpha_\ell x_{\ell-1} + \beta_\ell f(x_{\ell-1}),$$

where coefficients α_ℓ and β_ℓ can be set so that the *forward variance* is conserved by imposing that $\alpha_\ell^2 + \beta_\ell^2 = 1$. Different strategies can be employed to choose their relative value.

More recently, [22] proposed to additionally perform a runtime layer-wise normalization of the weights [70, 120], together with the empirical channel-wise intialization scheme. However, we show in the following that the latter scheme, while enjoying perfectly conserved forward variances, induces the network to work in a regime of *exploding gradients*, i.e., the variance of the

gradients of the shallowest layers is exponentially larger than that of the deepest ones. Reasonably, [21] found the use of a tailored adaptive gradient clipping to be beneficial because of this reason.

5.4 Proposed Method



(a) Standard pre-activated Residual Block

(b) Generalized Normalizer-Free Residual Block

Figure 5.2: Architectures of Residual Blocks. For both pictures the grey arrow marks the easiest path to propagate the information.

In order to overcome the issue discussed at the end of the previous section, we propose to modify the residual block of ResNet architectures so that, at the beginning of the training, the mean of either the activations or the gradients is zero and the variance is preserved throughout the network. In our view, our proposal is a natural extension of the work of [62] for the case of ResNet architectures. Note that, to develop an effective initialization scheme, the residual block has to be slightly modified.

Namely, we propose the following general scheme (see Figure 5.2b):

$$x_\ell = c \cdot (h(x_{\ell-1}) + f_\ell(x_{\ell-1}))\,, \tag{5.7}$$

where c is a suitable constant, h is a generic function operating on the skip branch and $f_\ell(x_{\ell-1})$ represents the output of the convolutional branch.

We are able, through a proper initialization, to have zero mean and controlled variance (either backward or forward) for each block f_ℓ.

In a typical ResNet architecture, f_ℓ is a sequence of two or three convolutions, each one preceded by a ReLU activation - *pre-activation* [64] - allowing to control both mean and variance through the standard initialization schemes (5.3) and (5.4). Note that *post-activated* ResNets do not allow f_ℓ to have zero (either gradient or activation) mean, which corroborates the analysis done by [63].

We perform the analysis in this general setting, deriving the condition h and c must satisfy in order to preserve either the forward or backward variance. Then, we propose different ways in which h and c can be defined to satisfy such conditions.

5.4.1 Forward Case

Firstly, we note that initializing the weights of each block f following rule (5.3), hypothesizing that $\mathbb{E}[x_{\ell-1}] = 0$ and $\mathrm{Var}[x_{\ell-1}] = 1$, we can easily obtain that

$$\mathbb{E}[f_\ell(x_{\ell-1})] = \mathbb{E}[x_{\ell-1}] = 0, \quad \mathrm{Var}[f_\ell(x_{\ell-1})] = \mathrm{Var}[x_{\ell-1}] = 1.$$

Recalling [131], we can make the reasonable assumption that $f_\ell(x_{\ell-1})$ and $h(x_{\ell-1})$ have zero correlation, thus, getting

$$\begin{aligned}
\mathbb{E}[x_\ell] &= c \cdot (\mathbb{E}[h(x_{\ell-1})]) + \mathbb{E}[f_\ell(x_{\ell-1})]) \\
&= c \cdot \mathbb{E}[h(x_{\ell-1})] \\
\mathrm{Var}[x_l] &= c^2 \cdot (\mathrm{Var}[h(x_{\ell-1})] + \mathrm{Var}[f_\ell(x_{\ell-1})]) \\
&= c^2 \cdot (\mathrm{Var}[h(x_{\ell-1})] + 1)
\end{aligned}$$

i.e., the activation signal can be preserved by defining h so that $\mathbb{E}[h(x_{\ell-1})] = 0$ and $\mathrm{Var}[h(x_{\ell-1})] = \frac{1}{c^2} - 1$.

5.4.2 Backward Case

For the gradients at layer $\ell - 1$, we have

$$\frac{\partial \mathcal{L}}{\partial x_{\ell-1}} = \frac{\partial \mathcal{L}}{\partial x_\ell} \frac{\partial x_\ell}{\partial x_{\ell-1}} = c \cdot \frac{\partial \mathcal{L}}{\partial x_\ell} \left(\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}} + \frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}} \right).$$

We can assume by induction that the gradients at layer l have zero mean, i.e., $\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right] = 0$. Then, we get

$$\begin{aligned}\mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= c \cdot \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\frac{\partial x_\ell}{\partial x_{\ell-1}}\right] \\ &= c \cdot \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\mathbb{E}\left[\frac{\partial x_\ell}{\partial x_{\ell-1}}\right] = 0.\end{aligned}$$

Assuming zero correlation between $\frac{\partial \mathcal{L}}{\partial x_\ell}$ and $\frac{\partial x_\ell}{\partial x_{\ell-1}}$, we can further write

$$\begin{aligned}\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= c^2\left(\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\mathrm{Var}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}} + \frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right]\right. \\ &\quad + \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\mathbb{E}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}} + \frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right]^2 \\ &\quad \left. + \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]^2 \mathrm{Var}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}} + \frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right]\right) \\ &= c^2\left(\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\left(\mathrm{Var}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right] + \mathrm{Var}\left[\frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right]\right)\right. \\ &\quad + \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\left(\mathbb{E}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right] + \mathbb{E}\left[\frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right]\right)^2 \\ &\quad \left. + \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]^2\left(\mathrm{Var}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right] + \mathrm{Var}\left[\frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right]\right)\right).\end{aligned}$$

Now, we also know that, if we initialize the weight of each block f_ℓ by rule (5.4), it holds $\mathbb{E}\left[\frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right] = 0$ and $\mathrm{Var}\left[\frac{\partial f_\ell(x_{\ell-1})}{\partial x_{\ell-1}}\right] = C$. Therefore we can conclude

$$\begin{aligned}\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= c^2 \cdot \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\left(\mathrm{Var}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right] + 1\right) \\ &\quad + c^2 \cdot \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_\ell}\right]\mathbb{E}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right]^2.\end{aligned} \tag{5.8}$$

The preservation of the gradients signal can thus be obtained by suitably defined h and c.

