From Model Complexity Reduction to Feature Selection in Deep Learning: a Regularization Story

Enzo Tartaglione

INSTITUT POLYTECHNIQUE DE PARIS

Mémoire d'habilitation à diriger des recherches

From Model Complexity Reduction to Feature Selection in Deep Learning: a Regularization Story

Mémoire d'habilitation à diriger des recherches de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 Ecole doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité : Informatique, données, IA

Thèse présentée et soutenue à Palaiseau, le 04/09/2024, par

ENZO TARTAGLIONE

Composition du Jury :

Florence D'Alché-Buc Professeure, Télécom Paris, Institut Polytechnique de Paris	Présidente
Enrico Magli Full Professor, Politecnico di Torino	Examinateur
Aline Roumy Directrice de Recherche, Inria Rennes	Examinatrice
Wojciech Samek Full Professor, Technische Universität Berlin	Rapporteur
Nicu Sebe Full Professor, Università degli Studi di Trento	Rapporteur
Johan Suykens Full Professor, Katholieke Universiteit Leuven	Rapporteur

To my family and friends
who entice me to move forward.

Your positivity grounds my days
in the glowing years of my life.

Contents

Chapter 1

Introduction

This manuscript synthesizes the research I have conducted in the field of Deep Neural Networks (DNNs) since completing my Ph.D. thesis. At the heart of my work is an exploration of DNN algorithms renowned for their ability to solve complex tasks such as image classification and object detection. Driven by the dual challenges of optimizing learning strategies and questioning the necessity of current model complexities, my research aims to not only refine these algorithms but also to address their broader implications, including their potential for bias and how to address such a threat.

Central to my investigation are two interlinked questions.

- How can we enhance DNNs' learning capabilities to focus on relevant information automatically?
- Can we achieve efficient model performance without the extensive computational demands currently seen as necessary?

These research questions led me to explore model pruning and debiasing - efforts that not only seek to enhance the understanding we have regarding AI algorithms by optimizing the size of these models, but also to mitigate potential biases in the obtained solution.

The core of the research I have conducted this far lies in properly conditioning learning problems. This can be declined in a broad range of cases, from correcting ill-posed problems (like in the case of algorithmic bias occurrence), to preventing overfit. In both cases, *regularization*, namely a process that either adds an explicit term to the optimization problem or is implicitly included through extra constraints, can come to the rescue and propose elegant and effective solutions.

Timeline of my Work As a conclusive work for my Ph.D., I began working on the theme of neural network pruning [11]. This was the starting point of my exploration in

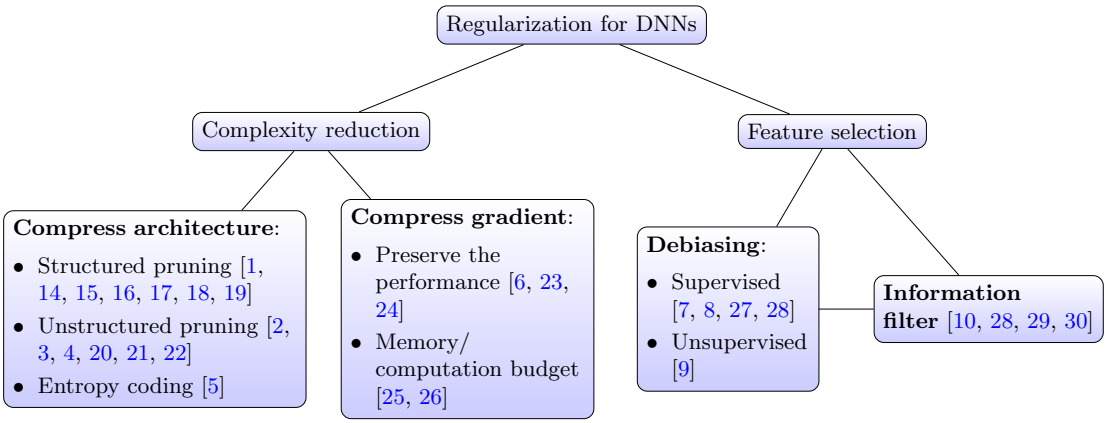


FIGURE 1.1: Research topics overviewed in the manuscript.

the field of neural network compression, conducted during my postdoc at the University of Turin, between 2019 and 2021 and leading as well to some of my recent efforts that evolved to concepts like layer collapse [15] or gradient compression [6, 26]. My engagement in the field is also witnessed by my participation in the European Lighthouse of AI for Sustainability (ELIAS), collaborating in the task "Reducing the energy requirements of computation". During the years of my postdoc I also collaborated, in the frame of the European project DeepHealth, to the realization of three medical usecases, which gave me the opportunity to get in touch with the problem of performing a proper feature selection to achieve better control on the DNN output generation. The problem of managing biased outputs in DNNs culminated in the early works of COVID detection from radiographic images, prophetically claimed as "working" but heavily affected by biases [31]. This experience motivated my will to explore two research axes in parallel (complexity reduction and feature selection), even when moved to Télécom Paris in 2021. Already in 2020, I believed that the two axes one day will join, as also witnessed by some preliminary experiments [28]. A summary of the topics treated in the manuscript is provided in Fig. 1.1. Beyond these, I have pursued related challenges, which include (but are not limited to) neural network watermarking [12, 32, 33], application of deep learning to medical tasks [31, 34, 35, 36, 37, 38, 39], input compression for novel view synthesis [40], video extrapolation [41, 42], learned image compression [43, 44], capsule networks [45] and efficient ensembling [13].

Complexity Reduction The advent of GPUs has undeniably propelled the scalability and usability of DNNs, enabling the execution of increasingly sophisticated models. More specifically, their ability to massively parallelize a sequence of operations like sums and products resulted in a great match with the requirements of Deep Learning. From that, we witnessed fast-paced scientific progress oriented to training deeper and deeper models. Among these, some notable knowledge bricks resulting essential for the current practice in deep learning include (but are not limited to):

- the advent of the ReLU activation [47, 48, 49];
- the design of the Xavier initializer before [50] and the Kaiming one later [51];
- the proposal of skip connections [52];
- the advent of the Adam optimizer [53];
- the design of self-attention layers, leading to transformer architectures [54].

All of this progress set the ground for the uprisal of *foundation models* [55]: large-scale models trained on a massive quantity of (heterogeneous) data, potentially adaptable to a broad variety of downstream tasks. While in principle employing these models in transfer learning or adaptation scenarios is convenient when dealing with small data, they pose some practical problems in terms of efficient adaptation to specific tasks (fine-tuning), and at deploy time (in terms of both energy and memory consumptions).

One critical metric to estimate computational efficiency for a DNN is Floating Point Operations Per Second (FLOPs), which serves as a first benchmark for computational demand. Looking even at more traditionally employed architectures like ResNet-50, it requires 7.7 GFLOPs for image processing tasks (at the typical ImageNet resolution). It is known that these architectures can be optimized: for example, MobileNet demands only 1.1 GFLOPs while achieving comparable accuracy. Such a gap evidently exists for more modern architectures and extends to other applications like Natural Language Processing, where the computational requirements can escalate to TFLOPs.

Despite the advancements in DNN design, many hand-crafted architectures exhibit over-parameterization. To address these inefficiencies, my initial efforts were focused on *pruning* techniques, aimed at removing redundant or superfluous parameters. According to recent trends in the field, the imperative to mitigate the energy consumption of deep models has gained more and more momentum, encapsulating this challenge within the fields of efficient edge AI and leading to the achievement of a "frugal AI".

This background underscores the first core aspect of my research: finding a proper trade-off between computational power and efficiency, framed within the context of sustainable and responsible AI development.

Feature Selection In the last decade, the over-excitement of the potentialities in terms of performance offered by deep learning pushed the models to be progressively more complex and sophisticated, sometimes obfuscating key aspects like transparency and explainability. This gap prompted legislative bodies, such as the European Commission, to propose regulatory frameworks like the Artificial Intelligence Act (AI Act)

in 2021. This act categorizes certain AI technologies as "high-risk" due to their potential societal impacts, spanning sectors from education to financial services. The call for (more) stringent guardrails imposes to address problems like the learning of spurious data correlations by DNNs: a phenomenon that might generate a *bias* in the AI outcomes. This bias manifests in two primary forms:

- *algorithmic*, emerging from the technicalities of AI development;
- *social*, a reflection of the broader effect on consequences discrimination-wise.

Although intrinsically linked together, addressing biases necessitates distinct approaches, from data curation to accounting for the effect a design might have. My research delves into this complexity, questioning (in the longer term) whether it is feasible to devise models that are both computationally efficient and (algorithmically) unbiased. The ultimate goal will be then to obtain models that leverage minimal computation without compromising on performance, avoiding them relying on (simpler) spurious correlations that lead to biases.

Considering these aspects, my work's ambition is not only to contribute to the technical evolution of DNNs but also to comply with upcoming regulations in the matter of AI, which shape the future of artificial intelligence. This dual focus emphasizes the need for a unified perspective between complexity reduction and feature selection.

1.1 Regularization and its Role in Deep Learning

Background on Regularization Pinpointing the exact birth of regularization methods poses a challenge, but they are commonly associated with the groundbreaking contributions of Tikhonov, as evidenced in seminal works such as [56, 57] and following efforts devoted by the community [58, 59]. The major motivation behind the first developments of regularization derives from the concept of *ill-posedness*, as a counterpart for well-posedness.

A problem is well-posed if a solution to it:

1. exists;
2. is unique;
3. depends continuously upon the input data.

Pioneering works around well-posed problems are attributed to Jacques Hadamard in the context of partial differential equations [60]. In Hadamard's idea, physical phenomena should be described by mathematical models showcasing the three core characteristics of well-posed problems: some examples of this are indeed the Dirichlet problem for Laplace's equation and the heat equation (with known initial conditions). However, *inverse problems* like denoising, compressive sensing, and training a DNN are often ill-posed, which poses both theoretical and practical challenges to their solution.

From the outset, it was recognized that to compute meaningful solutions it is necessary to approximate ill-posed problems with well-posed ones, often through parameterized families governed by some regularization parameter(s). Tikhonov, drawing from his background in topology, initially explored restricting the problem domain to compact sets in various topologies, leading to the concept of *conditional well-posedness*. In Hilbert spaces, norm balls around the origin emerged as a natural choice, being compact in the weak topology: the radius of these balls, or its inverse, naturally served as the regularization parameter, a method referred to as the "selection method" for solutions, yielding to what are named as *quasi-solutions*. For example, give a least-squares minimization problem $\|\boldsymbol{x}\boldsymbol{W} - \hat{\boldsymbol{y}}\|_2^2$ where $\boldsymbol{x}$ is the input, $\boldsymbol{W}$ represents the model's parameters, and $\hat{\boldsymbol{y}}$ is the target output, it is possible to favor specific solutions to the problem by including in the objective function to be minimized, for example, an extra term $\frac{\lambda}{2}\|\boldsymbol{W}\|_2^2$, with λ being a Lagrange multiplier.

From these developments, in the following half-century, a multitude of approaches to regularize ill-posed problems have been proposed for a broad range of applications, of which some notable applications include:

- the employment of linear filters to smoothen noisy data [61, 62];
- the development of the approximate inverse method [63];
- truncated singular value decomposition [64, 65];
- approaches for inversion of non-linear systems [66];
- the steepest descent methods [67];
- regularized Newton methods [68].

Although this list of contributions is by far non-representative of the many contributions in the field of regularization for ill-posed problems, to which a comprehensive analysis is provided by [69, 70, 71], it appears evident the big contribution of these foundational works to solving ill-posed problems, with clear applications to deep learning and machine learning in general.

Focusing on the matrix inversion issue (i.e. determining the parameters of a system), the issue of ill-posedness based on determinants is well-known, and if paired with non-linear functions, it can lead to multiple extrema, thereby violating the well-posedness required for optimization. However, despite the awareness of ill-posedness, these techniques are frequently employed: such effect is mild through the inclusion of regularization that can be either:

- explicit, like the commonly known ℓ_2 regularization - also referred to as *weight decay*;
- implicit, like with methods such as *early stopping* or *outlier removal*.

Through the employment of these techniques, with proper tuning of related hyper-parameters, a unique (family of) solution(s) for the model can be obtained. Evidently, the design and careful tuning of the regularization mechanism becomes crucial to the success of the model's training.

1.2 The Role of Regularization in my Work

While advancing complex DNN architectures, one of the core challenges is to achieve a complete understanding of the underlying mechanisms enabling learning. Achieving this is extremely challenging, but the integration of regularization functions can help the research to progress in such a direction.

Introducing a regularization term based on the ℓ_0 norm to determine the cardinality of the model's parameters, coupled with the conventional loss function, is twice effective: it not only facilitates the learning of the designated task by avoiding over-parametrization, but also reduces the computation complexity of the model itself. A proper design of a differentiable version for the ℓ_0 norm still remains a challenge, despite multiple efforts devoted by the community [72, 73, 74]. A promising approach involves designing algorithms capable of decomposing deep models into many sub-networks, of which the input-output function would be a-priori known. Achieving this would enable the application of cutting-edge optimization techniques to simplify the model's complexity in both an effective and interpretable way. Such exploration reflects a core aspect of my research, underscoring a commitment to enhancing the functionality and efficiency of DNNs.

The foundational pillars of my investigation - the analysis of DNNs' learned representations and their parameterizations - are hence deeply connected. While these domains

have traditionally been explored in isolation, their mutual dependence is intriguing, and sets the path towards effective model compression and, in the longer run, even enhancing *interpretability*. As such, the second research area presented in this work is bound to what I name after *feature selection*, intended as constraining internal features of the model to encode specific information. As such, the fields of *debiasing* and information filtering are very compatible with such an objective, given that some specific information flowing in the DNN should be either weighted or entirely filtered. Originating from techniques that prune DNNs and constrain information processing, my research's ambition in the long term will be to tackle both in synergy.

I wish to highlight that, in the interest of brevity, this manuscript does not fully encapsulate the entirety of my research endeavors. Certain investigations are currently in their infancy, and others, though relevant, necessitate detailed exploration beyond the scope of this document (as an example, my studies on DNN watermarking). A catalog of most of my publications is provided at the end of the manuscript.

1.3 Structure of the Manuscript

This manuscript is structured around pruning and debiasing, the two main axes of research led through my research years. I first discuss pruning, providing an overview of the state-of-the-art and my research's contributions, and integrating the new research directions and implications of model compression in terms of scalability of trained models and of back-propagation itself (Chapter 2). Then, I will provide a discussion of the problem of bias in DNNs, providing an overview of the currently most popular approaches and of my contributes in the field (Chapter 3). Finally, I will provide an overview of my current and prospective research (Chapter 4).

Chapter 2

Pruning in Deep Neural Networks

In this Chapter, I will discuss pruning, one of the possible approaches to reducing the size of a DNN model. Traditionally this literature tackles the problem of reducing the number of learnable parameters of a neural network as a proxy for their complexity: architectures such as AlexNet and VGG, conceptualized before 2015, already had a complexity in the order of 60 and 130 million parameters respectively. Similar architectures can be challenging to deploy in scenarios where resources such as memory or storage are limited. For example, already the 8-layer AlexNet [75] memory footprint exceeds 240 MB, whereas the 19-layer VGG-Net [76] exceeds 500 MB. Currently employed architectures such as Residual Neural Networks (ResNets) and Vision Transformers (ViT) showcase similar memory requirements or go even beyond. The need for compact DNNs is witnessed also by the fact that the Moving Pictures Experts Group (MPEG) of ISO has broadened its scope beyond multimedia contents issuing an exploratory call for proposal to compress neural networks [77].

Notably, other approaches are possible to cope with neural networks' memory requirements, inference time, and energy consumption. For example, re-designing the network topology by enforcing precise neural connectivity or weight sharing, can reduce the number of parameters, or the complexity of the network [52, 78]. This can be achieved by handcrafted designs, of which MobileNet is one of the most exemplificative architectures [79], or by automatic architecture-generating approaches, typically referred to as Neural Architecture Search agorithms [80, 81]. Another possible approach to model complexity's reduction involves the reduction of the numerical precision required to store the model or to perform computation [82, 83, 84]. Although related, pruning techniques aim at learning sparse topologies by selectively dropping synapses between neurons (or neurons altogether when all incoming synapses are dropped). The major difference established by pruning relies upon a strong prior pruning benefits from the

initial unpruned model, and by the fact that unpruned parameters preserve the same numerical precision. Although some hybrid approaches are currently in use [84, 85], we will focus here on pure pruning approaches.

Pruning is favored by the broadly known knowledge that DNNs are typically over-parametrized [86, 87]. Attempts to reduce the number of parameters from learned models deepen its roots in the past. In 1989, Mozer and Smolensky proposed *skeletonization*, a technique to identify less relevant neurons in a trained model and to remove them [88]. This is accomplished by evaluating the global effect of removing a given neuron, evaluated as an error function penalty from a pre-trained model. In the same years, LeCun *et al.* proposed a work where the information from the second-order derivative of the error function is leveraged to rank the parameters of the trained model on a *saliency* basis [89]. This allows the selection of a trade-off between the size of the network (in terms of the number of parameters) and performance. Despite these works, pruning lived a revival around 2015, thanks to the big effort invested in Deep Learning. Old approaches like [88, 89], although building the foundations for pruning, were conceptualized for shallow and small neural networks, and can not easily scale to deep and complex architectures. This generated the urgency to find proper trade-offs between invested computation and the effectiveness of the proposed approach. In such a sense, the work by Han *et al.* [90] opened the road to the more recent class of pruning approaches for DNNs.

2.1 Structure of the Chapter

In this Chapter, it is first presented an overview of the commonly shared scheme for iterative pruning approaches (Sec. 2.2). This is constituted mainly by three elements: pruning policy, stop criterion (Sec. 2.2.2.1), and fine-tuning strategy (Sec. 2.2.2.3). After these preliminaries, it will be presented at a glance my research as follows.

- *Sensitivity-based regularization* enforces sparsity at training/fine-tuning time for the subset of parameters having gradient null. I will present a structured (Sec. 2.3.1) and an unstructured (Sec. 2.3.2) variant of this formulation. Empirically it is observed that, despite removing consistently fewer parameters than the corresponding unstructured, enforcing structured pruning provides better savings in memory and computation [16].
- *Structured pruning for adapters* in pre-trained models is designed. Here the specific structure of adapters is leveraged to evaluate how much they impact the output generation and it is possible to trim them, reducing the extra memory cost

they induce (Sec. 2.5). It has also been applied a similar approach to different architectures, like Neural Radiance Fields, leading to comparable advantages [17].

- Under training on noisy data, the occurrence of the *sparse double descent* phenomenon [91] undermines the applicability of the typical stop criteria due to the double descent trend of performance evaluated on the validation set. It has been proposed an entropy measure that indicates when entering the classical bias-variance trade-off region, enabling these schemes. Besides, it has been observed that knowledge distillation from large over-parametrized models avoids shallower student models entering a double descent phase (Sec. 2.4).

- After observing that unstructured pruning makes the neural network architecture collapse when employing rectifier units, it has been developed an approach that *reduces the depth* of DNNs leveraging an "unstructured pruning" paradigm. This is concretely reducing the depth of a DNN model via pruning (Sec. 2.6), opening the doors to the phenomenon of *layer collapse* [15].

- The lottery ticket hypothesis quantitatively shows the existence - already at initialization (or in the first phases of training) - of subnetworks that, when trained in isolation, are able to match the performance of the full model [74], I have evidenced obstacles in finding a proper pruning strategy at initialization [23]. This analysis inspired the formulation of a different approach, where *the back-propagation graph is pruned* instead. It evaluates when neurons have reached an equilibrium condition and are not required to be further updated (Sec. 2.7). This gave birth to new research directions: notably, porting this concept to on-device learning scenarios where computation and memory are limited [26], or tuning the learning rate and formulating a stop criterion based on it.

2.2 Typical Pruning Scheme

In this section, I will illustrate the typical pruning framework employed by the vast majority of the works in the literature performing pruning. After some preliminaries (Sec. 2.2.1), the general iterative pruning approach is presented (Sec. 2.2.2), including a short review of typical approaches.

2.2.1 Preliminaries and Definitions

Let a feed-forward, a-cyclic, multi-layer artificial neural network be composed of $N-1$ hidden layers. We identify with $n=0$ the input layer and $n=N$ the output layer. For the i-th neuron of the n-th layer $x_{n,i}$, we define:

- $y_{n,i}$ as its output,
- $\boldsymbol{y}_{n-1}$ as its input vector,
- $\boldsymbol{\theta}_{n,i}$ as its own parameters: $\boldsymbol{w}_{n,i}$ the weights and $b_{n,i}$ the bias.

Each neuron has its own *activation function* $\phi_{n,i}(\cdot)$ to be applied after some affine function $f_{n,i}(\cdot)$ which can be for example a convolution or the dot product. Hence, the output of a neuron is given by $y_{n,i} = \phi_{n,i}[p_{n,i}]$, where $p_{n,i}$ is the *post-synaptic potential* of the neuron defined as $p_{n,i} = f_{n,i}(\boldsymbol{\theta}_{n,i}, \boldsymbol{y}_{n-1})$. A simplified representation is proposed in Fig. 2.1.

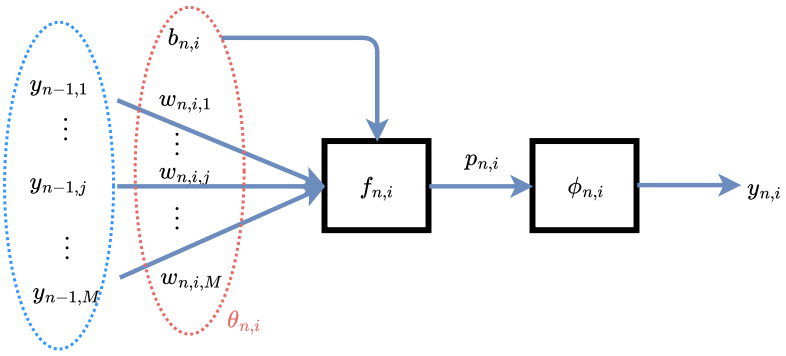


FIGURE 2.1: Simplified representation of a neuron.