We argue that some of the techniques proposed by [21, 22] to train deep Residual Networks (weight normalization layers, adaptive gradient clipping, etc.) become necessary because initialization (5.5) focuses on the preservation of the forward activation signal while disregarding the backward one.

Indeed, the correction factor $\gamma_g^2 = 2/(1 - \frac{1}{\pi})$ in (5.5) breaks the conservation property of the gradients signal, as opposed to (5.3). As we back-propagate through the model, the factor γ_g^2 amplifies the gradients signal at

each layer, so that the gradients at the last layers are orders of magnitude larger than those at the first layers (going from output to input layers), i.e., the network is in a regime of *exploding gradient*. In the section devoted to the numerical experiments we will show the forward and backward behaviour of these nets.

5.4.3 Gradient signal preserving setups

It is well know that *exploding gradients* make training hard (from an optimization perspective). Indeed, without further algorithmic or architectural tricks we are unable to train very deep networks. It is important to note that in the seminal analyses from [53] and [62] the derivation implied that preserving the forward variance entailed preserving also the backward variance too (at least to some reasonable amount). Indeed forward and backward variance can be equally preserved if, as already noted, for each layer, the number of input and output channels is equal. On the contrary, in the derivation of [21, 22], this relationship between forward and backward variance is lost so that conserving the forward variance implies *exploding gradients*.

For this reason, in the following we mainly focus on the backwards signal, which we argue being a more important thing to look at when forward and backward variance are not tightly related. For this reason, we propose three different possible schemes for choosing c and h in (5.7). In particular:

1. **scaled identity shortcut (IdShort):** $h(x) = x$, $c = \sqrt{0.5}$.

   This choice, substituting in (5.8), leads to

   $$\begin{aligned}\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= \frac{1}{2} \cdot \left(\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right] \cdot 1 + \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right] \cdot 1\right) \\ &= \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right],\end{aligned}$$

   i.e., the variance of gradients is preserved. As for the activations, we get $\mathbb{E}[x_\ell] = 0$ and

   $$\mathrm{Var}[x_\ell] = 0.5 \cdot (1 + 1) = 1.$$

   Thus, activations signal is preserved for all layers where input and output have the same size.

   Note that the latter scheme is significantly different from approaches, like those from [8, 36, 152], that propose to add a (learnable) scalar

that multiplies the skip branch. In fact, in the proposed scheme the (constant) scalar multiplies both branches and aims at controlling the total variance, without biasing the network towards the identity like in the other approaches.

This is the simplest variance preserving modification of the original scheme that can be devised, only adding a constant scalar scaling at the residual block.

2. **scaled identity shortcut with a learnable scalar (LearnScalar):** $h(x) = \alpha x$, α initialized at 1, $c = \sqrt{0.5}$. In (5.8) we again get at initialization

$$\begin{aligned}\operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] &= \frac{1}{2}\cdot\left(\operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right]\cdot 1 + \operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right]\cdot \alpha^2\right)\\ &= \operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right],\end{aligned}$$

and similarly as above we also obtain the forward preservation at all layers with $N = \hat{N}$.

3. **scaled identity shortcut with a (1×1)-strided convolution (ConvShort):** $h(x) = W_s x$ initialized by (5.4), $c = \sqrt{0.5}$. Since we use He initialization on the convolutional shortcut [64], we have $\mathbb{E}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right] = 0$ and $\operatorname{Var}\left[\frac{\partial h(x_{\ell-1})}{\partial x_{\ell-1}}\right] = 1$, hence we obtain in (5.8)

$$\operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell-1}}\right] = \frac{1}{2}\cdot\left(\operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right]\cdot 2 + \operatorname{Var}\left[\frac{\partial \mathcal{L}}{\partial x_{\ell}}\right]\cdot 0\right).$$

Again, if we consider the layers with equal size for inputs and outputs, we also get $\mathbb{E}[x_\ell] = 0$ and $\operatorname{Var}[x_\ell] = 0.5 \cdot (1 + 1) = 1$.

Note that this setting (without the scale factor) is commonly used in most ResNet architectures when $x_{\ell-1}$ and $f_\ell(x_{\ell-1})$ have not the same pixel resolution (for instance because f contains some strided convolution) or the same number of channels.

5.5 Experiments

We start the investigation by numerically computing forward and backward variances for the different initialization schemes. We employ the recently

introduced Signal Propagation Plots [22] for the forwards variance and a modification that looks at the gradients instead of the activations for the backwards case.

We employ the ResNet-50 and ResNet-101 architectures to extract the plots.

In particular we extract the plots for

- classical ResNet with He initialization, *fan in* mode (5.3) and *fan out* mode (5.4);
- same of the preceding with batch normalization;
- ResNet with the three proposed residual summation modifications and their proper intialization to preserve the backwards variance[1];
- same as the preceding but employing the intialization of Brock et al. [22].

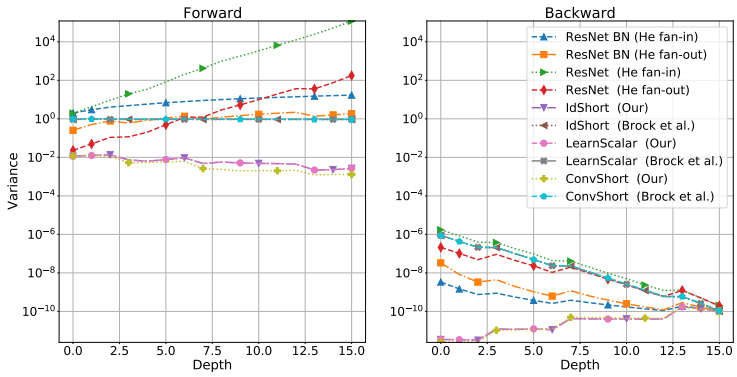
For all the initialization schemes, we perform the empirical standardization to zero mean and desired variance of weights at each layer, after the random sampling.

From Figure 5.3 we first note that, as already pointed out by [22], classical ResNets with He initialization do not preserve neither forwards nor backwards signals while the use of batch normalization manages to fix things up. Interestingly, we note that the observed trends are more conspicuous in deeper networks.

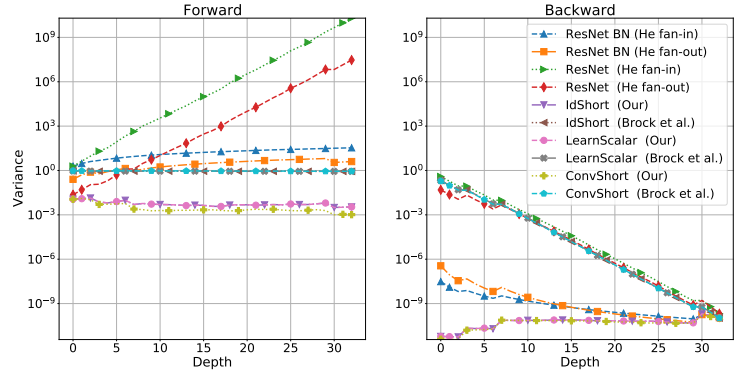
Next, we note that employing the proposed strategies (with proper initialization) we are able to conserve the variance of the gradients. On the contrary, the initialization proposed by [22] amazingly preserves the forward signal but puts the network in a regime of exploding gradients. Namely, the variance of the gradients exponentially increases going from the deepest to the shallowest residual layers. Additionally, we can also note how the proposed strategies also preserve the activations variance, up to some amount, while when employing the scheme of [22] the relationship between forward and backward variance is lost.

We continue the analysis by performing a set of experiments on the well-known CIFAR-10 dataset [78] in order to understand if an effective training

[1]Note that, as in the standard implementation, in IdShort and LearnScalar we employ ConvShort when x has not the same pixel resolution or number of channels of $f(x)$.



(a) ResNet-50



(b) ResNet-101

Figure 5.3: Signal propagation plots representing the variance of the forward activations (on the left) and the backward gradients variance (on the right) under different initialization schemes: both values refer to residual block output. The x-axis is the residual layer depth, while on the y-axis the variance of the signal is reported in a logarithmic scale.

can be actually carried out under the different schemes and compare them in terms of both train and test accuracy. In particular, we are interested in checking out if the proposed schemes can reach batch normalization test

performance.

All the experiments described in what follows have been performed using SGD with an initial learning rate of 0.01, a momentum of 0.9 and a batch size of 128 (100 for ImageNet), in combination with a Cosine Annealing scheduler [97] that decreases the learning rate after every epoch. Moreover, in addition to the standard data augmentation techniques, we have also employed the recently proposed RandAugment method [34] and, just for ImageNet, the Label Smoothing technique [150].

In Figure 5.4 both train and test accuracies are shown for all the configurations. The results report the mean and the standard deviation of three independent runs.

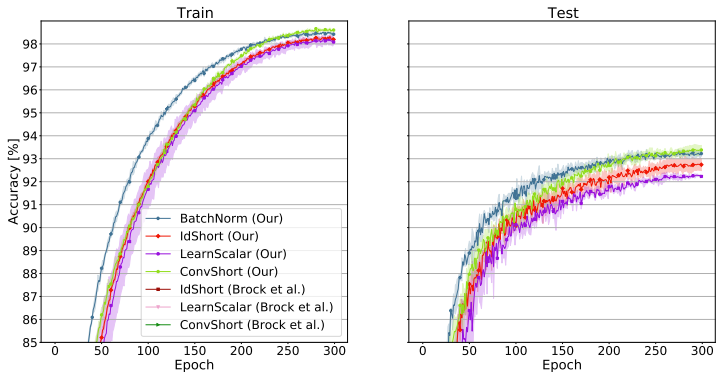
The first thing to notice is that with the initialization scheme of [22] we are unable to train the network (the curve is actually absent from the plot) for both ResNet-50 and ResNet-101. This is due to the fact that the network, at the start of the training, is in a regime of exploding gradients, as observed in the SPPs.

On the contrary, we can see how, thanks to the correct preservation of the backward signals, training is possible for all the proposed schemes when a gradient preserving initialization scheme is employed.

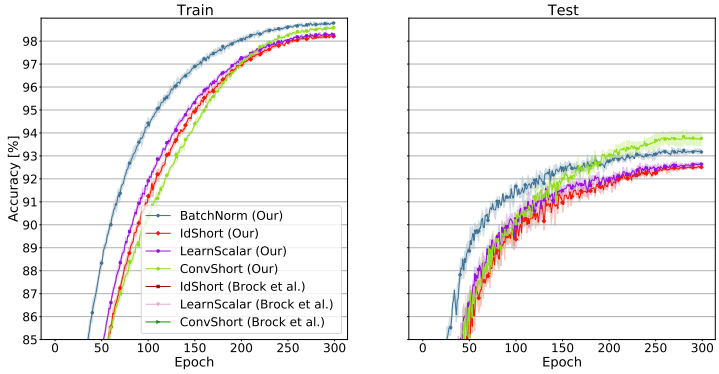
We also notice that, while all the schemes achieve satisfactory test accuracies, only the *ConvShort* modification has an expressive power able to close the gap (and even outperform at the last epochs) with the network trained using with Batch Normalization. Thus, according to Figure 5.4a and 5.4b, *ConvShort* appears to be an architectural change that, in combination with the proposed initialization strategy, is able to close the gap with a standard pre-activated ResNet with Batch Normalization.