2.2.2 Iterative Pruning Strategy

To proceed with applying an iterative pruning strategy on a target DNN trained on a specific dataset, we need to define a *pruning policy*, namely the strategy to select the parameters to remove from the model, a *fine tuning strategy* that aims at both recovering the performance of the model and to enforce sparsity, and a *stop criterion*, telling us when to terminate the iterative pruning algorithm. An overview of the typical iterative pruning scheme is presented in Fig. 2.2.

2.2.2.1 Pruning Policy and Stop Criterion

While most of the research focuses on how to perform fine-tuning, some devote their efforts to jointly optimizing the model and providing specific metrics to rank the parameters or the neurons to remove from a DNN model. Despite big research efforts

devoted in the last years, it has been found that ranking the parameters simply by their magnitudes provides a fair trade-off in terms of performance and computation complexity [92]. This is due to multiple effects, which include the regularizing effect of weight decay. Despite this, finding the optimal threshold for the parameter's removal remains an open challenge, as most of the pruning algorithms fix it as a hyper-parameter.

The act of removing parameters from a DNN model is often referred to as *thresholding*. Let us define $\tau(\cdot)$ as a function that extracts a proper value to be employed for the thresholding. When thresholding, we have

$$w_{n,i,j} = \begin{cases} w_{n,i,j} & \tau(w_{n,i,j}) > T \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

where T is typically a hyper-parameter, or is found through a quantile function on the parameters magnitudes distribution. For the mostly employed *magnitude pruning* approaches, we have that $\tau(\cdot) = |\cdot|$, ranking the parameters according to their magnitude. Similarly, gradient-based approaches rank the parameters according to the magnitude of their gradient. Because of the stochasticity introduced by mini-batch-based optimizers, parameters pruned during a thresholding iteration may be reintroduced by the following fine-tuning iteration. To overcome this effect, pruned parameters are enforced not to be updated during the following training iterations: we can name this behavior as *parameter pinning*).

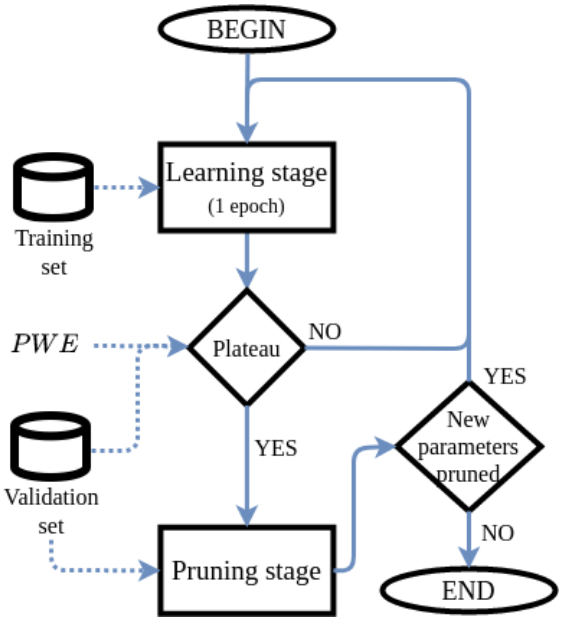


FIGURE 2.2: A general training procedure for iterative pruning approaches. Image adapted from [1].

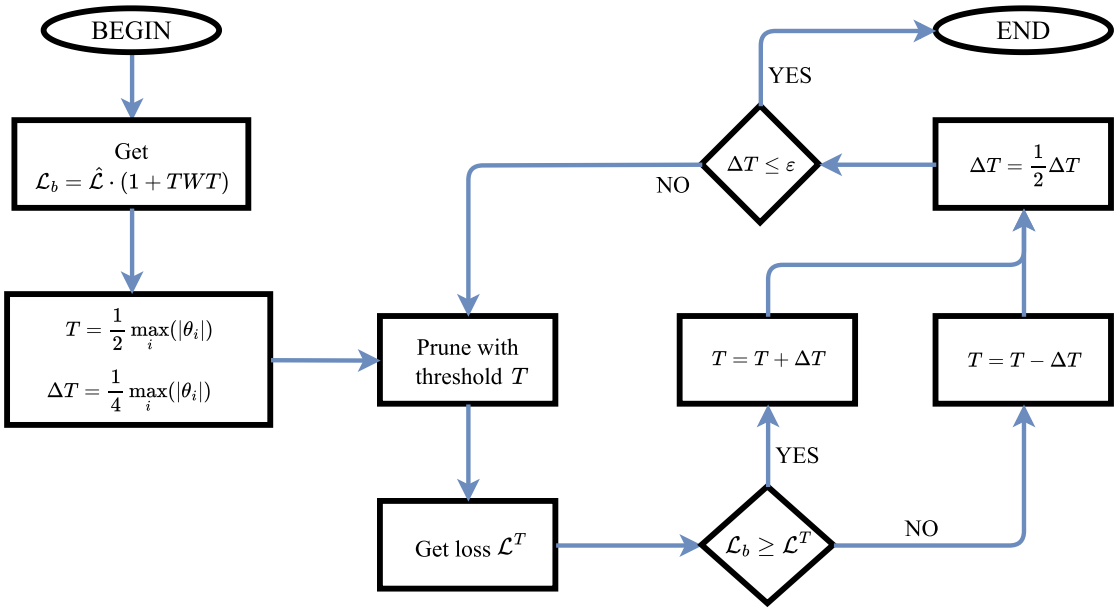


FIGURE 2.3: A possible dynamic threshold selection strategy for pruning. Given an initial model loss $\hat{\mathcal{L}}$ and a relative allowable worsening of the loss TWT, the algorithm finds the threshold T to prune the parameters and stops when the search interval ΔT drops below ε. Image adapted from [3].

Within my research, as an alternative to mainstream approaches, I have proposed a method in [1, 3] where the threshold T is found as a function of a relative worsening of the loss evaluated on the validation set. The pruning threshold T is selected so that the performance worsens at most of a relative value we call *thresholding worsening tolerance* (TWT) we provide as a hyper-parameter. We assume that, for small induced perturbations, the loss function is locally a smooth, monotone function of T, if the parameter's population is sufficiently large. The threshold T can be found using a bisection approach, converging to the choice of T value in log-time steps. A schematic overview of this approach is provided in Fig. 2.3.

Finally, the typical stop criterion is informally synthesized as "the iterative procedure is stopped when the performance degrades too much". This means that, given the initial model, the performance on a validation set is recorded, and when the fine-tuning can not recover it (potentially within a certain tolerance), then the whole iterative procedure is stopped. We will see that this can lead to some threats when working with real, noisy datasets, due to a sparse double descent effect, in Sec. 2.4.

2.2.2.2 Zero-shot Pruning

Recently, it has been observed by Frankle and Carbin that only a small portion of parameters are relevant to successfully conduct a model's training [74]. This is also known as *the lottery ticket hypothesis*, which underlying suggests that all the other

parameters can be removed from the learning process before training, without affecting the network performance. These latter parameters can effectively be determined a-posteriori, and the current challenge is to find ways to determine them before training. Lots of efforts have recently been devoted towards making pruning mechanisms more efficient: for example, Wang *et al.* show that some sparsity is achievable pruning weights at the very beginning of the training process [93], Liebenwein *et al.* build saliency scores to rank filters to be pruned [94], or Lee *et al.*, with their "SNIP", are able to prune weights in a one-shot fashion [95]. However, these approaches achieve limited sparsity only, while strategies based on iterative pruning usually enable higher sparsity [20].

I have shown that the difficulty of performing zero-shot pruning is due to improper management of extreme pruning in the loss landscape. More specifically, when removing a parameter a perturbation is introduced in the DNN model, which causes a drastic change in the local landscape of the loss [23]. Without a proper thresholding mechanism, or rather a look-ahead mechanism as in the lottery ticket hypothesis, it is unfortunately very easy to land in locally flat regions for the loss, making the optimization problem hard. On the contrary, as we will see in Sec. 2.7, it is possible to save computation in back-propagation by resorting to pruning the back-propagation graph, even at initialization. Notably, this problem has recently potentially found a way out, where some of the research is being ignited by the Neural Tangent Kernel theory [96] and finds one of its recent expressions in NTK-SAP [97].

2.2.2.3 Fine-tuning Strategies

Dropout Dropout aims at preventing a network from over-fitting by randomly dropping some neurons at learning time [98]. Despite dropout tackling a different problem, it has inspired some techniques aiming at sparsifying deep architectures. Kingma *et al.* [99] have shown that dropout can be seen as a special case of Bayesian regularization. Furthermore, they derive a variational method that allows to use dropout rates adaptively to the data. Molchanov *et al.* [73] exploited such variational dropout to sparsify both fully-connected and convolutional layers. In particular, the parameters having a high dropout rate are always ignored and they can be removed from the network. Even if this technique achieves good performance, it is quite complex and it is reported to behave inconsistently when applied to deep architectures [100]. Furthermore, this technique relies on the belief that the Bernoulli probability distribution (to be used with dropout) is a good variational approximation for the posterior. Another dropout-based approach is *Targeted Dropout* [101]: here, fine-tuning the ANN model is self-reinforcing its sparsity by stochastically dropping connections. They also target structured sparsity without, however, reaching state-of-the-art performance.

Regularization-based Pruning Regularization-based approaches rely on an explicit regularization term (designed to enhance sparsity) to minimize the loss function at training time. Louizos *et al.* propose a proxy for the ℓ_0 regularization to prune the network parameters during training [72]. Such a technique penalizes the non-zero value of a parameter vector, promoting sparse solutions. As a drawback, it requires solving a complex optimization problem, besides the loss minimization strategy and other regularization terms. In [102], Wen *et al.* propose a regularizer based on group lasso, whose goal is to cluster filters in convolutional layers. However, such a technique cannot be generalized to bulky fully-connected layers, where most of the complexity (in terms of number of parameters) lies. A sound approach towards pruning parameters consists of exploiting a ℓ_2 regularizer in a prune-and-finetune scheme. In particular, a standard ℓ_2 regularization term is included in the minimized cost function (to penalize the magnitude of the parameters): all the parameters dropping below some threshold are pinpointed to zero, thus learning a sparser topology [92]. Such an approach is effective since regularization replaces ill-posed problems with nearby and stable ones by introducing a prior on the parameters [103]. However, as a drawback, this method requires preliminary training to learn the threshold value; furthermore, all the parameters are blindly, equally penalized by their ℓ_2 norm: some parameters, which can introduce large errors (if removed), might drop below the threshold because of the regularization term. This introduces sub-optimalities as well as instabilities in the pruning process. Guo *et al.* attempted to address this issue with their DNS [104], where they propose an algorithmic procedure to correct eventual over-pruning by enabling the recovery of severed connections, or another possible approach has been proposed by He *et al.* with a "soft pruning" strategy [105, 106]. Overall, the general lack of these formulations lies in an explicit dependency of regularization strategies to the loss function.

2.3 Sensitivity-based Approaches

In this section, it is first formulated the sensitivity of the output of a DNN with respect to the post-synaptic potential of a target neuron. If such a value is low, then we can introduce a regularization that attempts to drive the norm of its parameters to zero and to be, eventually, pruned (Sec. 2.3.1). Then, this is extended to a variant where we target specific parameters (Sec. 2.3.2). As a reference scenario, according to the vast majority of the literature, a multi-class classification problem with C labels is considered; however, the same strategy can be extended to also other learning tasks.

2.3.1 Neuron Sensitivity

It will be here presented the work "Serene: Sensitivity-based regularization of neurons for structured sparsity in neural networks" [1]. It builds its foundations on my prior work "Learning sparse neural networks via sensitivity-driven regularization" [2].

Here I introduce the definition of *neuron sensitivity*. In this case, the objective would be to prune entire neurons rather than single parameters to achieve what is referred to in the literature as *structured sparsity* [102, 107, 108]. Let us assume that a pre-trained network is already provided (and eventually trained through any vanilla training strategy). To estimate the relevance of the i-th neuron belonging to the n-th layer, we evaluate the neuron contribution to the network output $\boldsymbol{y}_N$. To this end, we first provide intuition on how small variations of the post-synaptic potential $p_{n,i}$ of the neuron affect the k-th output of the network $y_{N,k}$. By a Taylor series expansion, for small variations of $p_{n,i}$, let us express the variation of $y_{N,k}$ as

$$\Delta y_{N,k} \approx \Delta p_{n,i} \frac{\partial y_{N,k}}{\partial p_{n,i}}. \tag{2.2}$$

In the case $\Delta y_{N,k} \to 0, \forall k$, for small variations of $p_{n,i}$, $y_{N,k}$ does not change. Such a condition allows to drive the post-synaptic potential $p_{n,i}$ to zero without affecting the network output $y_{N,k}$ (and, for instance, its performance). Otherwise, if $\Delta y_{N,k} \neq 0$, any variation of $p_{n,i}$ might alter the network output, possibly impairing its performance. We can now properly quantify the effect of small changes to the network output by defining the *neuron sensitivity*. The sensitivity of the network output $\boldsymbol{y}_N$ with respect to the post-synaptic potential $p_{n,i}$ of neuron $x_{n,i}$ is:

$$S_{n,i}(\boldsymbol{y}_N, p_{n,i}) = \frac{1}{C} \sum_{k=1}^{C} \left| \frac{\partial y_{N,k}}{\partial p_{n,i}} \right|. \tag{2.3}$$

Intuitively, the higher $S_{n,i}$, the higher the fluctuation of $\boldsymbol{y}_N$ for small variations of $p_{n,i}$.

Before moving on, we would like to clarify our choice of leveraging the post-synaptic potential $p_{n,i}$ rather than directly the neuron output $y_{n,i}$ in (2.3). To understand our choice, we re-write (2.3) using the chain rule:

$$S_{n,i}(\boldsymbol{y}_N, p_{n,i}) = \frac{1}{C} \sum_{k=1}^{C} \left| \frac{\partial y_{N,k}}{\partial y_{n,i}} \cdot \frac{\partial y_{n,i}}{\partial p_{n,i}} \right|. \tag{2.4}$$

Without loss of generality, let us assume $\frac{\partial y_{N,k}}{\partial y_{n,i}} \neq 0$. Under the hypothesis that $p_{n,i} < 0$, in case we are employing a rectifier activation such as ReLU, $\frac{\partial y_{n,i}}{\partial p_{n,i}} = 0$ for the considered ReLU activation. Had we written (2.3) as a function of the neuron output $y_{n,i}$, the vanishing gradient $\frac{\partial y_{n,i}}{\partial p_{n,i}} = 0$ would have prevented us from correctly estimating the neuron sensitivity.

Boundaries for the Sensitivity Given the complexity of efficiently coding (2.3) in traditional automatic differentiation schemes (given that each possible output channel k should be back-propagated in parallel), two computationally cheaper bounds to the sensitivity function were derived. Popular frameworks for DNN training rely on differentiation frameworks such as `autograd`, for automatic variable differentiation along computational graphs. Such frameworks take as input some objective function and automatically compute all the gradients along the computational graph. The major difficulty in (2.3) lies in the need of computing the absolute value of the whole chain rule, output element per output element. A lower bound to the sensitivity in (2.3) can be computed as

$$S_{n,i}^{\text{low}} = \left| \frac{1}{C} \sum_{k=1}^{C} \frac{\partial y_{N,k}}{\partial p_{n,i}} \right|, \tag{2.5}$$

which directly follows from the triangular inequality; and in the same spirit we can derive the upper bound

$$S_{n,i}^{\text{upper}} = \frac{1}{C} \left(\sum_{k=1}^{C} \left| \frac{\partial y_{N,k}}{\partial \boldsymbol{y}_{N-1}} \right| \right) \cdot \prod_{l=n+1}^{N-1} \left(\left| \frac{\partial \boldsymbol{y}_l}{\partial \boldsymbol{y}_{l-1}} \right| \right) \cdot \boldsymbol{\delta}_{n,i} \cdot \left| \frac{\partial y_{n,i}}{\partial p_{n,i}} \right|, \tag{2.6}$$

where $\boldsymbol{\delta}_{n,i}$ is a one-hot vector that selects the i-th neuron in the n-th layer. Both of the boundaries can be easily coded and requires only one back-propagation call.

Parameters Update Rule Now we show how the proposed sensitivity definition can be exploited to promote neuron sparsification. As hinted before, if the sensitivity $S_{n,i}$ is small, i.e $S_{n,i} \to 0$, then the target neuron yields a small contribution to the output and its parameters may be moved towards zero with little perturbation. To this end, we define the *insensitivity* function $\overline{S}_{n,i}$ as

$$\overline{S}_{n,i} = \max\{0, 1 - S_{n,i}\} = (1 - S_{n,i}) \cdot \Theta\left(1 - S_{n,i}\right), \tag{2.7}$$

where $\Theta(\cdot)$ is the one-step function. Empirically it is observed that in typical DNN learning regimes, it is rare to have $S_{n,i} > 1$, which would result in potential gradient explosion issues. The higher the insensitivity of a neuron (i.e., $\overline{S}_{n,i} \to 1$ or equivalently $S_{n,i} \to 0$), the less the neuron affects the network output. Therefore, if $\overline{S}_{n,i} \to 1$,

then the neuron contributes little to the network output, and its parameters $\boldsymbol{\theta}_{n,i}$ can be driven towards zero without significantly perturbing the network output. Using the insensitivity definition in (2.7), it is proposed the following update rule:

$$\theta_{n,i,j}^{t+1} = \theta_{n,i,j}^{t} - \eta \frac{\partial \mathcal{L}^t}{\theta_{n,i,j}^{t}} - \lambda \theta_{n,i,j}^{t} \overline{S}_{n,i}^{t}, \tag{2.8}$$

where:

- the first contribution term is the classical minimization of a loss function $\mathcal{L}(\cdot)$, ensuring that the network still solves the target task, e.g. classification;
- the second one represents a penalty applied to the parameter $\theta_{n,i,j}$ belonging to the target i-th neuron in the n-th layer, proportional to the insensitivity of the output to its variations.

The whole approach described is here named SeReNe. Empirical results presented in [1] show that the proposed approach enhances structured sparsity in DNNs while maintaining comparable performance as traditional approaches, thanks to the regulation induced by the sensitivity term.

2.3.2 Parameter-based Sensitivity

It will be here presented the work "Loss-based sensitivity regularization: towards deep sparse neural networks" [3].

DNNs are typically trained via gradient descent-based optimization, i.e. minimizing the loss function. Methods based on mini-batches of samples have gained popularity as they allow better generalization than stochastic learning while also being memory and time-efficient. A network parameter $\theta_{n,i}$ is updated towards the averaged direction which minimizes the averaged loss for the minibatch, i.e. using stochastic gradient descent or one of its variants. Our ultimate goal is to assess to which extent a variation of the single parameter θ_i would affect the error on the network output $\mathbf{y}_N$: the parameters not affecting the network output could be hardwired to zero, i.e. pruned away. In the same spirit as done in Sec. 2.3.1, we can make a first attempt towards this end by introducing a small perturbation $\Delta\theta_{n,i}$ over $\theta_{n,i}$ and measuring the variation of $\mathbf{y}_N$ as

$$\Delta \mathbf{y}_N = \sum_k |\Delta y_{N,k}| \approx \Delta \theta_{n,i} \sum_k \left| \frac{\partial y_{N,k}}{\partial \theta_{n,i}} \right|. \tag{2.9}$$

A direct evaluation of (2.9), as already observed in Sec. 2.3.1, is computationally expensive [2]. In this context, however, we can directly estimate the variations of the error function, using some differentiable proxy, i.e. the loss function:

$$\Delta\mathcal{L} \approx \Delta\theta_{n,i} \left| \frac{\partial \mathcal{L}}{\partial \mathbf{y}_N} \cdot \frac{\partial \mathbf{y}_N}{\partial \theta_{n,i}} \right| = \Delta\theta_{n,i} \left| \frac{\partial \mathcal{L}}{\partial \theta_{n,i}} \right| . \tag{2.10}$$

The use of (2.10) in place of (2.9) shifts the focus from the output to the error of the network. We can here define the *sensitivity* $S(\cdot)$ for a given parameter w_i as

$$S(\mathcal{L}, \theta_{n,i}) = \left| \frac{\partial \mathcal{L}}{\partial \theta_{n,i}} \right| . \tag{2.11}$$

Large $S(\cdot)$ values indicate large variations of the loss function for small perturbations of $\theta_{n,i}$. Plugging (2.11) directly in the optimization scheme is very computationally friendly, given that $\frac{\partial \mathcal{L}}{\partial \theta_{n,i}}$ should be calculated in any case to perform the loss minimization. Given the sensitivity definition in (2.11), we can promote sparse topologies by pruning parameters with both low sensitivity $S(\cdot)$ (i.e., in a flat region of the loss function gradient, where a small perturbation of the parameter has a negligible effect on the loss) and low magnitude. To this end, it is proposed the following parameter update rule to promote sparsity:

$$\theta_{n,i}^{t+1} = \theta_{n,i}^{t} - \lambda\Gamma\left(\mathcal{L}, \theta_{n,i}^{t}\right) - \frac{\partial \mathcal{L}}{\partial \theta_{n,i}^{t}} \left[\eta - \operatorname{sign}\left(\frac{\partial \mathcal{L}}{\partial \theta_{n,i}^{t}} \right) \lambda\Gamma\left(\mathcal{L}, \theta_{n,i}^{t}\right) \right] , \tag{2.12}$$

where

$$\Gamma\left(y, x\right) = x \cdot P\left(\frac{\partial y}{\partial x} \right) , \tag{2.13}$$

$P(x) = \Theta(1 - |x|)$, and $\Theta(\cdot)$ is the Heaviside function. In (2.12) we observe two different components of the proposed regularization term: a weight decay-like term $\Gamma\left(L, w_i\right)$ which is enabled/disabled by the magnitude of the gradient on the parameter; and a correction term for the learning rate. In particular, the full learning process follows an *equivalent* learning rate

$$\tilde{\eta} = \eta - \operatorname{sign}\left(\frac{\partial \mathcal{L}}{\partial \theta_{n,i}} \right) \lambda\Gamma\left(\mathcal{L}, \theta_{n,i}\right) . \tag{2.14}$$

In a nutshell, if we receive a signal from the loss gradient (meaning that we can optimize in a given direction), the contribution from the regularization is shut down; on the contrary, when we are in a flat region, we enforce regularization. This can be seen as a proxy for a second-order optimization method, that tries to enforce sparsity in a loss-aware fashion. A visualization of such an effect is displayed in Fig. 2.4.

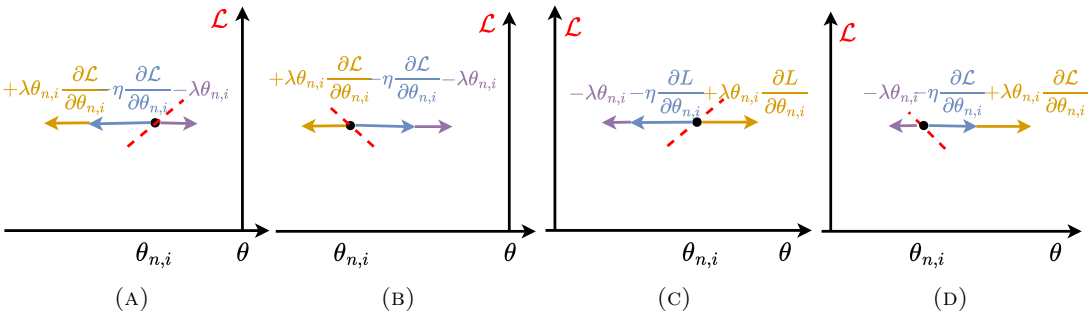


FIGURE 2.4: Update rule effect on the parameters. The red dashed line is the tangent to the loss function in the black dot, in blue the standard SGD contribution, in purple the weight decay while in orange the LOBSTER contribution. Image adapted from [3].

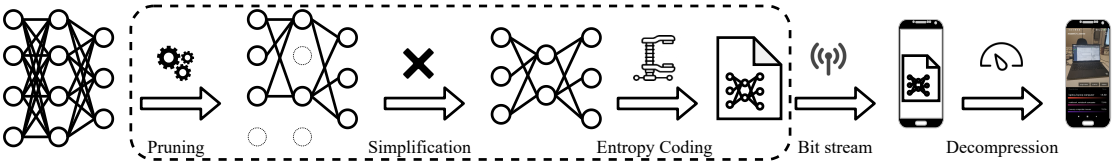


FIGURE 2.5: Neural network compression and pipeline (in the dashed box), inspired by the MPEG-7 part 17 standard. Image taken from [16].

The overall approach described here is referred to as LOBSTER. More details, discussion, and results are provided in [3].

2.3.3 Better Structured or Unstructured Sparsity?

It will be here presented the work "On the role of structured pruning for neural network compression" [16].

Traditionally, a large part of the literature focused on the so-called unstructured pruning, ie. focusing on maximizing the number of pruned parameters for a given performance target, without imposing constraints on the resulting tensor topology (similarly to what the objective of LOBSTER in Sec. 2.3.2). These approaches, such as those referenced by [92, 109], often achieve very high pruning ratios but may result in randomly sparse parameter tensors.

In contrast, structured pruning approaches impose constraints on the pruning process to enforce a more regular structure on the pruned topology (as achieved for SeReNe in Sec. 2.3.1). These methods, exemplified by works like [72, 102], may lead to network representations that are easier to store in memory, despite typically achieving lower pruning ratios due to the additional constraints.

In the spirit of assessing the potential advantages and disadvantages of either of the two approaches, it was conducted some quantitative comparisons for either structured

TABLE 2.1: Benchmark of pruning strategies. From the left: percentage of pruned parameters, size of the simplified network topology, and size of the compressed bitstream. Right: inference time on different embedded devices: Raspberry Pi 3B (RPi 3B), Huawei P20 (P20), Xiaomi MI 9 (MI9) and Samsung Galaxy S6 lite (S6L). Table taken from [16].

Dataset	**Arch.**	**Pruning**	**Pruning ratio [%]**	**Simplified topology [MB]**	**Compressed bitstream [MB]**	**Inference time [ms]**			
						RPi 3B	P20	MI9	S6L
CIFAR-10	VGG-16	No pruning	-	60.0	3.6	647	204	153	251
		LOBSTER [3]	**92.44**	58.61	1.61	610	191	146	242
		SeReNe [1]	47.16	**31.02**	**0.34**	**594**	**99**	**85**	**106**
	ResNet-32	No pruning	-	2.0	0.30	580	32	30	31
		LOBSTER [3]	**81.19**	1.96	0.12	545	32	26	30
		SeReNe [1]	52.80	**1.0**	**0.09**	**536**	**25**	**17**	**25**
CIFAR-100	AlexNet	No pruning	-	94.6	10.1	246	131	84	168
		LOBSTER [3]	**98.90**	48.84	0.40	224	95	67	120
		SeReNe [1]	59.87	**37.07**	**0.20**	**186**	**75**	**53**	**96**
ImageNet	ResNet-101	No pruning	-	178.4	26.24	11919	958	416	1008
		LOBSTER [3]	**87.39**	173.87	9.24	11879	956	403	985
		SeReNe [1]	1.09	**172.53**	**7.51**	**11699**	**929**	**371**	**974**

or unstructured pruning, choosing the MPEG-7 part 17 neural network compression pipeline (synthetically displayed in Fig. 2.5). To this, it is added a *simplification* stage, where neurons having no left parameters (ie. all its parameters are removed by the pruning algorithm) are concretely removed by the architecture, generating no memory or computation overhead. For such a purpose, a library named `simplify` has been developed and open-sourced [110].

A quantitative comparison is reported in Tab. 2.1. The models here showcase the same performance on the trained task for a matter of fair comparison. As anticipated, unstructured pruning approaches like LOBSTER can achieve an extremely high compression ratio, effectively removing more parameters from the network. However, structured pruning approaches like SeReNe produce more compact and simplified network topologies, which lead to benefits in terms of memory footprint and needed storage memory, besides faster computation. Despite a larger hype regarding achieving the highest parameter removal percentages, when deploying DNNs on general-purpose hardware, structured pruning approaches showcase their practical effectiveness.

A deeper discussion around this topic is presented in [16].

2.4 Pruning while Training on Noisy Data

It will be here presented the work "DSD²: Can We Dodge Sparse Double Descent and Compress the Neural Network Worry-Free?" [4]. However, this work builds on top of preliminary results presented in "Dodging the Double Descent in Deep Neural Networks" [22] and "Sparse Double Descent in Vision Transformers: real or phantom threat?" [21], whose contribute is hinted in the section.

Beyond Traditional Bias-variance Trade-off In real-world scenarios, data acquisition often introduces noise [111], whether from the data collection process itself or during labeling. Various approaches have been proposed to mitigate annotation noise. For instance, Li *et al.* suggest a unified distillation framework that leverages knowledge from a small, clean dataset and semantic knowledge graphs to rectify noisy labels [112]. Other solutions draw inspiration from the beneficial aspects of noise in biological nervous systems. Arani *et al.* demonstrate that introducing constructive noise at different levels within a collaborative learning framework can effectively train models and distill desirable characteristics in the student model, particularly for training high-performance, compact, adversarially robust models [113].

However, when limited information is available about the noise, there's a risk of overfitting the training set, leading to a phenomenon known as the "memorization phase". In this phase, the model memorizes individual samples from the training set and may learn incorrect features, thus impairing its generalization performance on unseen data [114]. Recently, a surprising phenomenon known as *double descent* (DD) [115] has been observed in extremely over-parametrized models. Beyond the traditional overfitting regime, as the size of the model increases, the generalization gap between training and test performance initially decreases and then narrows further (Fig. 2.6). This trend persists even with increasingly larger models [116].

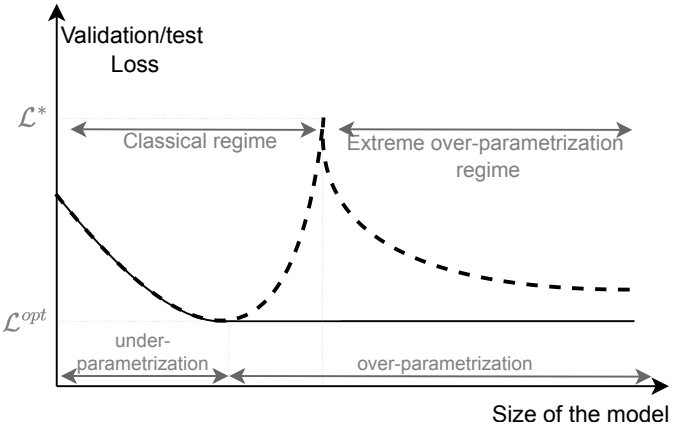
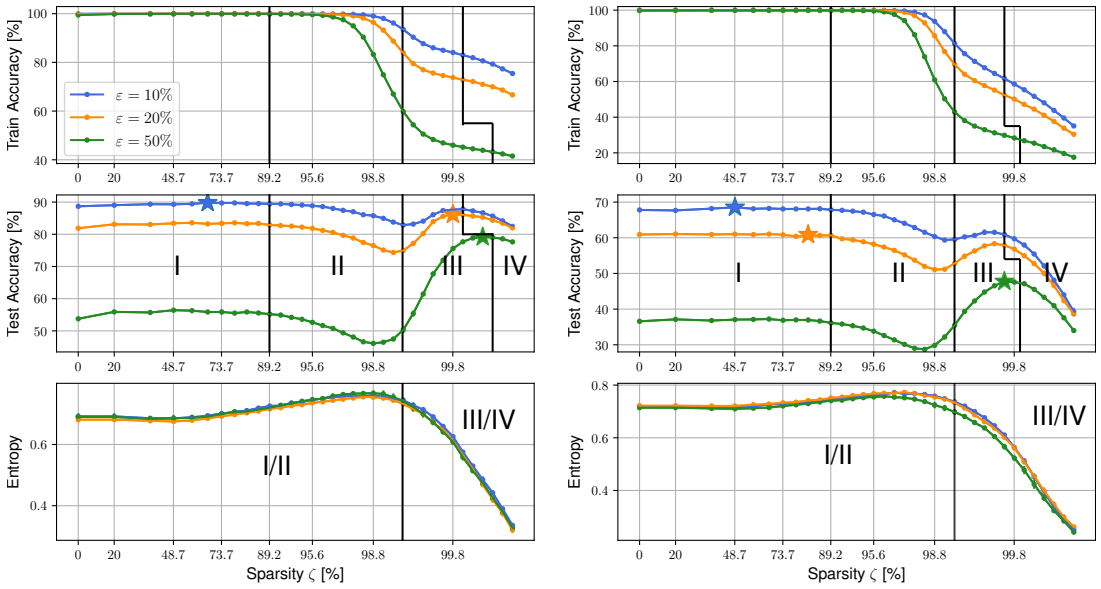


FIGURE 2.6: The double descent phenomenon (dashed line). Image adapted from [22].

DD prompts inquiries into the optimal sizing of models to achieve the best performance while minimizing their size. Numerous approaches have suggested employing regularization functions to alleviate DD in models used for regression and classification tasks.



(A) (B)

FIGURE 2.7: Performance of ResNet-18 with different amount of noise ε on CIFAR-10 (a) and CIFAR-100 (b). **I**: Light Phase. **II**: Critical Phase. **III**: Sweet Phase. **IV**: Collapsed Phase. Figure taken from [4].

However, the practical application of these methods is hindered by the complexity involved in optimally tuning the regularization hyperparameters and determining optimal early stopping, as highlighted by [117].

The Sparse Double Descent While the conventional analysis of DD focuses on transitioning from small models to larger ones, a recent study has identified a similar phenomenon occurring in the reverse direction – from an over-parametrized model to a smaller one. This observation is made possible through pruning, and takes the name of *sparse double descent* (SDD) [91]. In essence, while we know that in classical regimes pruning can lead to performance enhancement as parameters are removed from the model, followed by a performance drop, in SDD this trend can repeat, as sparsity increases. Such a behavior questions traditional stop criteria as the pruning can be wrongly early-stopped (as discussed in Sec. 2.2.2.1).

In the following, it will be first characterized the sparse double descent phenomenon (Sec. 2.4.1), proposing a metric that hints when a model is entering the classical regime that enables back stop criteria (Sec. 2.4.2), and then it will be proposed a regularization strategy based on knowledge distillation that boosts the performance of a smaller student model avoiding SDD regions (Sec. 2.4.3).

2.4.1 Pruning Exhibits Sparse Double Descent

In this section, I will briefly report the occurrence of the sparse double descent phenomenon, as observed in [91] and validated by my work [4].

Let us take a model trained on the train set $\mathcal{D}_{\text{train}}$ (whose performance is evaluated on the validation set $\mathcal{D}_{\text{val}}$). When we prune the ζ-th fraction of parameters from the model (referred to as sparsity in Fig. 2.7), the parameters are projected to a parameter sub-space. We here observe four phases, as also reported in [91]:

- first we have a *light* phase where the performance is almost constant (phase I);
- then a *critical* phase, where the performance degrades, occurs (phase II);
- thirdly the *sweet* phase where the performance increases again is observed (phase III);
- finally, when the model becomes under-parameterized, the *collapsed* phase is achieved (phase IV).

Worth of noticing, the last two phases are typically referred to as *classical regime*, where the traditional bias-variance trade-off occurs, while the first two are the *overparametrization regime*.

Better Low or Extreme Over-parametrization? There's currently a significant debate regarding whether simple techniques, such as early stopping, suffice to achieve good generalization performance. For instance, Rice *et al.* delve into extreme over-parametrization for adversarially trained deep networks [118]. They note that overfitting the training set significantly impairs robust performance in adversarially robust training across multiple datasets, a challenge that can be addressed simply by employing early stopping.

To combat overfitting, some works propose self-training to smooth logits, coupled with stochastic weight averaging trained by the same model [119], or by employing stochastic weight averaging independently [120]. While these studies primarily focus on DD, similar effects can be applied to sparse double descent. The question arises: is it always the case that the optimal model lies within the sweet phase?

In Fig. 2.7, we observe a correlation between the optimal model (denoted by a $\star$) and the level of noise present in the dataset. Specifically, on CIFAR-10, the optimal model falls within the sweet phase for ε values of 20% and 50%, whereas on CIFAR-100, it

resides in it only for $\varepsilon = 50\%$. Thus, empirical findings suggest a relationship between the extent of noise in the training set and the positioning of the optimal model.

2.4.2 An Entropy-Based Interpretation to the Sparse Double Descent

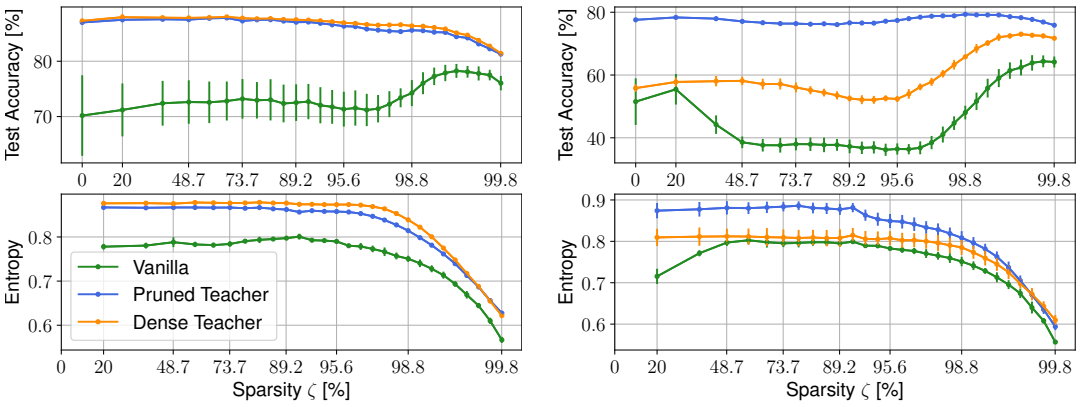
Examining the SDD phenomenon through the lenses of learning dynamics, we are prompted into the information bottleneck theory. This theory, initially proposed in the works by Tishby [121, 122], estimates the mutual information between the layers' processed information and the input and output variables. Consequently, it becomes feasible to compute optimal theoretical limits and establish benchmarks for generalization error. Several studies have embraced this theory, exploring variations in learning dynamics based on the choice of activation functions employed [123]. Furthermore, efforts have been made to refine estimation approaches, as evidenced by Pan *et al.* [124], while verifying that the theory accommodates DD in regression tasks [125].

Inspired by this literature, we can conjecture that as the model size, measured in terms of the number of parameters per neuron, decreases from the light phase, the entropy of the features within the model remains stable, requiring only minor fine-tuning for the parameters. However, upon exiting the interpolation regime immediately following the interpolation threshold and entering the sweet phase, the entropy begins to decline.

To empirically validate this conjecture, Fig. 2.7 also reports the entropy of the activations within the trained models. As it is possible to observe, the entropy is indeed stationary or even growing in the first two phases, while declining in the last two. Such behavior has been consistently observed across the typical datasets employed to study the SDD phenomenon. This observation enables back traditional early stop criteria for pruning: once the entropy of the activations starts declining, as we enter the classical regime, we can use again the known criteria to stop the pruning. This allows us to also capture the best fitting model, regardless whether it is located in the sweet or the light phase.

2.4.3 Distilling Knowledge to Avoid the Sparse Double Descent

While employing a standard ℓ_2 regularization method can indeed help mitigate SDD, it comes with its own set of drawbacks. Nakkiran *et al.* demonstrated that optimally-tuned ℓ_2 regularization can lead to monotonic test performance for certain linear regression models with isotropic data distributions, as both the sample size and model size increase [126]. However, as highlighted by previous research [22], in certain image classification setups such as ResNet-18 on CIFAR datasets, SDD remains noticeable even



(A) (B)

FIGURE 2.8: Performance of a shallow VGG-like model on CIFAR-10 for different label noises. Left: $\varepsilon = 20\%$. Right: $\varepsilon = 50\%$. Figure adapted from [4].

with the use of ℓ_2 regularization. This is due to the difficulty of optimally tuning the regularization and training policies [21].

One possible approach, inspired by recent literature in both pruning [127, 128, 129] and DD [130, 131], is to formulate a learning problem as a knowledge distillation one, where an over-parametrized teacher regularizes a shallower student one model. Since the (over-parametrized) teacher is in the light phase, it will embody good generalization, driving the student to a solution that generalizes well. Given that the student will be under-parameterized compared to the teacher, it will be forced directly into the sweet phase, avoiding SDD.

In a quantitative numerical validation (of which one part is reported in Fig. 2.8, while a more extensive analysis is reported in [4] for traditional convolutional architectures and in [21] for Transformer models), we observe that applying distillation effectively prevents a student to showcase SDD when pruned, which on the contrary happens in vanilla scenarios.

Although further research is necessary and ongoing in this domain, my recent findings suggest that the focus on model compression should transition from single parameters to neurons or even entire layers.

2.5 Adapters in Pre-trained Models

It will be here presented the work "Mini but Mighty: Finetuning ViTs with Mini Adapters" [14].

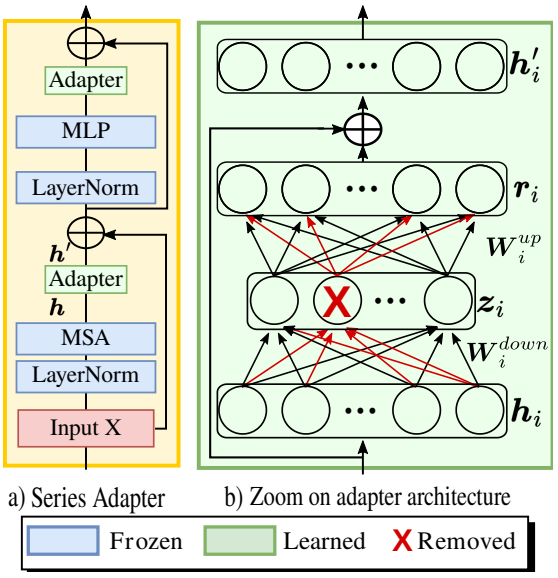
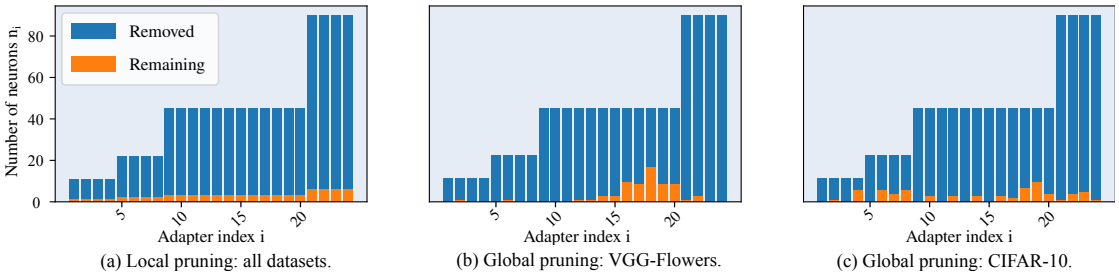


FIGURE 2.9: The adapter structure injected into ViT model. MSA and MLP are multi-head self-attention and feed-forward blocks, respectively. Image taken from [14].

With the uprisal of the new Vision Transformers (ViT) architectures [132, 133] and their increasing size, a new class of approaches, namely Parameter-Efficient Training (PET), have been popularized with the goal of adapting these large pre-trained models to new tasks, with just a marginal increment of their parameters [134, 135, 136]. Despite originally conceptualized for convolutional architectures [137, 138, 139, 140], they found one of the most appealing applications in ViTs. This is due to their lack of inductive biases, which makes their finetuning on new tasks easily susceptible to overfitting to fine-tuning tasks [141, 142].