To confirm the effectiveness of the proposed method we also considered more resource-intensive settings, where gradient clipping is expected to be necessary. In particular, we considered the well-known datasets CIFAR-100 [79] and ImageNet [38]. Based on the results obtained with CIFAR-10, we decided to test the most promising among our architectures, namely, *ConvShort* modification. Because of the limitations of the hardware and the computational resources at our disposal, we have not been able to carry out experiments with all the configurations. Moreover, no accurate hyperparameter validation could be carried out and experiments have been performed with a single random seed.

In Figure 5.5 we report the results obtained using our ShortConv mod-



(a) ResNet-50



(b) ResNet-101

Figure 5.4: Test and Train accuracies of ResNet under different combinations of residual block modifications and initialization: standard ResNet with BatchNorm and IdShort, LearnScalar, ConvShort using both [22] and ours initialization. Each experiment has been run three times: the solid line is the mean value while the surrounding shadowed area represents the standard deviation. Finally, the x-axis is the epoch at the which the accuracy (reported in the y-axis) has been computed.

ification and a standard ResNet-50 with BatchNormalization. Training is

Model	Input Resolution	Params (M)	#FLOPs (G)
ResNet-50 BatchNorm	$32 \times 32 \times 3$	38.02	4.2
ResNet-50 IdShort	$32 \times 32 \times 3$	23.47	2.6
ResNet-50 LearnScalar	$32 \times 32 \times 3$	23.47	2.6
ResNet-50 ConvShort	$32 \times 32 \times 3$	38.02	4.2
ResNet-101 BatchNorm	$32 \times 32 \times 3$	74.78	8.92
ResNet-101 IdShort	$32 \times 32 \times 3$	42.41	5.02
ResNet-101 LearnScalar	$32 \times 32 \times 3$	42.41	5.02
ResNet-101 ConvShort	$32 \times 32 \times 3$	74.78	8.92

Table 5.1: Computational cost and number of parameters of the considered architectures.

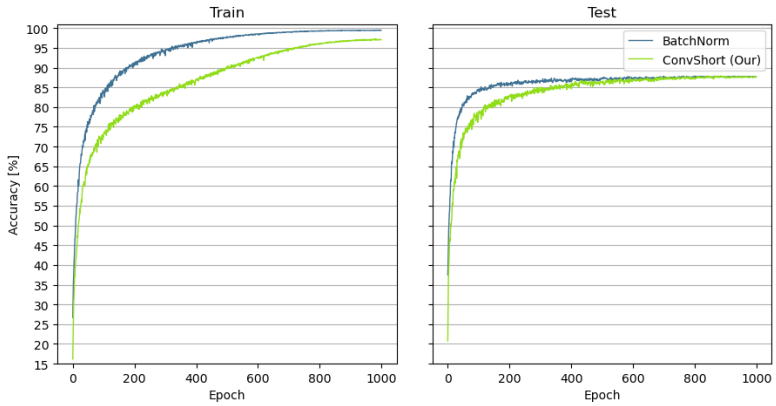


Figure 5.5: Comparison of Train and Test accuracies of ResNet-50 between standard ResNet with BatchNorm and ConvShort with our initialization using CIFAR-100. Values on x-axis denote the epoch at the which the accuracy on the y-axis has been computed.

slower for our setup, but the performance gap eventually disappears and testing accuracy of our approach becomes even slightly superior at the end of the process. In Figure 5.6 we show the results obtained with our ShortConv on the well-know ImageNet dataset. In order to evaluate the soundness of our proposal, we compare our results with the accuracy, reported on PyTorch [116], reached by a standard ResNet-50 trained on ImageNet. We can observe that the performance obtained with our architecture is in line with

the state-of-the-art.

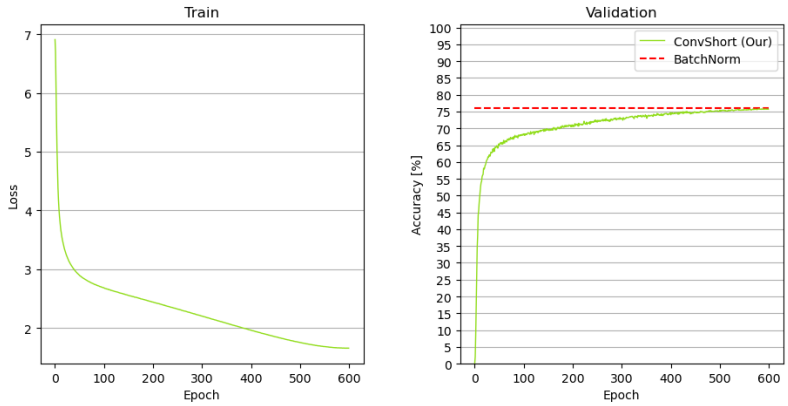


Figure 5.6: Results obtained training ResNet-50 with our ConvShort modification on ImageNet. Values on x-axis denote the epoch at the which the accuracy on the y-axis has been computed. The dashed red line is the accuracy reported by PyTorch [116] for a standard ResNet-50 trained on ImageNet.

The overall trend seems to indicate that DNNs can be trained up to state-of-the-art performance even without BN, even if this might come at the cost of a longer training; moreover, a strong data augmentation might be needed to compensate the lack of the implicit regularization effects of BN.

To conclude, we report the number of parameters and FLOPs for the considered architecture in Table 5.1. It is important to note that, despite *ConvShort* and *BatchNorm* have the same computational cost, our proposed method have some desirable characteristics (like the independence between the examples in a mini-batch). Moreover, the others configurations can be employed as more light-weight alternatives.