Adapters Among PET methods, adapters [143] and its variants [134, 144, 145] are frequently employed for Natural Language Processing (NLP) tasks. In essence, adapters are compact modules integrated into transformer blocks (constituted of two sub-layers: a multi-head self-attention and a multilayer perceptron), typically applied after each sub-layer. Their primary function is to facilitate efficient adaptation of data representation for downstream tasks. Notably, adapters deliver comparable performance to full fine-tuning (involving the update of all parameters), yet they demand a minimal number of trainable parameters [143, 146].

More formally, let us consider the i-th adapter added to our pre-trained ViT, and $\boldsymbol{h}_i \in \mathbb{R}^{M_i}$ denote its input, of size M_i. According to [143], adapters employ a first fully-connected layer down-projecting $\boldsymbol{h}_i$ into $\boldsymbol{z}_i \in \mathbb{R}^{N_i}$ with some non-linear activation $\phi(\cdot)$. This is parametrized by a linear projection matrix $\boldsymbol{W}_i^{\text{down}} \in \mathbb{R}^{M_i \times N_i}$. Then, a second fully connected layer with parameters $\boldsymbol{W}_i^{\text{up}} \in \mathbb{R}^{N_i \times M_i}$ up-samples $\boldsymbol{z}_i$, producing as output $\boldsymbol{r}_i \in \mathbb{R}^{M_i}$. Finally, a residual skip-connection is employed inside the adapter module



(a) Local pruning: all datasets. (b) Global pruning: VGG-Flowers. (c) Global pruning: CIFAR-10.

FIGURE 2.10: Layer-wise analysis of adapter's neurons distribution. Bar plots represent the number of neurons n_i at each adapter i for VGG-Flowers and CIFAR-10, respectively. Global neuron selection leads to different neuron distributions depending on the dataset. Compared to VGG-Flowers, fewer adapters are completely removed on CIFAR-10. Image taken from [14].

such that, if $\boldsymbol{r}_i$ is close to zero, the whole adapter module degenerates to an identity function. The total number of parameters in the adapter is $2 \cdot N_i \cdot M_i$. A graphical representation of adapters is portrayed in Fig. 2.9.

Given that the size of an adapter is determined a-priori, it is here tackled the quest to properly size adapters for a specific downstream task. Leveraging on the peculiar architecture of adapters, it is indeed possible to evaluate the impact of removing specific dimensions (through structured pruning) on the computation flow of the entire backbone, sizing properly each injected adapter for a specific downstream task. In the following, it will be presented the core idea behind [14].

Finetuning ViTs with Mini Adapters at a Glance Based on the adapter design introduced earlier, we can devise a specific threshold function: if an entire row in $\boldsymbol{W}_i^{\text{down}}$ and an entire column in $\boldsymbol{W}_i^{\text{up}}$ are both zero, then our adapter is essentially equivalent to one with a smaller dimension M_i. Thus, we propose a scoring function that calculates the sum of the ℓ_1 norm of the corresponding row in $\boldsymbol{W}_i^{\text{down}}$ and the corresponding column in $\boldsymbol{W}_i^{\text{up}}$. Specifically, our importance score to guide the structured pruning of adapters is defined as follows:

$$\mathcal{I}^{ij} = \frac{1}{N_i + M_i} \left(\sum_{k=1}^{M_i} \left| W_{i,j,k}^{\text{down}} \right| + \sum_{k=1}^{N_i} \left| W_{i,k,j}^{\text{up}} \right| \right), \tag{2.15}$$

Differently from traditional structured pruning algorithms, this importance score encloses a "look-ahead" strategy: we observe, besides the output of a specific j-th neuron in the hidden space, also the impact of such an output in the next layer, in the hindsight of the final contribution to $\boldsymbol{r}_i$. This choice is empirically substantiated by numerous studies in the literature [92, 147, 148, 149]. Notably, $\mathcal{I}^{ij}$ is normalized by the total number of parameters associated with a specific adapter dimension: such a normalization facilitates fair comparison across adapters, even when they have different input and hidden layer sizes.

Fig. 2.10 reports the distribution of adapter's sizes for two different tasks, compared with a proportional greedy pruning as displayed in Fig. 2.10a. Different tasks prompt varying allocations of adapters within the same pre-trained ViT. This variability is motivated by factors like, for example, domain shift compensation. For instance, when a ViT architecture is pre-trained on ImageNet-1k, most adapters tend to concentrate on the last layers when the downstream task is VGG-Flowers, as the input distribution is similar to the upstream samples. Conversely, the adapter allocation extends closer to the input when the downstream task is CIFAR-10.

More details, experiments, and discussion can be found in [14].

2.6 Folding layers through pruning

It will be here presented the work "NEPENTHE: Entropy-Based Pruning as a Neural Network Depth's Reducer" [46] and The Simpler The Better: An Entropy-Based Importance Metric To Reduce Neural Networks' Depth [15]. This work builds on top of the preliminary results presented in "Can Unstructured Pruning Reduce the Depth in Deep Neural Networks?" [19].

In this whole Chapter, we have tackled the problem of parameter removal in DNN models through pruning. Unfortunately, at least intuitively, none of the aforementioned approaches is in general able to reduce the depth, namely the number of layers, in a DNN.

The impact of removing individual parameters or whole filters on recent computing resources, such as GPUs, in certain contexts can be considered marginal. Due to the parallelization of computations, the size of layers, whether larger or smaller, is primarily constrained by memory caching and core availability. The critical bottleneck in computation lies in the *critical path* that computations must traverse [150], a challenge that can be addressed by strategically removing layers. While some existing works implicitly address such concern, for example, employing knowledge distillation from deep teachers to shallower students [151], they fail to a-priori guarantee no performance loss (given that they impose a target shallow model), or avoid substantial perturbations. This motivates the exploration of designing an iterative pruning strategy, aimed at reducing the model's depth while preserving optimal performance.

Given the broad use of *rectifier* activation functions such as ReLU, GELU, and Leaky-ReLU, we can identify the average *state* of a given neuron for the trained task (in short, whether we are in the linear region(s)). From that, maximizing the utilization of one of

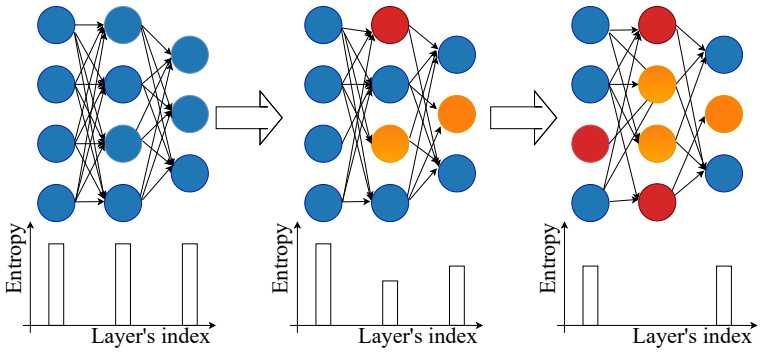


FIGURE 2.11: The average neuron's entropy calculated at the layer scale reduces as we induce some sparsity in the model. The main challenge is to make the average neuron's entropy go to zero for some layers, as it will be possible to remove it. Figure taken from [46].

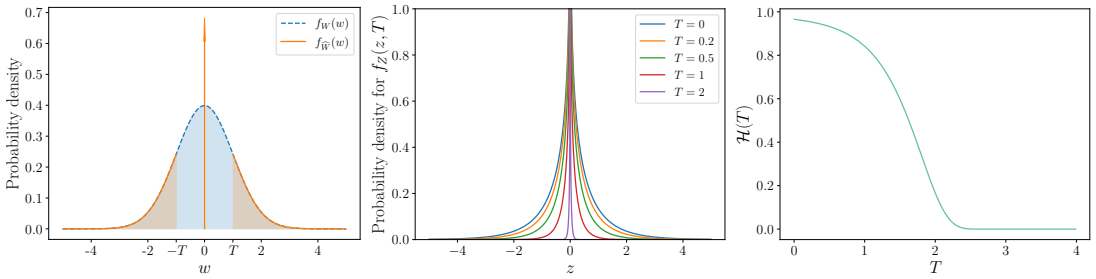
the regions, it is possible to design an approach able to force a target neuron to behave linearly through the state entropy's minimization (Fig. 2.11). It has also been observed that vanilla unstructured pruning is already implicitly minimizing such entropy, but is hardly able to completely force a whole layer to utilize one of these regions.

2.6.1 Unstructured Pruning Naturally Reduces the Entropy

Preliminaries. Let us assume $\phi(\cdot)$ is the rectifier of the l-th layer, populated by N_L neurons. We can monitor the output $y_{l,i}^{\boldsymbol{x}}$ of the i-th neuron from a given input $\boldsymbol{x}$ of the dataset $\mathcal{D}$ and write it as $y_{l,i}^{\boldsymbol{x}} = \phi(z_{l,i}^{\boldsymbol{x}})$. From this, we can define three possible "states" for the neuron:

$$s_{l,i}^{\boldsymbol{x}} = \begin{cases} +1 & \text{if } y_{l,i}^{\boldsymbol{x}} > 0 \\ -1 & \text{if } y_{l,i}^{\boldsymbol{x}} < 0 \\ 0 & \text{if } y_{l,i}^{\boldsymbol{x}} = 0 \end{cases} \tag{2.16}$$

More synthetically, for the output of the i-th neuron, we can easily identify in which of these states we are by simply applying the $\text{sign}(\cdot)$ function to $z_{l,i}^{\boldsymbol{x}}$, obtaining $s_{l,i}^{\boldsymbol{x}} = \text{sign}(z_{l,i}^{\boldsymbol{x}})$. Informally, we can say that the neuron is in the *ON state* when $s_{l,i}^{\boldsymbol{x}} = +1$ (as it is typically the linear region) while it is in the *OFF state* when $s_{l,i}^{\boldsymbol{x}} = -1$ (given that $\lim_{x \to -\infty} \phi(x) = 0$). There are a few exceptions to this, like LeakyReLU- in those cases, even though the activation will not converge to zero, we still like to call it OFF state as, given the same magnitude of input, the output's magnitude is lower. The third state $s_{l,i}^{\boldsymbol{x}} = 0$ is a special case, as it can be either mapped as an ON or OFF state. From the average over a batch of outputs for the neuron, we can obtain the probability (in the frequentist sense) of the i-th neuron of being in either the ON or the OFF states $p(s_{l,i}{=}{+}1)$ and $p(s_{l,i}{=}{+}1)$. From this, we can calculate the entropy of the i-th neuron



(A) (B) (C)

FIGURE 2.12: Distribution of a layer's parameters with magnitude pruning at threshold T (a), pre-activation distribution at varying T under the assumption of independence and centering of the Gaussian distributed input and layer's parameters (b), and entropy of the rectifier-activated neuron's output as a function of T (c), all in the large N limit. Figures adapted from [46].

in the l-th layer as

$$\mathcal{H}_{l,i} = -\sum_{s_{l,i}=\pm 1} p(s_{l,i}) \log_2 \left[p(s_{l,i})\right] . \tag{2.17}$$

With the definition in (2.17), $\mathcal{H}_{l,i}$ can be zero in two possible cases:

- $s_{l,i} = -1 \ \forall j$. In this case, $z_{l,i} \leq 0 \ \forall j$. When employing a ReLU, the output of the i-th neuron is always 0, and in this specific case, the neuron can be simply pruned.
- $s_{l,i} = +1 \ \forall j$. In this case, $z_{l,i} \geq 0 \ \forall j$. The output of the i-th neuron is always the same as its input,[1] this neuron can in principle be absorbed by the following layer as there is no non-linearity between them anymore.

By averaging the entropy values for the total number of neurons N_l inside the l-th layer, we can estimate the average entropy $\widehat{\mathcal{H}}_l$ of the neurons in the l-th layer. Since we aim to minimize the depth of deep neural networks by eliminating zero-entropy layers, we would like to have $\widehat{\mathcal{H}}_l = 0$. Unfortunately, directly minimizing the entropy in the optimization function is hard as it relies on non-differentiable measures. In the following paragraph, it will be shown that unstructured pruning naturally reduces (2.17).

Intuition Here we will suppress the layer and neuron indices (given that we will always consider the same entity). Let us assume the input $\boldsymbol{x}$ for a given neuron is a sequence of random variables $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$. Similarly, we can assume the N parameters populating such neuron, for a large N limit, follow as well a Gaussian distribution, and we model it as $W \sim \mathcal{N}(\mu_W, \sigma_W^2)$. Let us apply a magnitude-based pruning mask to the neuron's parameters, where we apply some threshold T. As such, we obtain a modified

[1] or very close as in GeLU

distribution for the layer's parameters:

$$f_{\widehat{W}}(w,T)=\begin{cases} \dfrac{1}{\sigma_W\sqrt{2\pi}}\exp\left[-\dfrac{1}{2}\left(\dfrac{w-\mu_W}{\sigma_W}\right)^2\right] & |w|>T \\ \\ \psi(T)\delta(w) & |w|\leq T, \end{cases} \tag{2.18}$$

where

$$\psi(T)=\frac{1}{2}\left[\operatorname{erf}\left(\frac{T-\mu_W}{\sigma_W\sqrt{2}}\right)-\operatorname{erf}\left(\frac{-T-\mu_W}{\sigma_W\sqrt{2}}\right)\right] \tag{2.19}$$

is the fraction of parameters pruned, or *pruning rate*, $\delta(\cdot)$ is the Dirac delta and $\operatorname{erf}(\cdot)$ is the error function. Fig. 2.12a displays an example of distribution when applying magnitude pruning having threshold T against the original distribution. Under the assumption of independent distributions, where $\mu_W=\mu_X=0$ and $\sigma_W^2=\sigma_X^2=1$, we can obtain the distribution for the pre-activation z (resulting from the product of the weights and the input, modeled through the random variable Z). According to the result obtained by [152, 153], it can be expressed as

$$f_Z(z,T)=\frac{1}{\pi}K_0\left[\left|\frac{1}{1-\operatorname{erf}\left(\frac{T}{\sqrt{2}}\right)}\cdot z\right|\right], \tag{2.20}$$

where $K_n(\cdot)$ is the n-th order modified Bessel function of the second kind. We can observe, from Fig. 2.12b, how $f_Z(\cdot)$ is affected by increasing the thresholding T. Now, let us assume the activation function of such a neuron is a rectifier function, and we are interested in observing what is the probability of the post-activation output being in the linear region: we are interested in measuring

$$p[Z>0]=\frac{1}{\pi}\int_{\epsilon}^{+\infty}K_0\left[\left|\frac{1}{1-\operatorname{erf}\left(\frac{T}{\sqrt{2}}\right)}\cdot z\right|\right]dz=\frac{1}{2}\left[1-I\left(\frac{\epsilon}{1-\operatorname{erf}\left(\frac{T}{\sqrt{2}}\right)}\right)\right], \tag{2.21}$$

where

$$I(x)=x[L_{-1}(x)K_0(x)+L_0(x)K_1(x)], \tag{2.22}$$

$L_n(\cdot)$ is the n-th order modified Struve function, and ϵ is a positive small value. From this, we can easily obtain the complementary probability $p[Z\leq\epsilon]$ and calculate the entropy between the two states. Due to finite numerical precision in the computation (especially employing mixed precision and fast inference algorithms for Deep Learning, but with standard IEEE 754 is $\approx 10^{-7}$), values below a given threshold can be approximated to zero and we model this through ϵ, which motivates our choice.

Fig. 2.12c displays the entropy as a function of the thresholding parameter T: as we observe, the entropy decreases given that the threshold increases: through unstructured

pruning, the neuron's output entropy is naturally minimized when employing rectified activations, even in the oversimplified case here treated. In the following, we will present how we are exploiting such a property of unstructured pruning towards layer entropy minimization.

2.6.2 A Layer Entropy-Aware Pruning Score

Driven by the promising theoretical results derived in Sec. 2.6.1, it is here proposed a relevance metric that will guide the unstructured pruning to lower the whole layer's entropy $\widehat{\mathcal{H}}_l$, named NEPENTHE. As we aim to increase the number of zero-entropy layers, intuitively more pruning should be applied to layers with lower entropy, as they are the best candidates to be removed. Concurrently, to minimize the impact on performance, only low-magnitude weights should be removed, as they are typically those providing the lowest contribution to the neural network's output [1, 3, 92]. To reach these two objectives, it is here defined an intra-layer's pruning irrelevance score $\mathcal{I}_l$:

$$\mathcal{I}_l = \frac{1}{N_l} \sum_{i=1}^{N_l} \widehat{\mathcal{H}}_{l,i} \cdot \frac{1}{\|\boldsymbol{\theta}_l\|_0} \|\boldsymbol{\theta}_{l,i}\|_1, \tag{2.23}$$

where $\|\boldsymbol{\theta}_l\|_0$ is the l-th layer's parameters cardinality (hence, not accounting for the already pruned weights). This metric accounts for the average parameter's magnitude and the layer's entropy at the same time: layers with few parameters but high entropy are less prone to be removed than layers with more parameters but lower entropy (under the same parameter's norm constraint). Besides, the parameter's magnitude of neurons with zero entropy is not accounted for in the importance score calculation. Symmetrically, to remove parameters from layers having lower pruning irrelevance, it is here defined the inter-layer's pruning relevance score $\mathcal{R}_l$ as

$$\mathcal{R}_l = \begin{cases} \dfrac{1}{\mathcal{I}_l} \displaystyle\sum_{j \in L} \mathcal{I}_j & \text{if } \mathcal{I}_l \neq 0 \\ \\ 0 & \text{otherwise.} \end{cases} \tag{2.24}$$

This measure becomes as large as the l-th layer's pruning irrelevance score is smaller, compared to the others.

Numerical Validation We propose here a preliminary evaluation assessed on ResNet-18 trained on CIFAR-10. A broader evaluation and experimental details can be found in [46]. Table 2.2 reports the entropy trend of the six layers showing the lowest entropy. The iterative magnitude approach (IMP) removes progressively, in this setup, the 50% of the parameters from the model, following a vanilla global unstructured magnitude

Approach	$\widehat{\mathcal{H}}_1$	$\widehat{\mathcal{H}}_2$	$\widehat{\mathcal{H}}_3$	$\widehat{\mathcal{H}}_4$	$\widehat{\mathcal{H}}_5$	$\widehat{\mathcal{H}}_6$	**top-1**
Dense	0.647	0.680	0.728	0.785	0.791	0.797	91.66
IMP (iter #1)	0.585	0.650	0.699	0.725	0.767	0.778	92.29
IMP (iter #2)	0.506	0.580	0.647	0.654	0.700	0.722	92.25
IMP (iter #3)	0.256	0.623	0.658	0.672	0.682	0.737	92.46
IMP (iter #4)	0.192	0.660	0.667	0.676	0.698	0.763	92.27
IMP (iter #5)	0.136	0.589	0.648	0.727	0.728	0.791	92.44
IMP (iter #6)	0.093	0.447	0.640	0.650	0.764	0.765	91.89
IMP (iter #7)	0.055	0.335	0.487	0.592	0.640	0.775	91.66
NEPENTHE	0	0	0	0.014	0.121	0.942	92.55

TABLE 2.2: Trend in the bottom six layer's entropies for ResNet-18 trained on CIFAR-10. Table taken from [46].

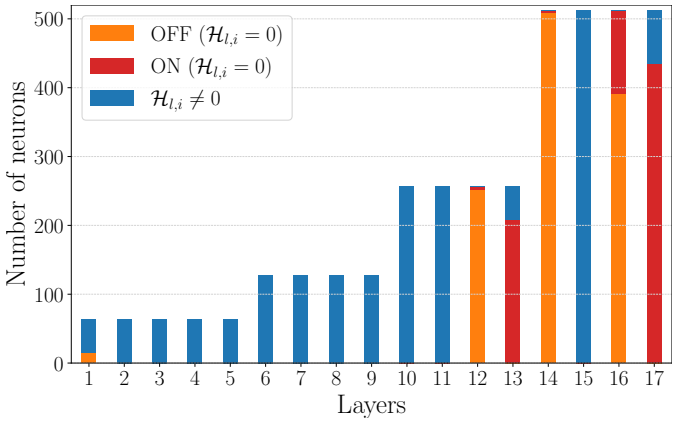


FIGURE 2.13: Distributions of neuron states per layer for ResNet-18 trained on CIFAR-10 pruned by NEPENTHE. In blue neurons having non-zero entropy, in orange always OFF, and in red always ON. Figure taken from [46].

pruning approach. As expected, as the pruning progresses (and implicitly T grows), the entropy is naturally decreased, showcasing very small values after some pruning iterations. However, we also observe that as the entropy $\widehat{H}_1$ decreases, the top-1 accuracy begins to deteriorate. This happens as there is no proper pruning re-allocation, that instead happens with NEPENTHE according to (2.24): indeed, in such case not only does the performance remain high, but we can successfully remove three layers from the model. Noticeably, $\widehat{H}_4$ and $\widehat{H}_5$ are also very low, while already starting from $\widehat{H}_6$ the entropy is very high. Contrarily to magnitude pruning where the entropy is in general in intermediate-range values, NEPENTHE tries to push all the encoded information toward layers having already high entropy, enabling effective layer removal with little (or in this case no) performance loss.

This is also illustrated in Fig. 2.13, showing the distribution of the neuron states per layer for ResNet-18 on CIFAR-10 trained with NEPENTHE. Our unstructured pruning approach effectively removes three layers by pushing all the neurons inside low-entropy layers to be either in the ON or in the OFF state. Besides, we also notice that in some

layers (like the 1, 13, and 17) there are entire units at zero entropy- we also achieve some structured sparsity by an unstructured approach, as already reported in some works [1, 92].

2.6.3 On-going Work for Layer Folding

The work illustrated in Sec. 2.6.1 and 2.6.2 shows it being successful in alleviating deep neural networks' computational burden by decreasing their depth. However, this method also presents some limits: compressing already parameter-efficient architectures that are not over-fitting remains challenging, and this approach is unable to reduce the depth of an already under-fitting architecture. Nevertheless, this research direction is new and less explored than traditional pruning and is recently gaining more and more momentum [154, 155, 156]. One current research direction under investigation involves the design of one-shot approaches to reduce the computational complexity, and the employment of Optimal Transport [157, 158] to eventually match the output distribution of different layers. Finally, another aspect to consider and not to underestimate is that it is underly assumed that reducing the depth of a DNN model will always speed up computation. Due to hardware optimizations, we know that having smaller kernels typically makes computation faster due to the effects of memory transfer: it is not obvious that, in GPU or TPU-equipped systems, a shallower DNN with larger kernels will be faster than a deeper one. A study on these effects is currently being conducted, evidencing some bottlenecks at the level of memory transfers.