5.6 Final considerations

In this Chapter we discussed a slight architectural modification of ResNet-like architectures that, coupled with a proper weights initialization, can train deep networks without the aid of Batch Normalization. Such initialization scheme is general and can be applied to a wide range of architectures

with different building blocks. Importantly, our strategy does not require any additional regularization nor algorithmic modifications, as compared to other approaches. We show that this setting achieves competitive results on CIFAR-10, CIFAR-100, and ImageNet. The obtained results are in line with the discussed theoretical analysis.

Chapter 6

Conclusion

In this dissertation, we discussed the problem of efficiency in Machine Learning. As we saw, efficiency can be declined in many ways: reducing the input feature set, pruning the neural network's nodes (both in fully connected or convolutional layers), or removing complex components like Batch Normalization. Each of these problems can be addressed using different techniques, and, as it is possible to imagine, designing a generic algorithm to address all of them is not possible. For this reason, in this thesis, we discuss, in each Chapter, a possible solution to address one of the problems mentioned above. Summarizing the contributions of each Chapter we have:

In Chapter 2, we define an algorithm to select the best subset of features in a logistic regression problem. This algorithm, under suitable assumptions, has strong theoretical guarantees and shows robust empirical results.

In Chapter 3, we propose a novel approach to prune fully connected layers. In particular, exploiting the recently developed spectral reformulation [52], it is possible to rank the node's importance easily. According to this ranking, removing the less important ones is possible.

In Chapter 4, we propose a first analysis of a novel way to filter convolutional channel filters. More specifically, given an initial architecture, it is possible to formulate the problem of finding the best subset of convolutional filters as a bilevel optimization problem with a sparsity constraint. In this context, we develop, under suitable assumptions, a penalty decomposition algorithm to solve the bilevel problem, and we show some preliminary empirical results.

In Chapter 5, we characterize how the forward and backward signals prop-

agate inside ResNet-like models. Exploiting this analysis, we define a robust initialization that allows successful training without Batch Normalization.

To conclude, considering the results obtained for each proposal, we identify some possible future works. Specifically:

- Taking into consideration the problem of feature subset selection, the case of multi-class classification is also of great interest. However, the problem is challenging. Specifically, the complexity in directly extending the approach described in Chapter 2 to the multinomial case lies in defining the piece-wise linear approximation of the objective function. Indeed, in the multi-class scenario, the number of weights is $n \times m$, being m the number of classes, and $N \times m$ pieces of the objective function need to be approximated. Thus, we have to handle an increasingly high number of variables and constraints, which might become rapidly unmanageable, even exploiting our decomposition approach. Hence, future work might focus on devising alternative decomposition approaches designed to tackle the multinomial case.

- The extensions for the procedures described in Chapter 3 are two-fold. Preliminary results show that a suitable regularization of the eigenvalues yields a general improvement of the proposed method. Moreover, a possible extension for the convolutional layers can be designed.

- Improve the empirical analysis proposed in Chapter 4 to strengthen the soundness of the proposed approach.

Appendix A

Publications

This research activity has led to publications in international journals and conferences. These are listed below.[1]

International Journals

1. **Enrico Civitelli**, Matteo Lapucci, Fabio Schoen, Alessio Sortino. "An effective procedure for feature subset selection in logistic regression based on information criteria", *Computational Optimization and Applications*, 80 (1), 1-32.

2. Lorenzo Buffoni, **Enrico Civitelli**, Lorenzo Giambagli, Lorenzo Chicchi, Duccio Fanelli. "Spectral pruning of fully connected layers", *Scientific reports* , 12 (1), 1-9.

Submitted

1. **Enrico Civitelli**, Alessio Sortino, Matteo Lapucci, Francesco Bagattini, Giulio Galvan. "A Robust Initialization of Residual Blocks for Effective ResNet Training without Batch Normalization", *IEEE Transactions on Neural Networks and Learning Systems.* (Submitted after major revision)

[1]The author's bibliometric indices are the following: H-index = 2, total number of citations = 10 (source: Google Scholar on January 27, 2023).

International Conferences and Workshops

1. Gabriele Goretti, Benedetta Terenzi, Elisabetta Cianfanelli, Pierluigi Crescenzi, Carlo Colombo, **Enrico Civitelli**. "A phygital approach to playful experience in learning process for kids with special educational needs", in *International Conference on Education Technology and Computers*, London (United Kingdom), 2020.

2. Andrea Gemelli, Sanket Biswas, **Enrico Civitelli**, Josep Lladós, Simone Marinai. "Doc2Graph: a Task Agnostic Document Understanding Framework based on Graph Neural Networks", in *European Conference on Computer Vision (ECCV) - Workshop*, Tel Aviv (Israel), 2022.

Preprint

1. Tommaso Aldinucci, **Enrico Civitelli**, Leonardo Di Gangi, Alessio Sestini. "Contextual Decision Trees".

Patents

1. Tommaso Bianconcini, **Enrico Civitelli**, Simone Magistri, Francesco Sambo, Leonardo Sarti, Fabio Schoen, Leonardo Taccari. "Systems and Methods for Utilizing Machine Learning for Vehicle Detection of Adverse Conditions", U.S. Patent and Trademark Office. (Requested)

To be published

1. **Enrico Civitelli**, Gabriel Villalonga, Antonio M. López. "On-board Depth Estimation fusing RADAR and Single-Camera Data".

2. **Enrico Civitelli**, Simone Magistri, Tommaso Bianconcini, Leonardo Sarti, Francesco Sambo, Leonardo Taccari, Fabio Schoen. "Are large models really necessary to detect adverse driving conditions from on-board camera?".

Bibliography

[1] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[2] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.

[3] ——, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[4] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," *Advances in neural information processing systems*, vol. 32, 2019.