2.7 Pruning Back-propagation: Neurons at Equilibrium

As hinted in Sec. 2.2.2.2, pruning a DNN model at initialization is hard; however, we know that progressively during training some neurons have learned their input-output function, meaning that they no longer need to be updated (and for instance, they might not require to have their gradient been calculated). Finding these (evidently without the full gradient calculation) leads to potential computation savings, without sacrificing the performance. Sec. 2.7.1 first analyzes and defines what is the concept of "neuron at equilibrium", and then Sec. 2.7.2 sketches the road to prospectively move the same notion to resource-constrained devices and potentially to hyper-parameters optimization.

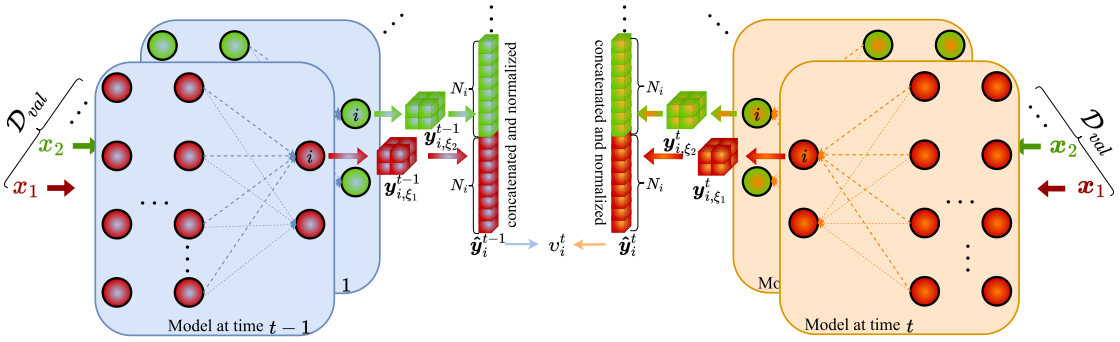


FIGURE 2.14: For a given time t the model (either in blue or orange) receives samples from the validation set (in red or green). These outputs are squeezed, concatenated and the obtained vector is then normalized, used to calculate υ_i. Image adapted from [6].

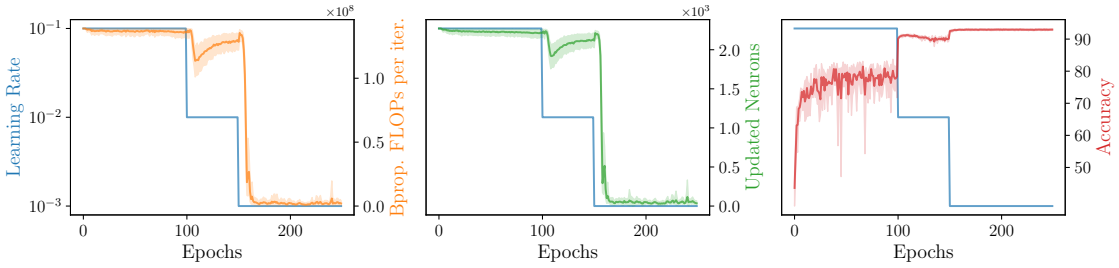
2.7.1 Neurons at Equilibrium

It will be here presented the work "To update or not to update? Neurons at equilibrium in deep models" [6].

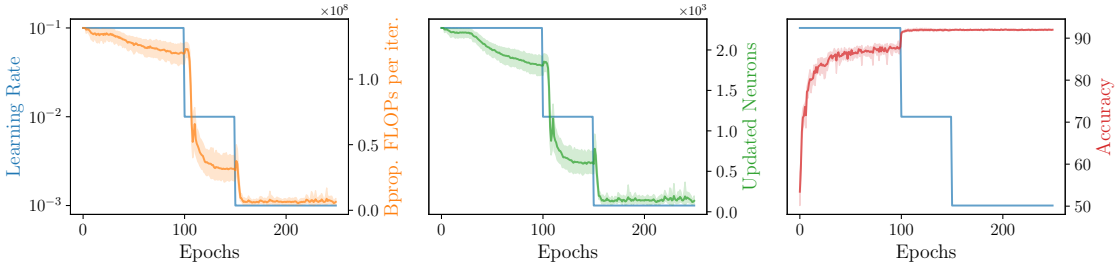
To save computation deriving from gradient computation (at a single neuron's level), we are here interested in assessing when the relationship between the input of the model and the output of the i-th neuron is modified during training. When this happens, we say the neuron is at *non-equilibrium*, meaning that its learned function, in the whole picture (or in other words, taking into account the evolution of the neurons in the previous layers as well), is still "evolving". We are interested in identifying the scenarios where the neuron is at *equilibrium* at the net of the interactions with the other neurons. To assess it, let us define the cosine similarity υ_i between all the outputs of the i-th neuron at time t and at time $t-1$ for the whole validation set $\mathcal{D}_{\text{val}}$. A schematic of this process is visualized in Fig. 2.14. When $\upsilon_i = 1$, the i-th neuron produces the same (eventually scaled) output between the evaluation at time t and at time $t-1$ for the same input $\boldsymbol{x}$ of the model. However, when we have that $\lim_{t\to\infty} \upsilon_i^t = k$, we say that the neuron is *at equilibrium*. When $k \neq 1$, this condition can be verified for example when working in recurrent neural networks, or even when the learning rate is too high.

To assess the convergence to equilibrium, it is possible to introduce the *variation of similarities* $\Delta\upsilon_i^t = \upsilon_i^t - \upsilon_i^{t-1}$ that can be even expressed as a velocity introducing a memory factor, to keep track of its evolution across the epochs.

The selection of the neurons at equilibrium is performed across a thresholding mechanism: all the neurons having a variation of similarities below ε (with ε modeling a tolerance typically in the order of 10^{-3}, as empirically assessed in [6]) are considered at equilibrium and their gradient computation can be avoided for the whole next epoch. We observe that this evaluation comes at the cost of memorizing the activations from



(A) ResNet-32 trained with SGD.



(B) ResNet-32 trained with Adam.

FIGURE 2.15: Back-propagation FLOPs (left, orange), updated neurons (center, green), and classification accuracy (right, red) for ResNet-32 trained on CIFAR-10. Image taken from [6].

the previous evaluation, and it costs one forward propagation on $\mathcal{D}_{\text{val}}$ plus the cosine similarity evaluation.

A set of both quantitative and qualitative experiments is reported in [6], which validate the approach, able to scale down the FLOPs demand for back-propagation up to more than 60% when training on Transformers, and observing that some optimizers like Adam, given their specific formulation, lead more neurons to equilibrium states than SGD (Fig. 2.15). Finally, we observe that the size of the validation set can be relatively small for the neuron equilibrium estimation (for image classification tasks it can be down one image per class only). This is due to the larger sampling happening in convolutional layers, where one image already produces multiple outputs.

In follow-up works, it has also been shown that the threshold ε can be also made learnable: leveraging on [159], we can apply a gradient directly to it already at training time, and tuning it in a loss-informed fashion. This indeed leads to even higher savings in terms of computation without performance loss [24].

2.7.2 Follow-ups of Neurons at Equilibrium

I present here two follow-ups opened by neurons at equilibrium: one is more oriented to hardware-constrained optimization, while the other is linked to hyperparameters-free optimization.

2.7.2.1 Neurons at Equilibrium with a Memory and a Computation Budget

It will be here presented the work "Towards On-device Learning on the Edge: Ways to Select Neurons to Update under a Budget Constraint" [26].

The field of on-device learning is experiencing a rapid growth, driven by the proliferation of embedded devices for IoT applications. Currently, the predominant approach involves offline training of models followed by compression and deployment on-device solely for inference. However, this approach often leads to suboptimal performance due to shifts in real data distribution compared to the training data [25]. Embedded devices face significant constraints in computational and memory resources during training, making it particularly challenging to train models. In on-device learning utilizing backpropagation, two main approaches are often combined to optimize training accuracies under resource constraints: enhancing the efficiency of the architecture and implementing *sparse updates* schemes. Focusing on the latter, there are primarily two prevailing approaches: statically determining the backpropagation graph or allowing for its dynamic evolution.

A prominent example in the first category is undoubtedly the work by Lin *et al.*, where they introduce an innovative algorithm-system co-design framework named Sparse Update [160]. Overcoming the memory constraints of IoT devices, they successfully achieved on-device training using just 256 kB of memory. This was made possible through a predefined selection of layers and weights to update.

In one of my recent works [26], it is explored the second category of approaches, employing Neurons at Equilibrium [6] as a metric to determine which neurons to update. In contrast to the approach outlined in Sec. 2.7.1, the purpose is not to exclude neurons at equilibrium here. Instead, it is prioritized updating neurons starting from those mostly in a non-equilibrium state, and continuing until the memory and computation budget is reached. While this approach may not guarantee optimality, according to the preliminary analysis reported in [26], on average it outperforms a static update scheme like SU, thanks to its dynamic adaptivity to the specific downstream task. The preliminary experiments are highly promising in this regard, illustrating the greater potential of a dynamic update scheme compared to a static one.

2.7.2.2 Estimating Neural Velocity to Scaling the Learning Rate

Hyperparameter tuning plays an important role in a successful neural network's training. Among these are, for example, the learning rate and the policy used to adjust it over time. On the other hand, choosing the right stop conditions, namely the conditions to be met to declare early completion of training is also important to avoid potential overfit.

Hyper-parameters can be tuned in different ways, from manual initialization to computationally expensive grid search. Yet, finding suitable hyperparameters requires holding some validation samples out of the training set. Holding out samples from training entails however several drawbacks. First, the model's learning capability is reduced, especially when few annotated samples are available. Second, the validation set may not represent the true distribution, leading to inexact hyperparameter tuning [161]. For these reasons, there is a lot of interest nowadays in validation-less learning approaches.

One work-in-progress is to tune the learning rate decay policy and the stopping criterion without needing a held-out validation set. From the intuition as in Sec. 2.7.1, it is possible to derive a neural velocity as the trend of neurons to "change" during training. If the velocity drops to zero, then the neuron itself has converged to a stable solution, and the training can be terminated: this property of the proposed velocity definition allows one to cut the training times. One intriguing possibility is to assess the neural velocity without necessarily resorting to a validation set but rather randomly sampling noise to the input, making this estimation data-free in contexts when little data is indeed available. One speculation here is that on overfitting regimes the velocity is very low but not null - identifying such a trend on the validation set, and properly characterizing the threshold ε to apply, can be the key to success.

Chapter 3

Biases in Deep Neural Networks

In this Chapter, the problem of debiasing will be treated. Although part of the literature is quite old and deepened its roots way before the advent of Deep Learning, the problem setup revolving around DNN debiasing is relatively new and requires proper contextualization. Given that my works are mostly oriented toward Computer Vision and that a big bulk of works focus on this area, the focus here will be mostly on the problem of debiasing for image classification. It is indeed known that Large Language Models (LLMs) and Generative Models are (or in general, can be) affected by the problem of biases in generation [162]: however, the same concepts that will be explored here can be transposed in more advanced setups.

3.1 Structure of the Chapter

In this Chapter, I will first introduce the problem and threat represented by biased predictions and, more in general, outputs (Sec. 3.2). Then, I will provide one possible categorization of debiasing approaches commonly employed by the literature (Sec. 3.3). Roughly, it is possible to divide them into supervised and "unsupervised" approaches. After these preliminaries, I will present my contributions, summarized as follows.

- It has been developed a *supervised debiasing* approach, named Entanging and Disentangling (EnD) [7]: in classification tasks, it entangles unbiased representations of the same target while disentangles representations of the same bias (Sec. 3.4). This work had a follow-up, that improved hyper-parameter tuning and provided an interpretation in the metric space [8].
- Contrarily to a standard approach taken by part of the literature [163, 164, 165], I have shown that removing entirely the information related to the bias might harm

the performance of an unbiased classifier [30]. More specifically, a proper debiasing approach *re-weights the information* related to the bias (Sec. 3.5).

- Inspired by [30], it has been proposed an *unsupervised debiasing approach* [9]. Specifically, when observing the latent space at the output of the encoder, there is one specific learning iteration when the bias is maximally fitted. It is possible to identify such a moment by observing the distances of misclassified samples (on the training set) by looking at distances from the Voronoi boundary of the target class. It is possible to extract at this point the information on bias-target alignment (the underlying assumption here is that the misclassified samples do not have the dominant bias for the target class) and it is possible to apply a simple strategy to train an unbiased model (Sec. 3.6).

3.2 The Threat of Biases

Recently, AI trustworthiness has been recognized as a major prerequisite for people and societies to use and accept such systems [166, 167]. In April 2019, the High-Level Expert Group on AI of the European Commission defined the three main aspects of trustworthy AI [166]: it should be:

- *lawful* - respecting all the regulations and laws;
- *ethical* - respecting all the basic ethical principles and values;
- *robust* - technically and accounting for the social environment.

Providing a warranty on this topic is currently a matter of study and discussion.

Regarding AI robustness, Attenberg *et al.* examined the challenge of identifying the so-called "unknown unknowns" within data [168]. These unknown unknowns represent instances where DNNs process information in unintended ways while exhibiting high confidence in their predictions. Such behavior has impacted numerous recent AI-based solutions, including those aimed at COVID detection from radiographic images. Regrettably, some of the early datasets available at the begin of the pandemic were often heavily biased [31]. Consequently, models frequently displayed overconfident predictions of COVID diagnoses due to the presence of unwanted biases. These biases included the detection of catheters or medical devices for positive patients, patient age (given that at the beginning of the pandemic, most infected individuals were elderly), or even the identification of data origins (where negative cases were augmented using samples from unrelated datasets) [31, 169, 170].

Learned biases can indeed harm the generalization ability of Deep Neural Networks (DNNs) [163, 164, 171, 172, 173, 174]. For instance, in the context of image classification tasks such as detecting pedestrians, if environmental cues (e.g., the presence of a sidewalk/crosswalk - Fig. 3.1) become spuriously correlated with the target classes, neural networks may exploit such correlations as *shortcuts* for classification [175], thereby leading to performance degradation when presented with images containing different backgrounds (e.g., pedestrian crossing the road not on pedestrian crossing lanes).

FIGURE 3.1: Pedestrian on the crosswalk (left) and on the road (right). If the model assumes pedestrians can be on the road if there is a crosswalk, it can have false negatives.

Many existing debiasing methods rely on prior knowledge about the bias, such as the presence of auxiliary labels indicating side information or the characteristics of the bias [7, 8, 163, 172, 176, 177]. However, obtaining these labels or information about the bias can be prohibitively expensive due to annotation costs or highly noisy, prompting the development of bias-agnostic approaches. Recent studies have revealed that bias features are often learned early in the training process [174, 178]: there are samples where the bias is learned alongside the target, leading to improved performance on the training set, and others where the bias leads to misaligned predictions. Bias-agnostic approaches typically leverage biased information from the training set by amplifying the initial features learned using methods like Generalized Cross-Entropy [179] and then discouraging their learning in an "unbiased" model. However, there's no guarantee that the earliest features learned are indeed the biased ones, posing a challenge in detecting and effectively addressing bias agnostically.

3.3 Overview on Debiasing Approaches

In this section, I briefly review state-of-the-art techniques designed to prevent models from learning biases. The techniques can be grouped into (but not limited to) two main categories: supervised approaches, where the information related to the bias is made available, and unsupervised ones, when such information is not shared at training time.

3.3.1 Supervised Debiasing Approaches

Supervised debiasing methods are typically divided into three categories: pre-processing methods, which modify the dataset before classification; in-processing methods, which modify the learning process of the model; and post-processing methods, which directly modify the output of the DNN.

Preprocessing Methods Among the most used preprocessing methods in the literature, driven data augmentation plays a prominent role. Generative Adversarial Networks (GANs) are widely used to generate realistic images: StyleGANs [180] is indeed one of the most used GANs in this context. For example, Kang *et al.* used it to generate handwritten text in specific styles [181]. In image classification, Geirhos *et al.* used style transfer to augment ImageNet with texture-bias-conflicting elements to create a more texture-balanced dataset [173].

Postprocessing Methods These methods have the advantage of neither re-training models nor requiring additional data for the training. With their Reject Option Classification, for example, Kamiran *et al.* proposed to take the samples classified with the most uncertainty (outside a predefined confidence margin) and to change their class to decrease the Disparate Impact metric [182]. In this same context, Equalized Odds Post-processing proposed by Hardt *et al.* maximizes the Equalized Odds metric [183]. Despite the potential advantages of these approaches, a major drawback lies in the low degrees of freedom for the corrections (since they can only access post-classification information), which limits their practical effectiveness.

In-processing: Debiasing within Training Most of the debiasing methods in the literature work directly on the model, learning from a biased dataset. In general, unbiased elements are weighted more than biased elements. This simple yet effective approach is nowadays very popular in supervised setups [184]. Other methods tackle supervised debiasing by adding regularization terms during the training of the deep model, which is the case of methods such as the proposed EnD [7] (and presented in Sec. 3.4) and FairKL [8]. Another intuitive approach relies upon simply removing the biased features from each sample in the dataset and performing the so-called *fairness by blindness*. However, the phenomenon known as *encoding redundancy* [183] states that information is very rarely encoded only once in the data [185], so removing a single value or label is probably not sufficient to remove the effect of the bias on classification.

3.3.2 Unsupervised Debiasing Approaches

Some recent methods do not rely on bias labels because they can be difficult to obtain on real-life datasets and we will refer to them as "unsupervised" or "bias-agnostic". All of these approaches follow a general scheme, which is typically divided into two phases: *bias inference*, where a first model, often called "bias capturing", aims to capture biases in the data; and *bias mitigation*, where a second model is trained to avoid the biases captured by the first model. These approaches rely on prior knowledge, which may be more or less specific to the target task.

Bias in the Texture Texture bias is a significant concern in image classification, leading some research to specifically target it [173]. Rebias [163], for instance, focuses on learning representations that significantly differ from those obtained using small receptive fields in convolutional layers, which are inherently biased towards learning specific textures. Addressing the same challenge of texture bias, HEX [177] suggested utilizing the gray-level co-occurrence matrix to encourage representations that are color-independent.

Bias Generates Imbalances between Groups Some unsupervised approaches involve identifying bias groups that optimize certain fairness metrics and training models to have representations orthogonal to those inferred by the biases. For instance, DebiAN [186] proposed by Li *et al.*, alternates between training bias-capturing and unbiased models, with the goal of minimizing the Equal Opportunity fairness metric. Similarly, EIIL [187] introduced by Creager *et al.*, identifies biases by maximizing the violation of an invariance principle measured by the objective function IRMv1 [188]. PGI [189] builds upon EIIL by minimizing the KL divergence of predictions over these groups.

Bias is Learned Early Some recent methods operate under the assumption that bias features are easy to learn, particularly in the first stages of training. With LfF [174], Nam *et al.* proposed a loss reweighing technique based on this premise: they train a biased neural network and enhance the predictions from its early stages (where the notion of "early stage" remains however blurry and stays as a hyper-parameter). Concurrently, they train a debiased model by amplifying the weights of "difficult samples". Building on this idea, DFA [178] introduced by Lee *et al.* employed data augmentation to disentangle bias features from intrinsic features using latent representations from bias-capturing and unbiased models. Similarly, LWBC [190] by Kim *et al.* directs the training of their primary classifier towards the most challenging samples identified by

their classifier committee. Lastly, with PGD [191], Ahn *et al.* utilizes the magnitude of sample gradients as a metric to increase their importance. Building on the shoulders of the findings in this context, it has been developed a strategy to detect biased samples by looking at their distribution in the latent space. The underlying assumption here is that indeed the bias is learned early- hence the misclassified samples in early learning stages are the most informative to extract such information (as will be discussed in Sec. 3.6).

3.4 Entangling and Disentangling Deep Representations

It will be here presented the work "EnD: Entangling and Disentangling deep representations for bias correction" [7].

In this study, it has been introduced a supervised debiasing regularization technique named "EnD", which aims to disentangle biased features while intertwining deep features extracted from patterns belonging to the same target class. The objective of EnD is to mitigate the propagation of bias within the DNN. Interestingly and differently from a portion of the literature, while we acknowledge the presence of the bias through a label on the data sample, we stay in principle agnostic regarding the nature of these biases (whether they manifest as specific colors, image features, or otherwise, that would enable other pre-processing strategies).

EnD operates by regularizing the output of a specific layer (typically the output of the encoder) within the deep model to establish an information bottleneck. This regularization process entangles feature vectors extracted from data belonging to the same target class while disentangling features extracted from data labeled with the same bias label. By training the deep model to minimize both the loss and EnD simultaneously, biased features are discouraged in favor of unbiased ones.

EnD at a Glance Our primary objective is to train our model to accurately classify data into the possible classes while mitigating the influence of bias features present in the data. To achieve this, we intend to introduce an information bottleneck, limiting the use of bias-related information for the target classification task.

Given that $\boldsymbol{y}_i$ is the output vector for a given data sample from the encoder (we suppress here the layer index), after a vector normalization step converting it to $\tilde{\boldsymbol{y}}$, we can construct a similarity matrix between all the samples in the training (mini) batch, denoted as $G = (\tilde{\boldsymbol{y}})' \cdot \tilde{\boldsymbol{y}}$, where $(\cdot)'$ signifies the transposed matrix and $\tilde{\boldsymbol{y}}$ represents per-representation normalization. G is a special case of a Gramian matrix, where each

$g_{i,j}$ falls within the range $[-1, +1]$, indicating the difference in direction between any two vectors $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. The matrix G exhibits several properties:

- it is symmetric and positive semi-definite;
- all elements on the main diagonal are exactly 1 by construction;
- if the subset of outputs $\tilde{\mathbf{y}}$ forms an orthonormal basis (or G is full-rank), then $G = \mathbb{I}$ by definition.