[5] D. Arpit, V. Campos, and Y. Bengio, "How to initialize your network? robust initialization for weightnorm & resnets," 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/e520f70ac3930490458892665cda6620-Paper.pdf

[6] D. Arpit, Y. Zhou, B. Kota, and V. Govindaraju, "Normalization propagation: A parametric technique for removing internal covariate shift in deep networks," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1168–1176.

[7] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *Found. Trends Mach. Learn.*, vol. 4, no. 1, p. 1–106, Jan. 2012. [Online]. Available: https://doi.org/10.1561/2200000015

[8] T. Bachlechner, B. P. Majumder, H. H. Mao, G. W. Cottrell, and J. J. McAuley, "Rezero is all you need: Fast convergence at large depth," *CoRR*, vol. abs/2003.04887, 2020. [Online]. Available: https://arxiv.org/abs/2003.04887

[9] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *arXiv preprint arXiv:1909.01377*, 2019.

[10] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," *arXiv preprint arXiv:1810.05723*, 2018.

[11] A. Beck and Y. C. Eldar, "Sparsity constrained nonlinear optimization: Optimality conditions and algorithms," *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1480–1509, 2013.

[12] A. Beck and N. Hallak, "On the minimization over sparse symmetric sets: projections, optimality conditions, and algorithms," *Mathematics of Operations Research*, vol. 41, no. 1, pp. 196–223, 2016.

[13] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," *arXiv preprint arXiv:1511.06297*, 2015.

[14] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[16] D. Bertsimas and V. Digalakis Jr, "The backbone method for ultra-high dimensional sparse machine learning," *arXiv preprint arXiv:2006.06592*, 2020.

[17] D. Bertsimas, A. King *et al.*, "Logistic regression: From art to science," *Statistical Science*, vol. 32, no. 3, pp. 367–384, 2017.

[18] D. Bertsimas, A. King, and R. Mazumder, "Best subset selection via a modern optimization lens," *The annals of statistics*, pp. 813–852, 2016.

[19] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya *et al.*, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optimization*, vol. 5, no. 2, pp. 186–204, 2008.

[20] H. Bozdogan, "Akaike's information criterion and recent developments in information complexity," *Journal of Mathematical Psychology*, vol. 44, no. 1, pp. 62–91, 2000.

[21] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," *ArXiv*, vol. abs/2102.06171, 2021.

[22] A. Brock, S. De, and S. L. Smith, "Characterizing signal propagation to close the performance gap in unnormalized resnets," in *International Conference on Learning Representations*, 2020.

[23] L. Buffoni, E. Civitelli, L. Giambagli, L. Chicchi, and D. Fanelli, "Spectral pruning of fully connected layers," *Scientific reports*, vol. 12, no. 1, pp. 1–9, 2022.

[24] O. P. Burdakov, C. Kanzow, and A. Schwartz, "Mathematical programs with cardinality constraints: reformulation by complementarity-type conditions and a regularization method," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 397–425, 2016.

[25] K. P. Burnham and D. R. Anderson, "Practical use of the information-theoretic approach," in *Model Selection and Inference*. Springer, 1998, pp. 75–117.

[26] ——, "Multimodel inference: understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.

[27] M. A. Carreira-Perpinán and Y. Idelbayev, ""learning-compression" algorithms for neural net pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8532–8541.

[28] J. Chang and J. Sha, "Prune deep neural networks with the modified $l\_\{1/2\}$ penalty," *IEEE Access*, vol. 7, pp. 2273–2280, 2018.

[29] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[30] L. Chicchi, L. Giambagli, L. Buffoni, T. Carletti, M. Ciavarella, and D. Fanelli, "Training of sparse and dense deep neural networks: Fewer parameters, same performance," *Physical Review E*, vol. 104, no. 5, p. 054312, 2021.

[31] E. Civitelli, M. Lapucci, F. Schoen, and A. Sortino, "An effective procedure for feature subset selection in logistic regression based on information criteria," *Computational Optimization and Applications*, vol. 80, no. 1, pp. 1–32, 2021.

[32] E. Civitelli, A. Sortino, M. Lapucci, F. Bagattini, and G. Galvan, "A robust initialization of residual blocks for effective resnet training without batch normalization," *arXiv preprint arXiv:2112.12299*, 2021.

[33] Y. Cooper, "Global minima of overparameterized neural networks," *SIAM Journal on Mathematics of Data Science*, vol. 3, no. 2, pp. 676–691, 2021.

[34] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[35] Y. N. Dauphin and S. Schoenholz, "Metainit: Initializing learning by learning to initialize," 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/876e8108f87eb61877c6263228b67256-Paper.pdf

[36] S. De and S. Smith, "Batch normalization biases residual blocks towards the identity function in deep networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[37] P. de Jorge, A. Sanyal, H. S. Behl, P. H. Torr, G. Rogez, and P. K. Dokania, "Progressive skeletonization: Trimming more fat from a network at initialization," *arXiv preprint arXiv:2006.09081*, 2020.

[38] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[39] L. Di Gangi, M. Lapucci, F. Schoen, and A. Sortino, "An efficient optimization approach for best subset selection in linear regression, with application to model selection and fitting in autoregressive time-series," *Computational Optimization and Applications*, vol. 74, no. 3, pp. 919–948, 2019.

[40] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[41] K. Dowling, R. Guzikowski, J. Ladd, H. Pangels, S. Singh, and W. Whittaker, "Navlab an autonomous navigation testbed," in *Vision and Navigation*. Springer, 1990, pp. 259–282.

[42] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[43] M. A. Duran and I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Mathematical programming*, vol. 36, no. 3, pp. 307–339, 1986.

[44] M. Efroymson, "Multiple regression analysis," *Mathematical Methods for Digital Computers*, pp. 191–203, 1960.