Leveraging these properties, we formulate the regularization strategy, which comprises two terms:

- a *disentangling* term, whose task is to try to de-correlate as much as possible all the patterns belonging to the same bias class b;
- an *entangling* term, which attempts to force correlations between data from different bias classes but having the same target class t.

More details on the formulation for these terms and on the numerical experiments can be found in [7]. Specifically, it is found that the disentangling term (that acts as a bias information remover) should contribute less than the entangling term (that weights more the information extracted by samples that are unbiased). Empirical results validate the effectiveness of this approach, which however has a major drawback: properly tuning the balance between entangling, disentangling, and loss on the target task can be very delicate. This issue has been solved through more recent work [8]. In this, the entangling and disentangling terms are formulated following the contrastive learning approach, assuming that biased samples belonging to the same class are *negative* (mimicking the disentangling term) and unbiased ones *positive* ones. The introduction of a margin between these samples models how strong the entangling/disentangling constraint should be, making the hyper-parameter optimization process much simpler.

3.5 Is Debiasing Equivalent to Information Removal?

In this section, I briefly present the problem of information removal in deep learning. It is possible to group it into two large families: privacy preservation approaches and debiasing techniques.

Privacy Preservation Recently, with advancements in computational capabilities, there has been a surge in research focusing on privacy preservation within computational frameworks. A study by Dwork *et al.* investigated the amount of noise required to ensure "differential privacy" [192] in data. Duchi *et al.* formalized convergence boundaries for training and explored the trade-off between privacy guarantees and the utility of resulting statistical estimators [193].

This knowledge has also been applied to deep learning frameworks, as demonstrated by Abadi *et al.* [194], who introduced tuned noise in the update rule to preserve privacy. Another approach to safeguard data privacy is *federated learning*. In this approach, private datasets are owned by the data proprietors, who train local neural network models directly. The model parameters are then transmitted to a master node, which distributes the general parameter configuration to all private computational nodes. This method, proposed by Shokri and Shmatikov, should enable parallel and private computation [195]. However, it does not address ethical biases such as gender or race; its primary guarantee is the non-disclosure of original data, though some sensitive information may still be exposed.

How Close is Debiasing to Information Removal? In some contexts, neural network debiasing implies a form of information removal. As an example, in HEX [177] the bias is identified within the texture and is explicitly removed prior to the learning process itself. Some works suggest the use of GANs to entirely clean up the dataset with the aim of providing fairness [196, 197], while others insert a GAN in the middle of the architecture to clean up the internal representation of data, purifying datasets from the bias [198]. At the architectural level, as also stated in [174], works like [163] force debiased models to learn a set of bias-independent features from the model. Specifically, in their work, Bahng *et al.* develop an ensembling-based technique called *ReBias*: this consists of solving a min-max problem where the target is to promote the independence between the network prediction and all biased predictions. Even works like [164] claim to remove entirely the information of the bias from the feature embedding, leveraging adversarial learning and gradient inversion. Similar claims are made by Thong *et al.*, where the bias mitigation problem is addressed by discouraging the optimization directions that favor the classifier to be biased [165].

For such a segment of literature, there is the (implicit) belief that debiasing and removing the information related to the bias are essentially the same concept. In the next section, through the proper design of an approach that *removes* specific information flowing inside the DNN model (IRENE), I evidence that debiasing approaches can entirely remove the information related to the bias, but that at the same time, such a

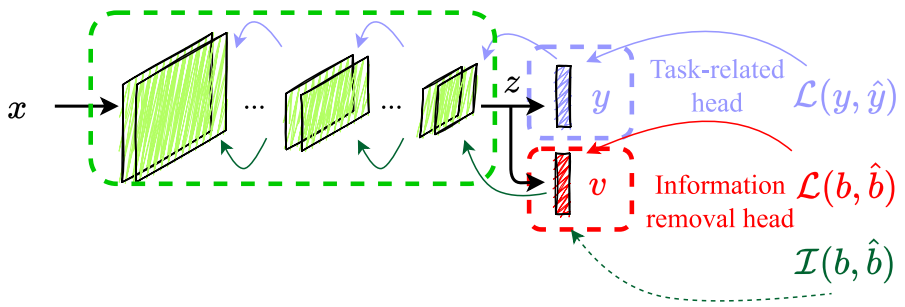


FIGURE 3.2: Schematics of IRENE. Image adapted from [30].

choice can be sub-optimal (Sec. 3.5.1). The purpose of debiasing is indeed just to *balance* the utilization of the information related to the bias to other features.

3.5.1 IRENE: Information Removal at the Bottleneck

> It will be here presented the work "Information removal at the bottleneck in deep neural networks" [30]. A different strategy sharing the same goals and coming as a natural evolution of [7], "Disentangling private classes through regularization" [10], will not be presented.

In this section, I introduce IRENE, a technique designed to eliminate targeted information at the bottleneck (which in our case is the output of the encoder) of DNNs. This approach estimates the information deemed "private" using an auxiliary classifier, enabling the assessment of the information to be removed at the bottleneck. Subsequently, information removal is accomplished by minimizing a differentiable proxy of the mutual information between bias labels extracted at the bottleneck.

IRENE at a Glance Let us associate to each input sample $\mathbf{x}$ a target output $\hat{y}$ and a companion ground truth label $\hat{b}$, marking a piece of different information from the task-related one (as it could be the information related to the bias). The objective here is to prevent its propagation for solving a specific task (like classification). To this end, after defining $\mathbf{z}$ as the output of the encoder, we would like to minimize $\mathcal{I}(z, \hat{b})$, where $\mathcal{I}(a, b)$ is the mutual information between a and b. Directly minimizing this quantity is computationally unfeasible: I overcome this difficulty by distilling from z how much information related from $\hat{b}$ is filtering employing an auxiliary classifier.

The whole training procedure, as synthesized in Fig. 3.2, includes the minimization of three terms.

- The loss $\mathcal{L}(y, \hat{y})$, to be minimized to train the model to learn the target task.

TABLE 3.1: Extract of results for the CelebA dataset (taken from [30]). Here the gender is the information to remove.

Target	**Method**	**Prediction accuracy (trained task) [%](↑)**	**Gender prediction accuracy (information to remove) [%](↓)**
Blond hair	RUBi [172]	95.29±0.14	88.55±1.22
	Rebias [163]	**95.59±0.11**	88.50±3.78
	LearnedMixin [176]	90.01±2.66	74.09±2.66
	IRENE	95.24±0.29	**53.58±10.71**
Heavy makeup	RUBi [172]	**90.40±0.08**	95.17±1.11
	Rebias [163]	90.28±0.34	93.78±2.55
	LearnedMixin [176]	84.88±3.28	68.09±10.55
	IRENE	83.31±3.41	**51.98±9.56**

- The loss $\mathcal{L}(b, \hat{b})$, to be minimized to train the information removal head to extract all the information about $\hat{b}$ from the bottleneck z.
- The mutual information $\mathcal{I}(b, \hat{b})$, to be minimized to accomplish our purpose of erasing the information from the bottleneck z of the model. It is evidently key not to update the parameters within the information removal head.

In [30] it is conducted an empirical analysis on the common setups for the debiasing community. Evidently, in simple setups where the bias and the target features are spatially and conceptually disentangled (like the background color and the shape of a number for the BiasedMNIST dataset), completely removing the information of the color from the bottleneck benefits the generalization ability of the model. However, in more complex setups, like recognizing the attribute "heavy makeup" in the CelebA dataset removing the information of the gender, makes the task to be solved much harder, harming the generalization of the model (as also reported in Tab. 3.1). This shows that performing model debiasing does not mean removing the information of the bias, but rather *re-weighting* it.

3.6 Unsupervised Debiasing by Looking at the Bottleneck

It will be here presented the work "Mining bias-target Alignment from Voronoi Cells" [9].

I present here an unsupervised debiasing technique that determines the best timing for extracting bias-target alignment information by observing, at the output of the encoder, the relative distances of misclassified samples to the Voronoi boundary of the correct target class. Leveraging this insight, it is possible to train an unbiased model by assigning

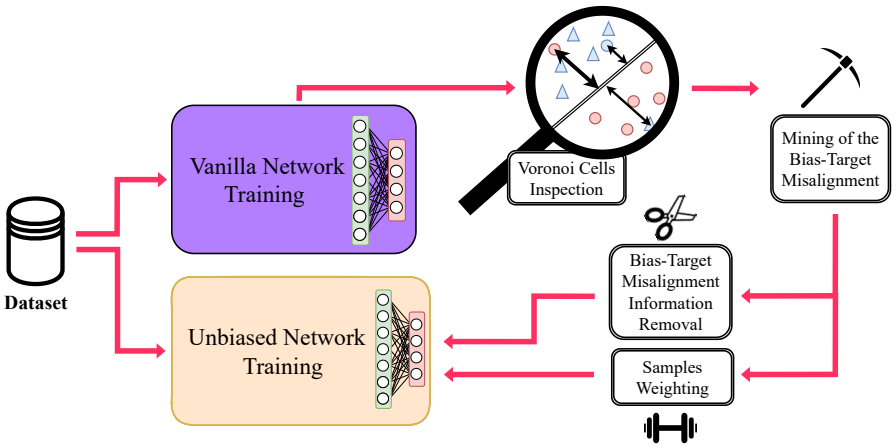


FIGURE 3.3: Overview of Mining bias-target alignment from Voronoi cells. Image taken from [9].

greater weight to bias-misaligned samples and mitigating bias-alignment information from the bottleneck layer.

The key contribution of this approach lies in its bias-agnostic nature, which guides the extraction of bias-target alignment information during the training of a standard model. Specifically, we extract this information when the distances of misclassified samples to the Voronoi boundary of the target class are maximized. Subsequently, we utilize the bias alignment information to adjust the loss contribution of each sample, thereby facilitating the learning of misaligned samples. Notably, since no prior information regarding the bias is a-priori given, through the whole process we are unable to associate a semantic meaning to it - this is currently a work in progress.

Mining Bias-target Alignment at a Glance Let us consider a supervised learning setup (but unsupervised bias-wise), where we have a dataset $\mathcal{D}$, where each sample $\boldsymbol{x}$ is associated with a ground truth target label y. A given deep neural network, trained for e epochs, produces a certain output trying to match the desired one through the minimization of a loss function $\mathcal{L}(\cdot)$. Unfortunately, this learning process does not impose any prior on the specific subset of features that are extracted, which can lead to a biased prediction over unseen data. We want to fight this effect. As synthetically displayed in Fig. 3.3, the approach is composed of two main steps.

First, the bias is inferred by the learning of a vanilla model: at the end of each epoch (or after a few iterations), the target class centroids and the Voronoi boundaries between them are computed from the well-classified samples at the bottleneck layer. The distance of the misclassified samples to the Voronoi boundary is computed to find the epoch e^* when the bias-target alignment is maximally learned. As visualized in Fig. 3.4, missing the exact extraction time results in having a model fitting on other features and

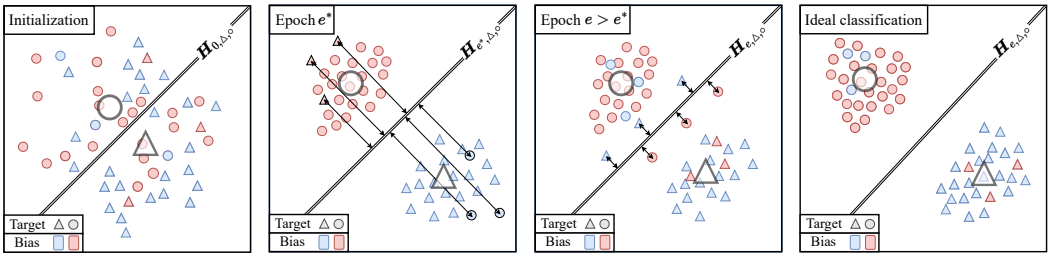


FIGURE 3.4: Representation of latent representations of a dataset at different learning stages. The target classes are shapes and the bias is color. Arrows represent the distance between misclassified elements and the Voronoi boundary (here represented by $\boldsymbol{H}$). Image taken from [9].

not exclusively on the bias, making the bias extraction process harder. To distinguish between bias-misaligned samples and bias-aligned ones, we assume that, after a few learning steps, the farther a misclassified sample is from its target Voronoi cell (defined in the bottleneck layer's output space), the more it has been strongly pulled by an attractor. Such an attractor, since it is not its target class centroid, can be considered as resulting from some bias. Hence, when the average distance between the misclassified samples and their target Voronoi cell reaches its maximum, the model *has learned bias features*, selecting the epoch e^*. At this point, we can collect the bias-target alignment information (saying, that all the correctly classified samples are bias-target aligned, while the misclassified ones are bias-target misaligned).

Then, a debiasing process follows: from the distances gathered from the previous step, we assign each sample a weight, which will be used in the weighted loss. In addition, at the bottleneck layer, we minimize the information about bias misalignment: this favors the unbiasedness of the classification head (it is here used a similar approach as IRENE, described in Sec. 3.5.1).

A complete set of experiments validating this approach is presented in [9]. Across empirical validation, we find that the proposed approach is not only state-of-the-art among the

TABLE 3.2: Results on CelebA, targeting the attribute "blond", with a bias towards gender. In yellow is the best supervised approach. Table taken from [9].

Method	**Bias agnostic**	**Test accuracy** [%] (↑)	
		Unbiased	Bias-Conflicting
Vanilla	✓	79.0	59.0
EnD [7]	✗	86.9	76.4
LNL [164]	✗	80.1	61.2
DI [199]	✗	90.9	86.3
BiasCon + BiasBal [200]	✗	91.4	87.2
Group DRO [201]	✓	85.4	83.4
LfF [174]	✓	84.2	81.2
Mining bias-target alignment	✓	**90.2±1.1**	**84.5±2.0**

unsupervised debiasing approaches, but in many setups, it stands among supervised approaches, and in some cases even shows superior performance. This effect can be due for example to imperfect bias label annotation employed in supervised scenarios. It is here reported, in Tab. 3.2, the performance achieved on the CelebA dataset for the "blond" attribute in dataset unbalanced training. In comparison even with non bias-agnostic (supervised) approaches, Mining bias-target alignment places itself competitively with other supervised approaches.

Chapter 4

Conclusion and Future Research

In this manuscript, I have presented some of my research since the end of my Ph.D. Specifically, I have worked on two core aspects of Deep Learning: making models efficient, and understanding how the information is processed and imposing restrictions to it. This translated into two research branches I have led in the last five years.

In the first branch, began at the end of my Ph.D. [2], I took care of exploiting *efficiency* for DNNs through the concept of pruning. Along such a path, it was important to explore alternatives to magnitude pruning [92] and its variants [72, 73], countering the lack of explicit loss constraints. In such a spirit, a sensitivity-based approach that senses the perturbations of the output when some specific parameter (or groups of parameters) are removed [1, 3, 14] has been proposed. Besides, it was also explored the challenging case when pruning is performed on noisy datasets, giving rise to the *sparse double descent* phenomenon [91]. This unfortunately challenges all the traditional pruning schemes, as early stop criteria might be suboptimal. To counter this, a framework that enables-back these schemes has been developed, through the employment of an entropic measure of the activations in the DNN, that discerns the occurrence of the classical bias-variance trade-off phase from the overparametrization regime [21]. Besides, it was shown that it is possible to avoid sparse double descent when resorting to knowledge distillation schemes [21]. On the exciting wave of the lottery ticket hypothesis [74], perspectives of zero-shot pruning (namely, pruning already at initialization) have been explored. Despite some works suggesting this as a concrete possibility [95, 202], in my works it was validated that iterative algorithms are still more performant [20]: due to a projection caused by pruning, the loss landscape becomes progressively flatter, making the optimization process hard [23]. However, locking the optimization process for a subset of parameters while allowing those having high error signal, can be an effective way of

saving computation at training time: through the estimation of equilibrium for neurons, it is possible to save computation when backpropagating the error signal [6].

The second branch, regarding *feature selection*, attracted my interest while working within the European project DeepHealth, between 2019 and 2021. There, I was collaborating on three use cases, and all of them were medical tasks: detection of colorectal cancer, acute ischemic stroke treatment, and lung cancer diagnosis. Besides, during the COVID pandemic, I collaborated within the CLAIRE COVID initiative, working on data provided by our local hospitals. From this experience, I got in touch with the problem of bias in data: the very first benchmarks for detecting COVID from Chest X-Ray images with DNNs were built by taking negative samples from the Chest X-Ray dataset, which collects radiographic images from children. As we remember, the first COVID cases were all aged people: this was producing misleading results by the community [31]. This, associated with the presence of medical devices, annotations on the side of the radiography etc. are essentially *spurious correlations* with the target task, which can give rise to the insurgence of bias in the prediction. I got personally interested in such a phenomenon, proposing first a work that disentangles bias features from the latent space [7, 8], and then, after observing that debiasing is not equivalent to removing all the information related to the bias (but simply reweighting it [30]), I worked to move the first steps to perform debiasing when the bias labels are unavailable [9]. What lies beyond the horizon for debiasing is to improve the efficiency of these approaches: right now, the most popular paradigm is to train a model twice: first, to capture the bias, then to train without over-relying on it. Finding a way to sense the insurgence of it already at training time, and minimizing its dominance through regularization, is certainly of interest to the community.

In synthesis, my research project lies at the intersection of model efficiency, efficient learning, and feature selection. In the following, I will discuss some natural challenges arising from my research, outlining potential directions and approaches to undertake.

4.1 Efficient Deep Learning On-device

As discussed in Chapter 2, the deployment of larger and larger DNNs is nowadays made possible through a joint effort of technical advancements that enable these models to be trained effectively to counter the gradient vanishing problem, the larger availability of computational resources [203] and some steps moved to compress these models. In particular, according to Sevilla *et al.*, the release of AlphaGo [204] in late 2015 historically marked the advent of a new era, which they call the "Large Scale Era" [205]. While on the one hand such a trend showcased the strength of DNNs, on the other hand it

raises concerns linked to power consumption and the requirement of larger embedded systems [206]: this makes the deployment of these models in extremely constrained environments difficult. Examples of hardware-constrained resources are edge devices such as mobile phones and embedded systems, where high computation costs or memory occupation should be kept at bay. While one possible solution, very popular in the last decade, relied on cloud resources, it comes at some costs: the imperative of faster and stronger connectivity (with associated costs in terms of energy and infrastructure) and concerns linked to the privacy of the shared data with far servers is making the paradigm shifting to perform most (or all) the computation at the edge.

Federating the Model's Learning Efficient deep learning methods can have a significant impact on distributed systems and embedded devices for artificial intelligence. One explored pathway, where data is not directly shared with a central server but the model's training is decentralized, is constituted by *federated learning*. In such a setup, multiple computation nodes collaborate to train a joint model, while at the same time ensuring that data is never explicitly shared, but instead model's parameters [207] or their gradients [195] are shared in update rounds [208]. Due to the decentralized nature of the data, federated learning for real applications faces the pressing challenge of performing local computations on non-independently or identically distributed data, which can harm the model's generalization and convergence. Besides, other technical difficulties include potentially longer convergence time, node's (a)synchronicity, and the extremely high bandwidth requirement challenges for federated learning [209]. These technical difficulties are being currently explored by the community [210, 211, 212, 213], but another very relevant challenge, typical in federated learning setups (despite the premises), is that it is still sensitive to information leakage, showcasing vulnerabilities to reconstruction attacks [214, 215, 216].

Continual Learning In several real-life applications, learning is a continual process that adapts to incoming data streams [217]. Continually updating a DNN on a data stream (where one realistic assumption is the high data correlation and for instance imbalances in the gradient estimation) leads to *catastrophic forgetting* [218]: prior knowledge is overwritten (and for instance, forgot) in favor of the new one injected. Although multiple approaches have historically fought this phenomenon [219, 220], they typically rely on schemes involving full adaptation of the DNN model on the typical simple benchmarks [221, 222, 223]. One recent opportunity is offered by foundation models, which offer big prospects in terms of adaptability to downstream tasks through efficient learning schemes [224, 225, 226, 227, 228]. While some works are already tackling such challenge [229, 230, 231, 232], such a field still lives in its infancy regarding the trade-off between computation demand and generalization guarantees.

Efficient Learning on Resource-constrained Hardware When moving the training (or adaptation) of a DNN to an edge device, where the challenge is to deal with energy consumption (or from a different perspective, latency constraints / computation demand) and memory, a series of technical challenges emerge. These are not frequently accounted for in schemes like federated or continual learning, while on the contrary, they are key to be solved before and then deployed in synergy with solutions for those.

Models trained offline on a large-scale dataset built up at a given time notoriously tend to fall victim to data drift when deployed "in the wild" [233]. Its combination with online learning strategies has the potential to enable continuous model improvement after deployment [234], thus adapting the model predictions to observed evolutions in the data distribution. Moving the computation entirely on the edge, when possible, provides stronger security and privacy guarantees, as the attacker should physically access the device and can not intercept the information flow. The main challenge limiting the feasibility of on-device learning with current learning schemes lies in the computational cost of backpropagation since gradient computation and parameter updating are considerably more expensive than the forward pass. Some approaches circumvent the memory problem by exploring alternatives to backpropagation, including the use of unsupervised learning for image segmentation [235], the Forward-Forward algorithm [236], and PEPITA [237], which in the longer period might constitute a valid option. At the current state, however, these methods underperform compared to backpropagation-based solutions. One option to reduce the backpropagation cost is provided by Lin *et al.*, showing that it is possible to fine-tune a deep neural network with just 256 KB of memory while maintaining good performance [160] thanks to the careful selection of a sub-network to be updated. Orthogonal to this work, Yang *et al.* proposes to reduce the number of unique elements in the gradient map by patch-based compression of the input and gradients of a given layer with respect to the output, thus reducing memory cost and accelerating the learning process [238]. These two works, jointly with my current research [6, 26], suggest that the backpropagation flow is typically sparse: identifying the error signal propagation accounting for memory and computation cost is the next challenge to be tackled, and prospectively can suggest the design, through Neural Architecture Search schemes, of more finetuning-friendly DNNs.