[45] F. Facchinei, V. Piccialli, and M. Sciandrone, "Decomposition algorithms for generalized potential games," *Computational Optimization and Applications*, vol. 50, no. 2, pp. 237–262, 2011.

[46] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[47] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz

et al., "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[48] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1165–1173.

[49] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1568–1577.

[50] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[51] A. Gemelli, S. Biswas, E. Civitelli, J. Lladós, and S. Marinai, "Doc2graph: a task agnostic document understanding framework based on graph neural networks," *arXiv preprint arXiv:2208.11168*, 2022.

[52] L. Giambagli, L. Buffoni, T. Carletti, W. Nocentini, and D. Fanelli, "Machine learning in spectral domain," *Nature communications*, vol. 12, no. 1, pp. 1–9, 2021.

[53] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[54] A. Gómez and O. Prokopyev, "A mixed-integer fractional optimization approach to best subset selection," *Optimization-online*, 2018.

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[56] L. Grippof and M. Sciandrone, "Globally convergent block-coordinate techniques for unconstrained optimization," *Optimization methods and software*, vol. 10, no. 4, pp. 587–637, 1999.

[57] M. Guignard and S. Kim, "Lagrangean decomposition: A model yielding stronger lagrangean bounds," *Mathematical programming*, vol. 39, no. 2, pp. 215–228, 1987.

[58] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

[59] B. Hanin and D. Rolnick, "How to start training: The effect of initialization and architecture," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/d81f9c1be2e08964bf9f24b15f0e4900-Paper.pdf

[60] E. J. Hannan and B. G. Quinn, "The determination of the order of an autoregression," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 2, pp. 190–195, 1979.

[61] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[63] ——, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[64] ——, "Identity mappings in deep residual networks," in *European conference on computer vision.* Springer, 2016, pp. 630–645.

[65] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu, "Reshaping deep neural network for fast decoding by node-pruning," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 245–249.

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

[67] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: Closing the generalization gap in large batch training of neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1729–1739.

[68] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression.* John Wiley & Sons, 2013, vol. 398.

[69] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[70] L. Huang, X. Liu, Y. Liu, B. Lang, and D. Tao, "Centered weight normalization in accelerating training of deep neural networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017.* IEEE Computer Society, 2017, pp. 2822–2830.

[71] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning.* PMLR, 2015, pp. 448–456.

[72] K. Jörnsten, M. Näsberg, and P. Smeds, *Variable Splitting: A New Lagrangean Relaxation Approach to Some Mathematical Programming Models*, ser. LiTH MAT R.: Matematiska Institutionen. University of Linköping, Department of Mathematics, 1985.

[73] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[74] S. Kamiya, R. Miyashiro, and Y. Takano, "Feature subset selection for the multinomial logit model via mixed-integer optimization," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 1254–1263.

[75] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares," *IEEE journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[76] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[77] S. Konishi and G. Kitagawa, *Information criteria and statistical modeling*. Springer Science & Business Media, 2008.

[78] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[79] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," *URL http://www. cs. toronto. edu/kriz/cifar. html*, vol. 5, no. 4, p. 1, 2010.

[80] M. Lapucci, T. Levato, F. Rinaldi, and M. Sciandrone, "A unifying framework for sparsity constrained optimization," *arXiv preprint arXiv:2104.13244*, 2021.

[81] M. Lapucci, "Theory and algorithms for sparsity constrained optimization problems," 2022.

[82] M. Lapucci, T. Levato, and M. Sciandrone, "Convergent inexact penalty decomposition methods for cardinality-constrained problems," *Journal of Optimization Theory and Applications*, vol. 188, no. 2, pp. 473–496, 2021.

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

[84] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[85] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[86] J. Lee, D. Joo, H. G. Hong, and J. Kim, "Residual continual learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 4553–4560.

[87] S.-I. Lee, H. Lee, P. Abbeel, and A. Y. Ng, "Efficient ℓ_1 regularized logistic regression," in *AAAI*, vol. 6, 2006, pp. 401–408.

[88] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[89] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. Van Gool, "Revisiting random channel pruning for neural network compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 191–201.

[90] Y. Li, X. Jin, J. Mei, X. Lian, L. Yang, C. Xie, Q. Yu, Y. Zhou, S. Bai, and A. L. Yuille, "Neural architecture search for lightweight non-local networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 297–10 306.

[91] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[92] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eYHoC5FX

[93] S. Liu, T. Chen, X. Chen, L. Shen, D. C. Mocanu, Z. Wang, and M. Pechenizkiy, "The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training," *arXiv preprint arXiv:2202.02643*, 2022.

[94] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[95] G. Liuzzi and F. Rinaldi, "Solving ℓ_0-penalized problems with simple constraints via the Frank-Wolfe reduced dimension method," *Optimization Letters*, vol. 9, no. 1, pp. 57–74, 2015.

[96] V. Lomonaco, D. Maltoni, and L. Pellegrini, "Rehearsal-free continual learning over small non-iid batches." in *CVPR Workshops*, 2020, pp. 989–998.

[97] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[98] Z. Lu and Y. Zhang, "Sparse approximation via penalty decomposition methods," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2448–2478, 2013.

[99] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," 2019.

[100] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International conference on machine learning*. PMLR, 2015, pp. 2113–2122.

[101] S. Magistri, F. Sambo, F. Schoen, D. C. de Andrade, M. Simoncini, S. Caprasecca, L. Kubin, L. Bravi, and L. Taccari, "A lightweight deep learning model for vehicle viewpoint estimation from dashcam images," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[102] M. Masana, J. van de Weijer, L. Herranz, A. D. Bagdanov, and J. M. Alvarez, "Domain-adaptive deep network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4289–4297.

[103] A. Mehra and J. Hamm, "Penalty method for inversion-free deep bilevel optimization," in *Asian Conference on Machine Learning*. PMLR, 2021, pp. 347–362.