4.2 Multimodal Foundation Models Collapse

In the last few years, we have witnessed a general transition from specialized DNNs trained on (relatively) limited data to more general foundation models [55]. Their practical success lies in their potential to operate in the "open-world", (sometimes arguably)

claiming zero-shot abilities [239]. A typical grouping of these models, proposed in the literature, consists of Large Language Models (LLMs) [240, 241, 242], Vision Transformers Models (ViTs) [132, 133], Latent Diffusion Models [243, 244], and Multimodal models [245, 246], where multimodal data such as text and images are aligned and processed into a unique latent space. The potential of the latter is evident: it is very frequent to collect data in multimodality, and treating jointly the information coming from this adds information that aids in the accomplishment of a target task. One common example comes from the medical domain, where besides the result of exams (that can consist of time series like ECG or images like X-Ray images), medical reports can be either employed as ground truth or as part of the input for illness forecasting. A proper design of latent space alignment mechanisms enables a convenient combination and pipelining of these already large models to accomplish a target goal [247, 248, 249, 250]. A viable scheme employed to train multimodal models consists of training first the models in a single modality in isolation, and then in a second phase performing a joint fine-tuning: this is computationally convenient and at the same time conditions the learning problem avoiding gradient vanishing and instability issues. However, after obtaining a working large multimodal model, one question arises: is all the complexity put in place really necessary?

As analyzed in Sec. 2.6, unstructured pruning, if properly conditioned, can induce a phenomenon of layer collapse, where an entire layer is linearizable and merged with the next one. This effect is also motivated by the known typical high redundancy in DNNs [251, 252]. Such a process requires to be gradual and loss-informed: large perturbations in multimodal foundation models are not tolerable, as recovering performance can be an extremely hard and computationally expensive task. This suggests that traditional pruning schemes like [3, 92, 253] are not the best choice for solving such a task. One intriguing possibility is offered by Optimal Transport, a mathematical framework with deep historical roots [254, 255] useful for probability distribution discrepancy quantification. Through the design of a proper regularization function, it could be possible to weakly induce distribution matching between input and output distribution of (sequences of) layers in the large model. It is not unrealistic that unimodal models, when deployed to solve a different (downstream) task, might not require the same complexity, and on the contrary, might suggest the uprisal of *shortcuts* that can precociously merge multimodal information. Such a research line has the potential to make these models scale in terms of computation and memory, and potentially enhance the error signal at fine-tuning time, further enhancing their generalization capability.

4.3 Understanding the Bias Encoding in Deep Models

As we have seen in Chapter 1 and 3, debiasing approaches are being more and more explored in the last luster due to upcoming regulations through the AI Act, and despite multiple efforts conducted by the community, all the typical approaches require heavy training procedures for either properly tuning the hyper-parameters of the models, or training multiple times the neural networks. This comes at another very relevant environmental cost, and besides it is not granted that such solutions can always apply to any architecture/task. This is particularly motivated by the inherent lack of knowledge behind the propagation of the information linked to the bias through the DNN model. A deeper understanding of it would seemingly lead to cheaper and more immediate and cheap training.

One trend of research around debiasing approaches focuses on LLM and foundation models, given the big threat the generative model can effectively represent for the world community. Notably, one trending subtopic in the community is *unlearning* [256, 257, 258]: large models trained on prohibited data (typically for copyright issues) should forget them, hopefully not resorting to the same expensive training scheme. This has links with some presented works on privacy and/or providing *guardrails*. As already evidenced in Sec. 3.5, there is a fundamental difference between hiding information and debiasing: the latter still allows the use of such information but in a more balanced way. Some works are already showing that debiasing large models might be easier than expected: for example, properly disabling some heads in the Multihead Self-Attention block in Transformers can remove biases [259]. However, this research still lives in its infancy, and current state-of-the-art methods are certainly inapplicable to large-scale models due to their computational costs. In a recent work, however, I have shown the existence, in DNN models affected by bias, of sub-networks that are unbiased, in a supervised debiasing setup [28]. Similar to what Frankle and Carbin did for their lottery ticket hypothesis [74], however, this existence has been only empirically shown: efficient methods to retrieve it without resorting to supervised methods are still a matter of research by the community.

Identifying (sub-)architectures that encode (or repel) certain features in the longer term would open the doors to joint work between feature selection and complexity reduction. By designing minimal architectures that satisfy known criteria in terms of information propagation, it would be possible to build a system of hierarchical rules to compose ad-hoc models compliant with specific regulations. Such a longer-term objective rightfully frames itself within the context of *interpretable AI*, given that, through a bottom-up approach, it would be possible to understand how the deployed DNN works.

My works

[1] Enzo Tartaglione, Andrea Bragagnolo, Francesco Odierna, Attilio Fiandrotti, and Marco Grangetto. "SeReNe: Sensitivity-Based Regularization of Neurons for Structured Sparsity in Neural Networks". In: **IEEE Transactions on Neural Networks and Learning Systems** (2021).

[2] Enzo Tartaglione, Skjalg Lepsøy, Attilio Fiandrotti, and Gianluca Francini. "Learning sparse neural networks via sensitivity-driven regularization". In: **Advances in Neural Information Processing Systems**. 2018.

[3] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto. "Loss-based sensitivity regularization: towards deep sparse neural networks". In: **Neural Networks** (2022).

[4] Victor Quétu and Enzo Tartaglione. "DSD^2: Can We Dodge Sparse Double Descent and Compress the Neural Network Worry-Free?" In: **AAAI Conference on Artificial Intelligence**. 2024.

[5] Enzo Tartaglione, Stéphane Lathuilière, Attilio Fiandrotti, Marco Cagnazzo, and Marco Grangetto. "HEMP: high-order entropy minimization for neural network compression". In: **Neurocomputing** 461 (2021).

[6] Andrea Bragagnolo, Enzo Tartaglione, and Marco Grangetto. "To update or not to update? Neurons at equilibrium in deep models". In: **Advances in Neural Information Processing Systems** 35 (2022).

[7] Enzo Tartaglione, Carlo Alberto Barbano, and Marco Grangetto. "End: Entangling and disentangling deep representations for bias correction". In: **IEEE/CVF conference on Computer Vision and Pattern Recognition**. 2021.

[8] Carlo Alberto Barbano, Benoit Dufumier, Enzo Tartaglione, Marco Grangetto, and Pietro Gori. "Unbiased Supervised Contrastive Learning". In: **International Conference on Learning Representations**. 2023.

[9] Rémi Nahon, Van-Tam Nguyen, and Enzo Tartaglione. "Mining bias-target Alignment from Voronoi Cells". In: **IEEE/CVF International Conference on Computer Vision**. 2023.

[10] Enzo Tartaglione, Francesca Gennari, Victor Quétu, and Marco Grangetto. "Disentangling private classes through regularization". In: **Neurocomputing** 554 (2023).

[11] Enzo Tartaglione, Skjalg Lepsøy, Attilio Fiandrotti, and Gianluca Francini. "Learning sparse neural networks via sensitivity-driven regularization". In: **Advances in Neural Information Processing Systems** 31 (2018).

[12] Carl De Sousa Trias, Mihai Petru Mitrea, Attilio Fiandrotti, Marco Cagnazzo, Sumanta Chaudhuri, and Enzo Tartaglione. "Find the Lady: Permutation and Re-synchronization of Deep Neural Networks". In: **AAAI Conference on Artificial Intelligence**. Vol. 38. 19. 2024.

[13] Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean marc Martinez, Andrei Bursuc, and Gianni Franchi. "Packed Ensembles for efficient uncertainty estimation". In: **International Conference on Learning Representations**. 2023.

[14] Imad Eddine Marouf, Enzo Tartaglione, and Stéphane Lathuilière. "Mini but Mighty: Finetuning ViTs With Mini Adapters". In: *IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024.

[15] Victor Quétu, Zhu Liao, and Enzo Tartaglione. "The Simpler The Better: An Entropy-Based Importance Metric To Reduce Neural Networks' Depth". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2024.

[16] Andrea Bragagnolo, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto. "On the role of structured pruning for neural network compression". In: *IEEE International Conference on Image Processing*. IEEE. 2021.

[17] Chenxi Lola Deng and Enzo Tartaglione. "Compressing explicit voxel grid representations: fast nerfs become also small". In: *IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023.

[18] Francesco Di Sario, Riccardo Renzulli, Enzo Tartaglione, and Marco Grangetto. "Two is Better than One: Achieving High-Quality 3D Scene Modeling with a NeRF Ensemble". In: *International Conference on Image Analysis and Processing*. Springer. 2023.

[19] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. "Can Unstructured Pruning Reduce the Depth in Deep Neural Networks?" In: *IEEE/CVF International Conference on Computer Vision Workshops*. 2023.

[20] Enzo Tartaglione, Andrea Bragagnolo, and Marco Grangetto. "Pruning Artificial Neural Networks: A Way to Find Well-Generalizing, High-Entropy Sharp Minima". In: *International Conference on Artificial Neural Networks*. Springer International Publishing, 2020.

[21] Victor Quétu, Marta Milovanović, and Enzo Tartaglione. "Sparse Double Descent in Vision Transformers: real or phantom threat?" In: *International Conference on Image Analysis and Processing*. Springer. 2023.

[22] Victor Quétu and Enzo Tartaglione. "Dodging the Double Descent in Deep Neural Networks". In: *IEEE International Conference on Image Processing*. 2023.

[23] Enzo Tartaglione. "The Rise of the Lottery Heroes: Why Zero-Shot Pruning is Hard". In: *IEEE International Conference on Image Processing*. 2022.

[24] Ziyu Li, Enzo Tartaglione, and Van-Tam Nguyen. "SCoTTi: Save Computation at Training Time with an adaptive framework". In: *IEEE/CVF International Conference on Computer Vision Workshops*. 2023.

[25] Gabriele Spadaro, Riccardo Renzulli, Andrea Bragagnolo, Jhony H Giraldo, Attilio Fiandrotti, Marco Grangetto, and Enzo Tartaglione. "Shannon Strikes Again! Entropy-Based Pruning in Deep Neural Networks for Transfer Learning Under Extreme Memory and Computation Budgets". In: *IEEE/CVF International Conference on Computer Vision Workshops*. 2023.

[26] Aël Quélennec, Enzo Tartaglione, Pavlo Mozharovskyi, and Van-Tam Nguyen. "Towards On-device Learning on the Edge: Ways to Select Neurons to Update under a Budget Constraint". In: *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops*. 2024.

[27] Enzo Tartaglione and Marco Grangetto. "A non-discriminatory approach to ethical deep learning". In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE. 2020.

[28] Rémi Nahon, Ivan Luiz De Moura Matos, Van-Tam Nguyen, and Enzo Tartaglione. *Debiasing surgeon: fantastic weights and how to find them*. 2024.

[29] Carlo Alberto Barbano, Enzo Tartaglione, and Marco Grangetto. "Bridging the gap between debiasing and privacy for deep learning". In: *IEEE/CVF International Conference on Computer Vision Workshops*. 2021.

[30] Enzo Tartaglione. "Information Removal at the bottleneck in Deep Neural Networks". In: *British Machine Vision Conference*. 2022.

[31] Enzo Tartaglione, Carlo Alberto Barbano, Claudio Berzovini, Marco Calandri, and Marco Grangetto. "Unveiling COVID-19 from Chest X-ray with deep learning: a hurdles race with small data". In: *Int. J. Environ. Res. Public Health* 17.18 (2020).

[32] Enzo Tartaglione, Marco Grangetto, Davide Cavagnino, and Marco Botta. "Delving in the loss landscape to embed robust watermarks into neural networks". In: *International Conference on Pattern Recognition*. IEEE. 2021.

[33] Carl De Sousa Trias, Mihai Mitrea, Enzo Tartaglione, Attilio Fiandrotti, Marco Cagnazzo, and Sumanta Chaudhuri. "A hitchhiker's guide to white-box neural network watermarking robustness". In: *European Workshop on Visual Information Processing*. IEEE. 2023.

[34] Enzo Tartaglione, Beatrice Biancardi, Maurizio Mancini, and Giovanna Varni. "A hitchhiker's guide towards transactive memory system modeling in small group interactions". In: *Companion Publication of the 2021 International Conference on Multimodal Interaction*. 2021.

[35] Carlo Alberto Barbano, Daniele Perlo, Enzo Tartaglione, Attilio Fiandrotti, Luca Bertero, Paola Cassoni, and Marco Grangetto. "Unitopatho, a labeled histopathological dataset for colorectal polyps classification and adenoma dysplasia grading". In: *IEEE International Conference on Image Processing*. IEEE. 2021.

[36] Carlo Alberto Barbano, Enzo Tartaglione, Claudio Berzovini, Marco Calandri, and Marco Grangetto. "A two-step radiologist-like approach for covid-19 computer-aided diagnosis from chest x-ray images". In: *International Conference on Image Analysis and Processing*. Springer. 2022.

[37] Umberto A Gava, Federico D'agata, Enzo Tartaglione, Riccardo Renzulli, Marco Grangetto, Francesca Bertolino, Ambra Santonocito, Edwin Bennink, Giacomo Vaudano, Andrea Boghi, et al. "Neural Network-derived perfusion maps: a Model-free approach to computed tomography perfusion in patients with acute ischemic stroke". In: *Frontiers in Neuroinformatics* 17 (2023).

[38] Yinghao Wang, Rémi Nahon, Enzo Tartaglione, Pavlo Mozharovskyi, and Van-Tam Nguyen. "Optimized preprocessing and tiny ml for attention state classification". In: *IEEE Statistical Signal Processing Workshop*. IEEE. 2023.

[39] Van-Tam Nguyen, Enzo Tartaglione, and Tuan Dinh. "AIoT-based Neural Decoding and Neurofeedback for Accelerated Cognitive Training: Vision, Directions and Preliminary Results". In: *2023 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE. 2023.

[40] Marta Milovanović, Enzo Tartaglione, Marco Cagnazzo, and Félix Henry. "Learn How to Prune Pixels for Multi-View Neural Image-Based Synthesis". In: *IEEE International Conference on Multimedia and Expo Workshops*. IEEE. 2023.

[41] Melan Vijayaratnam, Marco Cagnazzo, Giuseppe Valenzise, and Enzo Tartaglione. "All Predictions Matter: an Online Video Prediction Approach". In: *2023 11th European Workshop on Visual Information Processing (EUVIP)*. IEEE. 2023.

[42] Melan Vijayaratnam, Marta Milovanović, Marco Cagnazzo, Enzo Tartaglione, and Giuseppe Valenzise. "Unified Measures for the Rate-Distortion-Latency Trade-Off". In: *2023 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE. 2023.

[43] Alberto Presta, Attilio Fiandrotti, Enzo Tartaglione, and Marco Grangetto. "A Differentiable Entropy Model for Learned Image Compression". In: *International Conference on Image Analysis and Processing*. Springer. 2023.

[44] Alberto Presta, Gabriele Spadaro, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto. "Domain Adaptation for Learned Image Compression with Supervised Adapters". In: *2024 Data Compression Conference (DCC)*. IEEE. 2024.

[45] Riccardo Renzulli, Enzo Tartaglione, Attilio Fiandrotti, and Marco Grangetto. "Capsule networks with routing annealing". In: *International Conference on Artificial Neural Networks*. Springer. 2021.

[46] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. *NEPENTHE: Entropy-Based Pruning as a Neural Network Depth's Reducer*. 2024.

Bibliography

[47] Kunihiko Fukushima. “Cognitron: A self-organizing multilayered neural network”. In: *Biological cybernetics* (1975).

[48] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *International Conference on Machine Learning*. 2010.

[49] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011.

[50] Xavier Glorot and Yoshua 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.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2016.

[53] DP Kingma. “Adam: a method for stochastic optimization”. In: *International Conference on Learning Representations*. 2014.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in Neural Information Processing Systems* (2017).

[55] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. “On the opportunities and risks of foundation models”. In: *arXiv preprint arXiv:2108.07258* (2021).

[56] Andrey Nikolayevich Tikhonov et al. “On the stability of inverse problems”. In: *Dokl. akad. nauk sssr*. 1943.

[57] Andrei Nikolaevich Tikhonov. "On the solution of ill-posed problems and the method of regularization". In: *Doklady akademii nauk*. Russian Academy of Sciences. 1963.

[58] Valentin Konstantinovich Ivanov. "On linear problems which are not well-posed". In: *Doklady akademii nauk*. Russian Academy of Sciences. 1962.

[59] Anatolii Borisovich Bakushinskii. "A general method of constructing regularizing algorithms for a linear incorrect equation in Hilbert space". In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* (1967).

[60] J Hadamard. "Princeton University Bulletin". In: *1902* (1902).

[61] George Backus and Freeman Gilbert. "The resolving power of gross earth data". In: *Geophysical Journal International* (1968).

[62] RS Anderssen. "The linear functional strategy for improperly posed problems". In: *Inverse Problems: Proceedings of the Conference held at the Mathematical Research Institute at Oberwolfach, Black Forest, May 18–24, 1986*. Springer. 1986.

[63] AK Louis. "Approximate inverse for linear and some nonlinear problems". In: *Inverse problems* (1996).

[64] Douglas M Bates and Grace Wahba. *A truncated singular value decomposition and other methods for generalized cross-validation*. University of Wisconsin, Department of Statistics, 1983.

[65] PC Hansen. "The truncatedSVD as a method for regularization, BIT, 27, 534–553". In: *CrossRef MathSciNet MATH* (1987).

[66] Heinz W Engl, Karl Kunisch, and Andreas Neubauer. "Convergence rates for Tikhonov regularisation of non-linear ill-posed problems". In: *Inverse problems* (1989).

[67] Martin Hanke, Andreas Neubauer, and Otmar Scherzer. "A convergence analysis of the Landweber iteration for nonlinear ill-posed problems". In: *Numerische Mathematik* (1995).

[68] Barbara Kaltenbacher. "Some Newton-type methods for the regularization of nonlinear ill-posed problems". In: *Inverse Problems* (1997).

[69] Barbara Kaltenbacher, Frank Schöpfer, and Thomas Schuster. "Iterative methods for nonlinear ill-posed problems in Banach spaces: convergence and applications to parameter identification problems". In: *Inverse Problems* (2009).

[70] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*. Springer Science & Business Media, 1996.

[71] Martin Benning and Martin Burger. "Modern regularization methods for inverse problems". In: *Acta numerica* (2018).

[72] Christos Louizos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through L_0 Regularization". In: *International Conference on Learning Representations*. 2018.

[73] D. Molchanov, A. Ashukha, and D. Vetrov. "Variational dropout sparsifies deep neural networks". In: *International Conference on Machine Learning*. 2017.

[74] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *International Conference on Learning Representations* (2019).

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. 2012.

[76] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations* (2015).

[77] The Motion Picture Expert Group. "Compression of neural networks for multimedia content description and analysis". In: (MPEG 125 - Marrakesh).

[78] Yao Lu, G. Lu, R. Lin, Jinxing Li, and D. Zhang. "SRGC-Nets: Sparse Repeated Group Convolutional Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[79] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[80] Hanxiao Liu, Karen Simonyan, and Yiming Yang. "DARTS: Differentiable Architecture Search". In: *International Conference on Learning Representations*. 2019.

[81] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey". In: *Journal of Machine Learning Research* (2019).

[82] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations". In: *Advances in Neural Information Processing Systems* (2015).

[83] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.

[84] Simon Wiedemann, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinč, David Neumann, Tung Nguyen, Heiko Schwarz, Thomas Wiegand, et al. "Deepcabac: A universal compression algorithm for deep neural networks". In: *IEEE Journal of Selected Topics in Signal Processing* (2020).

[85] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. "Once-for-all: Train one network and specialize it for efficient deployment". In: *arXiv preprint arXiv:1908.09791* (2019).

[86] Hrushikesh N Mhaskar and Tomaso Poggio. "Deep vs. shallow networks: An approximation theory perspective". In: *Analysis and Applications* (2016).

[87] A. Brutzkus, A. Globerson, E. Malach, and S. Shalev-Shwartz. "SGD learns over-parameterized networks that provably generalize on linearly separable data". In: 2018.

[88] Michael C Mozer and Paul Smolensky. "Skeletonization: A technique for trimming the fat from a network via relevance assessment". In: *Advances in Neural Information Processing Systems*. 1989.

[89] Yann LeCun, John S Denker, and Sara A Solla. "Optimal brain damage". In: *Advances in Neural Information Processing Systems*. 1990.

[90] Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *International Conference on Learning Representations* (2016).

[91] Zheng He, Zeke Xie, Quanzhi Zhu, and Zengchang Qin. "Sparse Double Descent: Where Network Pruning Aggravates Overfitting". In: *International Conference on Machine Learning*. PMLR. 2022.

[92] Song Han, Jeff Pool, John Tran, and William Dally. "Learning both weights and connections for efficient neural network". In: *Advances in Neural Information Processing Systems*. 2015.

[93] Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. "Pruning from Scratch." In: *AAAI Conference on Artificial Intelligence*. 2020.

[94] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. "Provable Filter Pruning for Efficient Neural Networks". In: *International Conference on Learning Representations*. 2020.

[95] N. Lee, Thalaiyasingam Ajanthan, and P. Torr. "SNIP: Single-shot Network Pruning based on Connection Sensitivity". In: *International Conference on Learning Representations* (2019).

[96] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in Neural Information Processing Systems* (2018).

[97] Yite Wang, Dawei Li, and Ruoyu Sun. "NTK-SAP: Improving neural network pruning by aligning training dynamics". In: *arXiv preprint arXiv:2304.02840* (2023).

[98] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* (2014).

[99] Durk P Kingma, Tim Salimans, and Max Welling. "Variational dropout and the local reparameterization trick". In: *Advances in Neural Information Processing Systems*. 2015.

[100] Trevor Gale, Erich Elsen, and Sara Hooker. *The State of Sparsity in Deep Neural Networks*. 2019.