[104] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[105] A. Miller, *Subset selection in regression*. Chapman and Hall/CRC, 2002.

[106] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5191–5198.

[107] R. Miyashiro and Y. Takano, "Mixed integer second-order cone programming formulations for variable selection in linear regression," *European Journal of Operational Research*, vol. 247, no. 3, pp. 721–731, 2015.

[108] ——, "Subset selection by Mallows' Cp: A mixed integer programming approach," *Expert Systems with Applications*, vol. 42, no. 1, pp. 325–331, 2015.

[109] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[110] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[111] H. Mousavi, M. Loni, M. Alibeigi, and M. Daneshtalab, "Pr-darts: Pruning-based differentiable architecture search," *arXiv preprint arXiv:2207.06968*, 2022.

[112] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7197–7206.

[113] J. O. Neill, "An overview of neural network compression," *arXiv preprint arXiv:2006.03669*, 2020.

[114] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," *arXiv preprint arXiv:1509.06569*, 2015.

[115] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[116] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[117] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[118] F. Pedregosa, "Hyperparameter optimization with approximate gradient," in *International conference on machine learning*. PMLR, 2016, pp. 737–746.

[119] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *arXiv preprint arXiv:1802.05668*, 2018.

[120] S. Qiao, H. Wang, C. Liu, W. Shen, and A. Yuille, "Micro-batch training with batch-channel normalization and weight standardization," 2020.

[121] A. Radhakrishnan, M. Belkin, and C. Uhler, "Overparameterized neural networks implement associative memory," *Proceedings of the National Academy of Sciences*, vol. 117, no. 44, pp. 27 162–27 170, 2020.

[122] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[123] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International conference on machine learning*. PMLR, 2018, pp. 4334–4343.

[124] F. Rinaldi, F. Schoen, and M. Sciandrone, "Concave programming for minimizing the zero-norm over polyhedral sets," *Computational Optimization and Applications*, vol. 46, no. 3, pp. 467–486, 2010.

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

[126] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[127] K. A. Sankararaman, S. De, Z. Xu, W. R. Huang, and T. Goldstein, "The impact of neural network overparameterization on gradient confusion and stochastic gradient descent," in *International conference on machine learning*. PMLR, 2020, pp. 8469–8479.

[128] T. Sato, Y. Takano, R. Miyashiro, and A. Yoshise, "Feature subset selection for logistic regression via mixed integer optimization," *Computational Optimization and Applications*, vol. 64, no. 3, pp. 865–880, 2016.

[129] M. Schmidt and Y. Beck, "A gentle and incomplete introduction to bilevel optimization," 2021.

[130] G. Schwarz *et al.*, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[131] J. Shao, K. Hu, C. Wang, X. Xue, and B. Raj, "Is normalization indispensable for training deep neural network?" *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[132] X. Shen, W. Pan, Y. Zhu, and H. Zhou, "On constrained and regularized high-dimensional regression," *Annals of the Institute of Statistical Mathematics*, vol. 65, no. 5, pp. 807–832, 2013.

[133] N. D. Socci, D. D. Lee, and H. Sebastian Seung, "The rectified gaussian distribution," *Advances in Neural Information Processing Systems*, pp. 350–356, 1998.

[134] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou, "And the bit goes down: Revisiting the quantization of neural networks," *arXiv preprint arXiv:1907.05686*, 2019.

[135] D. J. Surmeier and R. Foehring, "A mechanism for homeostatic plasticity," *Nature neuroscience*, vol. 7, no. 7, pp. 691–692, 2004.

[136] T. Suzuki, H. Abe, T. Murata, S. Horiuchi, K. Ito, T. Wachi, S. Hirai, M. Yukishima, and T. Nishimura, "Spectral pruning: Compressing deep neural networks via spectral analysis and its generalization error," *arXiv preprint arXiv:1808.08558*, 2018.

[137] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[138] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[139] A. M. Tillmann, D. Bienstock, A. Lodi, and A. Schwartz, "Cardinality minimization, constraints, and regularization: A survey," *arXiv preprint arXiv:2106.09606*, 2021.

[140] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[141] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[142] H. Wang, Q. Zhang, Y. Wang, L. Yu, and H. Hu, "Structured pruning for efficient convnets via incremental regularization," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[143] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[144] X. Wang, F. Yu, L. Dunlap, Y.-A. Ma, R. Wang, A. Mirhoseini, T. Darrell, and J. E. Gonzalez, "Deep mixture of experts via shallow embedding," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 552–562.

[145] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero-norm with linear models and kernel methods," *Journal of Machine Learning Research*, vol. 3, no. Mar, pp. 1439–1461, 2003.

[146] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[147] J. Yan, R. Wan, X. Zhang, W. Zhang, Y. Wei, and J. Sun, "Towards stabilizing batch statistics in backward propagation of batch normalization," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkgGjRVKDS

[148] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7370–7379.

[149] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "An improved GLMNET for L1-regularized logistic regression," *Journal of Machine Learning Research*, vol. 13, no. 64, pp. 1999–2030, 2012. [Online]. Available: http://jmlr.org/papers/v13/yuan12a.html

[150] C.-B. Zhang, P.-T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, and M.-M. Cheng, "Delving deep into label smoothing," *IEEE Transactions on Image Processing*, vol. 30, pp. 5984–5996, 2021.

[151] D. Zhang, H. Wang, M. Figueiredo, and L. Balzano, "Learning to share: Simultaneous parameter tying and sparsification in deep learning," in *International Conference on Learning Representations*, 2018.

[152] H. Zhang, Y. N. Dauphin, and T. Ma, "Residual learning without normalization via better initialization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1gsz30cKX

[153] Z. Zheng, Y. Fan, and J. Lv, "High dimensional thresholded regression and shrinkage effect," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 76, no. 3, pp. 627–649, 2014.