[101] Aidan N. Gomez, Ivan Zhang, Kevin Swersky, Yarin Gal, and Geoffrey E. Hinton. "Learning Sparse Networks Using Targeted Dropout". In: *CoRR* (2019).

[102] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. "Learning structured sparsity in deep neural networks". In: *Advances in Neural Information Processing Systems* (2016).

[103] Charles W. Groetsch. *Inverse Problems in the Mathematical Sciences*. Vieweg, 1993.

[104] Yiwen Guo, Anbang Yao, and Yurong Chen. "Dynamic network surgery for efficient dnns". In: *Advances in Neural Information Processing Systems*. 2016.

[105] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. "Soft filter pruning for accelerating deep convolutional neural networks". In: *arXiv preprint arXiv:1808.06866* (2018).

[106] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. "Filter pruning via geometric median for deep convolutional neural networks acceleration". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[107] Junzhou Huang, Tong Zhang, and Dimitris Metaxas. "Learning with structured sparsity". In: *International Conference on Machine Learning*. 2009.

[108] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. "Chasing sparsity in vision transformers: An end-to-end exploration". In: *Advances in Neural Information Processing Systems* (2021).

[109] K. Ullrich, M. Welling, and E. Meeds. "Soft weight-sharing for neural network compression". In: *International Conference on Learning Representations*. 2019.

[110] Andrea Bragagnolo and Carlo Alberto Barbano. "Simplify: A python library for optimizing pruned neural networks". In: *SoftwareX* (2022).

[111] Shivani Gupta and Atul Gupta. "Dealing with noise problem in machine learning data-sets: A systematic review". In: *Procedia Computer Science* (2019).

[112] Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia Li. "Learning from noisy labels with distillation". In: *IEEE/CVF International Conference on Computer Vision*. 2017.

[113] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. "Noise as a resource for learning in knowledge distillation". In: *IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021.

[114] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. "Early-learning regularization prevents memorization of noisy labels". In: *Advances in Neural Information Processing Systems* (2020).

[115] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* (2019).

[116] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. "Deep Double Descent: Where Bigger Models and More Data Hurt". In: *International Conference on Learning Representations*. 2020.

[117] Kelvin Kan, James G Nagy, and Lars Ruthotto. "Avoiding The Double Descent Phenomenon of Random Feature Models Using Hybrid Regularization". In: *arXiv preprint arXiv:2012.06667* (2020).

[118] Leslie Rice, Eric Wong, and J. Zico Kolter. "Overfitting in adversarially robust deep learning". In: *International Conference on Machine Learning*. 2020.

[119] Tianlong Chen, Zhenyu (Allen) Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. "Robust Overfitting may be mitigated by properly learned smoothening". In: *International Conference on Learning Representations*. 2021.

[120] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. "Averaging weights leads to wider optima and better generalization". In: *UAI* (2018).

[121] N Tishby, FC Pereira, W Bialek, B Hajek, and RS Sreenivas. *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*. 1999.

[122] Naftali Tishby and Noga Zaslavsky. "Deep learning and the information bottleneck principle". In: *IEEE Information Theory Workshop*. 2015.

[123] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. "On the Information Bottleneck Theory of Deep Learning". In: *International Conference on Learning Representations*. 2018.

[124] Ziqi Pan, Li Niu, Jianfu Zhang, and Liqing Zhang. "Disentangled information bottleneck". In: *AAAI Conference on Artificial Intelligence*. 2021.

[125] Vudtiwat Ngampruetikorn and David J. Schwab. "Information bottleneck theory of high-dimensional regression: relevancy, efficiency and optimality". In: *Advances in Neural Information Processing Systems*. 2022.

[126] Preetum Nakkiran, Prayaag Venkat, Sham M. Kakade, and Tengyu Ma. "Optimal Regularization can Mitigate Double Descent". In: *International Conference on Learning Representations*. 2021.

[127] Yoon Kim and Alexander M Rush. "Sequence-level knowledge distillation". In: *EMNLP* (2016).

[128] Baiyun Cui, Yingming Li, and Zhongfei Zhang. "Joint structured pruning and dense knowledge distillation for efficient transformer model compression". In: *Neurocomputing* (2021).

[129] Zeyuan Wei, Li Hao, and Xueliang Zhang. "Model Compression by Iterative Pruning with Knowledge Distillation and Its Application to Speech Enhancement". In: *Interspeech*. 2022.

[130] Luca Saglietti and Lenka Zdeborová. "Solvable model for inheriting the regularization through knowledge distillation". In: *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*. 2022.

[131] Jinhyuk Park and Albert No. "Prune your model before distill it". In: *European Conference on Computer Vision*. 2022.

[132] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2020.

[133] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *IEEE/CVF International Conference on Computer Vision*. 2021.

[134] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. "Towards a Unified View of Parameter-Efficient Transfer Learning". In: *International Conference on Learning Representations*. 2021.

[135] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2021.

[136] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. *Visual Prompt Tuning*. 2022.

[137] Rodrigo Berriel, Stéphane Lathuillère, Moin Nabi, Tassilo Klein, Thiago Oliveira-Santos, Nicu Sebe, and Elisa Ricci. "Budget-aware adapters for multi-domain learning". In: *IEEE/CVF International Conference on Computer Vision*. 2019.

[138] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. "Piggyback: Adapting a single network to multiple tasks by learning to mask weights". In: *European Conference on Computer Vision*. 2018.

[139] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. "Learning multiple visual domains with residual adapters". In: *Advances in Neural Information Processing Systems* (2017).

[140] Sylvestre-Alvise Rebuffi, Andrea Vedaldi, and Hakan Bilen. "Efficient Parametrization of Multi-domain Deep Neural Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.

[141] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. "Cmt: Convolutional neural networks meet vision transformers". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[142] Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco De Nadai. "Efficient Training of Visual Transformers with Small Datasets". In: *Advances in Neural Information Processing Systems*. 2021.

[143] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. "Parameter-efficient transfer learning for NLP". In: *International Conference on Machine Learning*. PMLR. 2019.

[144] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. "Compacter: Efficient low-rank hypercomplex adapter layers". In: *Advances in Neural Information Processing Systems* (2021).

[145] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. "Parameter-efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks". In: *ACL/IJCNLP*. 2021.

[146] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. "AdapterFusion: Non-Destructive Task Composition for Transfer Learning". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main commentthree*. Online: Association for Computational Linguistics, Apr. 2021.

[147] Yves Chauvin. "A back-propagation algorithm with optimal use of hidden units". In: *Advances in Neural Information Processing Systems* (1988).

[148] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: *International Conference on Learning Representations*. 2017.

[149] Alex Renda, Jonathan Frankle, and Michael Carbin. "Comparing rewinding and fine-tuning in neural network pruning". In: *arXiv preprint arXiv:2003.02389* (2020).

[150] Christian H.X. Ali Mehmeti-Göpel and Jan Disselhoff. "Nonlinear Advantage: Trained Networks Might Not Be As Complex as You Think". In: *International Conference on Machine Learning*. 2023.

[151] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[152] Cecil C Craig. "On the frequency function of xy". In: *The Annals of Mathematical Statistics* (1936).

[153] Antonio Seijas-Macías and Amílcar Oliveira. "An approach to distribution of the product of two normal variables". In: *Discussiones Mathematicae Probability and Statistics* (2012).

[154] Amir Ben Dror, Niv Zehngut, Avraham Raviv, Evgeny Artyomov, Ran Vitek, and Roy Jevnisek. "Layer folding: Neural network depth reduction using activation linearization". In: *arXiv preprint arXiv:2106.09309* (2021).

[155] Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. "Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network?" In: *Advances in Neural Information Processing Systems* (2022).

[156] Like Hui, Mikhail Belkin, and Preetum Nakkiran. "Limitations of neural collapse for understanding generalization in deep learning". In: *arXiv preprint arXiv:2202.08384* (2022).

[157] Yunqiang Li, Jan C van Gemert, Torsten Hoefler, Bert Moons, Evangelos Eleftheriou, and Bram-Ernst Verhoef. "Differentiable transportation pruning". In: *IEEE/CVF International Conference on Computer Vision*. 2023.

[158] Alexander Theus, Olin Geimer, Friedrich Wicke, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. "Towards Meta-Pruning via Optimal Transport". In: *arXiv preprint arXiv:2402.07839* (2024).

[159] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic differentiation in machine learning: a survey". In: *Journal of Marchine Learning Research* (2018).

[160] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. "On-device training under 256kb memory". In: *Advances in Neural Information Processing Systems* (2022).

[161] Prechelt Lutz. "Automatic early stopping using cross validation: quantifying the criteria". In: *Neural Networks* (1998).

[162] Moreno D'Incà, Elia Peruzzo, Massimiliano Mancini, Dejia Xu, Vidit Goel, Xingqian Xu, Zhangyang Wang, Humphrey Shi, and Nicu Sebe. "OpenBias: Open-set Bias Detection in Text-to-Image Generative Models". In: *arXiv preprint arXiv:2404.07990* (2024).

[163] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. "Learning De-biased Representations with Biased Representations". In: *International Conference on Machine Learning*. 2020.

[164] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. "Learning Not to Learn: Training Deep Neural Networks With Biased Data". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

[165] William Thong and Cees G. M. Snoek. "Feature and Label Embedding Spaces Matter in Addressing Image Classifier Bias". In: *British Machine Vision Conference*. 2021.

[166] European Commission (AI HLEG). *Ethics guidelines for trustworthy AI*. High-Level Expert Group on Artificial Intelligence, 2019.

[167] Baobao Zhang and Allan Dafoe. "Artificial intelligence: American attitudes and trends". In: *Available at SSRN 3312874* (2019).

[168] Joshua Attenberg, Panos Ipeirotis, and Foster Provost. "Beat the machine: Challenging humans to find a predictive model's "unknown unknowns"". In: *Journal of Data and Information Quality (JDIQ)* (2015).

[169] Ioannis D Apostolopoulos and Tzani A Mpesiana. "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks". In: *Physical and Engineering Sciences in Medicine* (2020).

[170] Prabira Kumar Sethy and Santi Kumari Behera. "Detection of coronavirus disease (covid-19) based on deep features". In: *Preprints* (2020).

[171] Mohsan Alvi, Andrew Zisserman, and Christoffer Nellåker. "Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings". In: *European Conference on Computer Vision Workshops*. 2018.

[172] Remi Cadene, Corentin Dancette, Matthieu Cord, Devi Parikh, et al. "Rubi: Reducing unimodal biases for visual question answering". In: *Advances in Neural Information Processing Systems*. 2019.

[173] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." In: *International Conference on Learning Representations*. 2019.

[174] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. "Learning from Failure: Training Debiased Classifier from Biased Classifier". In: *Advances in Neural Information Processing Systems*. 2020.

[175] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. "Shortcut learning in deep neural networks". In: *Nature Machine Intelligence* (2020).

[176] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. "Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Association for Computational Linguistics, 2019, pp. 4067–4080.

[177] Haohan Wang, Zexue He, Zachary L. Lipton, and Eric P. Xing. "Learning Robust Representations by Projecting Superficial Statistics Out". In: *International Conference on Learning Representations*. 2019.

[178] Jungsoo Lee, Eungyeup Kim, Juyoung Lee, Jihyeon Lee, and Jaegul Choo. "Learning debiased representation via disentangled feature augmentation". In: *Advances in Neural Information Processing Systems* (2021).

[179] Zhilu Zhang and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels". In: *Advances in Neural Information Processing Systems* (2018).

[180] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. "StarGAN v2: Diverse Image Synthesis for Multiple Domains". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[181] Lei Kang, Pau Riba, Marcal Rusinol, Alicia Fornes, and Mauricio Villegas. "Content and style aware generation of text-line images for handwriting recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[182] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. "Decision theory for discrimination-aware classification". In: *2012 IEEE 12th international conference on data mining*. IEEE. 2012.

[183] Moritz Hardt, Eric Price, and Nati Srebro. "Equality of opportunity in supervised learning". In: *Advances in Neural Information Processing Systems* (2016).

[184] Faisal Kamiran and Toon Calders. "Data preprocessing techniques for classification without discrimination". In: *Knowledge and information systems* (2012).

[185] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. "Discrimination-aware data mining". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008.

[186] Zhiheng Li, Anthony Hoogs, and Chenliang Xu. "Discover and mitigate unknown biases with debiasing alternate networks". In: *European Conference on Computer Vision*. Springer. 2022.

[187] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. "Environment inference for invariant learning". In: *International Conference on Machine Learning*. PMLR. 2021.

[188] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. "Invariant risk minimization". In: *arXiv preprint arXiv:1907.02893* (2019).

[189] Faruk Ahmed, Yoshua Bengio, Harm van Seijen, and Aaron C. Courville. "Systematic generalisation with group invariant predictions". In: *International Conference on Learning Representations*. 2021.

[190] Nayeong Kim, Sehyun Hwang, Sungsoo Ahn, Jaesik Park, and Suha Kwak. "Learning debiased classifier with biased committee". In: *Advances in Neural Information Processing Systems* (2022).

[191] Sumyeong Ahn, Seongyoon Kim, and Se-Young Yun. "Mitigating Dataset Bias by Using Per-Sample Gradient". In: *International Conference on Learning Representations*. 2023.

[192] Cynthia Dwork and Jing Lei. "Differential privacy and robust statistics". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009.

[193] John C Duchi, Michael I Jordan, and Martin J Wainwright. "Privacy aware learning". In: *Journal of the ACM (JACM)* (2014).

[194] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016.

[195] Reza Shokri and Vitaly Shmatikov. "Privacy-preserving deep learning". In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015.

[196] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. "Fairgan: Fairness-aware generative adversarial networks". In: *2018 IEEE International Conference on Big Data*. IEEE. 2018.

[197] Prasanna Sattigeri, Samuel C Hoffman, Vijil Chenthamarakshan, and Kush R Varshney. "Fairness gan". In: *arXiv preprint arXiv:1805.09910* (2018).

[198] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. "Learning adversarially fair and transferable representations". In: *arXiv preprint arXiv:1802.06309* (2018).

[199] Zeyu Wang, Klint Qinami, Ioannis Christos Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. "Towards fairness in visual recognition: Effective strategies for bias mitigation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[200] Youngkyu Hong and Eunho Yang. "Unbiased classification through bias-contrastive and bias-balanced learning". In: *Advances in Neural Information Processing Systems* (2021).

[201] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang. "Distributionally Robust Neural Networks". In: *International Conference on Learning Representations*. 2020.

[202] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. "The difficulty of training sparse neural networks". In: *arXiv preprint arXiv:1906.10732* (2019).

[203] Toru Baji. "Evolution of the GPU Device widely used in AI and Massive Parallel Processing". In: *2018 IEEE 2nd Electron devices technology and manufacturing conference (EDTM)*. IEEE. 2018.

[204] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. "Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond". In: *IEEE/CAA Journal of Automatica Sinica* (2016).

[205] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. "Compute trends across three eras of machine learning". In: *2022 International Joint Conference on Neural Networks*. IEEE. 2022.

[206] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[207] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017.

[208] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. "Advances and open problems in federated learning". In: *Foundations and trends® in machine learning* (2021).

[209] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. "On the Convergence of FedAvg on Non-IID Data". In: *International Conference on Learning Representations*. 2020.

[210] Hatem Osama Ismail, Mohamed Waleed Fakhr, and A andMohamed. "A federated pure vision transformer algorithm for computer vision using dynamic aggregation model". In: *NeuroQuantology* (2022).

[211] Aritra Mitra, Rayana Jaafar, George J Pappas, and Hamed Hassani. "Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients". In: *Advances in Neural Information Processing Systems* (2021).

[212] Debora Caldarola, Barbara Caputo, and Marco Ciccone. "Improving generalization in federated learning by seeking flat minima". In: *European Conference on Computer Vision*. Springer. 2022.

[213] Tom Overman, Garrett Blum, and Diego Klabjan. "A primal-dual algorithm for hybrid federated learning". In: *AAAI Conference on Artificial Intelligence*. 2024.

[214] Anastasia Pustozerova and Rudolf Mayer. "Information leaks in federated learning". In: *Proceedings of the network and distributed system security symposium*. 2020.

[215] Dimitar Iliev Dimitrov, Mislav Balunovic, Nikola Konstantinov, and Martin Vechev. "Data Leakage in Federated Averaging". In: *Transactions on Machine Learning Research* (2022).

[216] Mark Vero, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin Vechev. "TabLeak: Tabular data leakage in federated learning". In: *International Conference on Machine Learning*. PMLR. 2023.

[217] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. "A survey on ensemble learning for data stream classification". In: *CSUR* (2017).

[218] Robert M French. "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* (1999).

[219] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. "Class-incremental learning: survey and performance evaluation on image classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[220] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. *A Comprehensive Survey of Continual Learning: Theory, Method and Application*. 2023.

[221] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* (2017).

[222] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. "Rotate your networks: Better weight consolidation and less catastrophic forgetting". In: *International Conference on Pattern Recognition*. 2018.

[223] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence". In: *International Conference on Machine Learning*. 2017.

[224] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. "Dualprompt: Complementary prompting for rehearsal-free continual learning". In: *European Conference on Computer Vision*. 2022.

[225] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. "Learning to prompt for continual learning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[226] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. "S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning". In: *Advances in Neural Information Processing Systems* (2022).

[227] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. "Pivot: Prompting for video continual learning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

[228] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. "CODA-Prompt: COntinual Decomposed Attention-Based Prompting for Rehearsal-Free Continual Learning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

[229] Paul Janson, Wenxuan Zhang, Rahaf Aljundi, and Mohamed Elhoseiny. "A simple baseline that questions the use of pretrained-models in continual learning". In: *NeurIPS Workshop on Distribution Shifts*. 2022.

[230] Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. *Revisiting Class-Incremental Learning with Pre-Trained Models: Generalizability and Adaptivity are All You Need*. 2023.

[231] Aristeidis Panos, Yuriko Kobe, Daniel Olmeda Reino, Rahaf Aljundi, and Richard E Turner. "First Session Adaptation: A Strong Replay-Free Baseline for Class-Incremental Learning". In: *IEEE/CVF International Conference on Computer Vision*. 2023.

[232] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton van den Hengel. "Ranpac: Random projections and pre-trained models for continual learning". In: *Advances in Neural Information Processing Systems* (2024).

[233] Berkman Sahiner, Weijie Chen, Ravi K Samala, and Nicholas Petrick. "Data drift in medical machine learning: implications and potential remedies". In: *The British Journal of Radiology* (2023).

[234] Tyler L Hayes and Christopher Kanan. "Online continual learning for embedded devices". In: *arXiv preprint arXiv:2203.10681* (2022).

[235] Junhuan Yang, Yi Sheng, Yuzhou Zhang, Weiwen Jiang, and Lei Yang. "On-Device Unsupervised Image Segmentation". In: *arXiv preprint arXiv:2303.12753* (2023).

[236] Geoffrey Hinton. "The forward-forward algorithm: Some preliminary investigations". In: *arXiv preprint arXiv:2212.13345* (2022).

[237] Danilo Pietro Pau and Fabrizio Maria Aymone. "Suitability of Forward-Forward and PEPITA Learning to MLCommons-Tiny benchmarks". In: *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. 2023.

[238] Yuedong Yang, Guihong Li, and Radu Marculescu. "Efficient On-device Training via Gradient Filtering". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

[239] Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. "A survey of resource-efficient llm and multimodal foundation models". In: *arXiv preprint arXiv:2401.08092* (2024).

[240] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019.

[241] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. "How to index item ids for recommendation foundation models". In: *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 2023.

[242] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[243] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[244] Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. "Latent diffusion for language generation". In: *Advances in Neural Information Processing Systems* (2024).

[245] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021.

[246] Nanyi Fei, Zhiwu Lu, Yizhao Gao, Guoxing Yang, Yuqi Huo, Jingyuan Wen, Haoyu Lu, Ruihua Song, Xin Gao, Tao Xiang, et al. "Towards artificial general intelligence via a multimodal foundation model". In: *Nature Communications* (2022).

[247] John Lee, Max Dabagia, Eva Dyer, and Christopher Rozell. "Hierarchical optimal transport for multimodal distribution alignment". In: *Advances in Neural Information Processing Systems* (2019).

[248] Thomas Theodoridis, Theocharis Chatzis, Vassilios Solachidis, Kosmas Dimitropoulos, and Petros Daras. "Cross-modal variational alignment of latent spaces". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020.

[249] Manuel Ladron De Guevara, Jose Echevarria, Yijun Li, Yannick Hold-Geoffroy, Cameron Smith, and Daichi Ito. "Cross-modal Latent Space Alignment for Image to Avatar Translation". In: *IEEE/CVF International Conference on Computer Vision*. 2023.

[250] Liqi He, Zuchao Li, Xiantao Cai, and Ping Wang. "Multi-modal latent space learning for chain-of-thought reasoning in language models". In: *AAAI Conference on Artificial Intelligence*. 2024.

[251] Yifei Yang, Zouying Cao, and Hai Zhao. "LaCo: Large Language Model Pruning via Layer Collapse". In: *arXiv preprint arXiv:2402.11187* (2024).

[252] Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. "The Unreasonable Ineffectiveness of the Deeper Layers". In: *ArXiv* (2024).

[253] Yang He and Lingao Xiao. "Structured Pruning for Deep Convolutional Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[254] Cédric Villani. *Optimal transport : old and new*. Springer, 2009.

[255] Gabriel Peyré and Marco Cuturi. "Computational Optimal Transport". In: *Foundations and Trends in Machine Learning* (2018).

[256] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. "Machine unlearning". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021.

[257] Ayush Kumar Tarun, Vikram Singh Chundawat, Murari Mandal, and Mohan Kankanhalli. "Deep regression unlearning". In: *International Conference on Machine Learning*. PMLR. 2023.

[258] Junyaup Kim and Simon S Woo. "Efficient two-stage model retraining for machine unlearning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[259] Abdelrahman Zayed, Gonçalo Mordido, Samira Shabanian, Ioana Baldini, and Sarath Chandar. "Fairness-aware structured pruning in transformers". In: *AAAI Conference on Artificial Intelligence*. 2024.