

# **Convolutional Neural Networks Pruning**

**Anis Zebiane**

*December, 8th 2025*

# 1 Abstract

Pruning convolutional neural networks is an important way to reduce computation, memory use, and energy consumption in modern deep learning models. A central idea in this area is to use channel scaling factors to decide which channels can be removed, as introduced in “Network Slimming” by Liu et al. (Proceedings of the IEEE International Conference on Computer Vision, 2017). More recent work, such as the proximal method of Bui et al. (International Conference on Machine Learning, Optimization, and Data Science, 2023), builds on this idea by using optimization tools that better handle sparsity. The goal of this paper is to provide a clear and structured explanation of these methods rather than proposing a new pruning technique. To do so, the paper reviews the mathematical background needed to understand sparsity-based pruning, summarizes the main approaches in the literature, and highlights the differences in how they enforce sparsity during training. The aim is to give readers a coherent overview of how these methods work and why they have become widely used in practice.

## 2 Introduction

Convolutional Neural Networks (CNNs) have become central to many modern applications, including image and video recognition, medical imaging, autonomous driving, and large-scale visual understanding. Their ability to learn rich feature representations has made them extremely successful in practice. However, as CNN architectures continue to grow deeper and wider to achieve higher accuracy, they also become increasingly expensive in terms of computation, memory, and energy consumption. This creates a practical mismatch between the capabilities of large CNNs and the hardware constraints of devices such as mobile phones, embedded systems, and real-time platforms.

Because of this gap, an important research direction has focused on making neural networks more efficient without a major loss in performance. One of the most widely studied approaches is network pruning, where the idea is to remove redundant or unimportant parameters from a trained model. Early influential work, such as Deep Compression by Han et al. [?], highlighted how pruning and retraining could drastically reduce network size with minimal impact on accuracy. Subsequent research has explored more structured forms of pruning, especially for CNNs, where the goal is to remove entire filters or channels.

Although pruning has become a common tool in deep learning, several questions remain open. For example, how should we measure the importance of a convolutional filter? Should pruning be done after the model is fully trained, or should it be part of the training objective itself? What types of optimization tools are most effective for encouraging sparsity, and how can we ensure that the pruning process behaves in a stable and reliable way? These questions motivate a careful look at how pruning methods have developed.

The goal of this paper is not to introduce a new pruning algorithm, but to make the reader interested in the topic by presenting the key ideas and mathematical tools behind structured CNN pruning. We focus on three influential approaches: the post-training, Taylor-based criterion of Molchanov et al. [?], the training-integrated method known as Network Slimming by Liu et al. [?], and the more recent proximal-gradient variant introduced by Bui et al. [?], which automatically enforces sparsity during training. Together, these papers show how pruning has evolved from heuristic filter scoring to principled optimization-driven techniques. By explaining their ideas in a unified way, we aim to highlight why pruning remains an active area of research and what makes it mathematically interesting.

### 3 Literature Review

Convolutional Neural Networks (CNNs) are currently one of the most effective machine learning models that are used in multiple applications such as image and video classification, object localization and classification, pedestrian recognition and car detection, in addition to many other fields. Usually, CNN models achieve strong accuracy, making them reliable; however, they face the problem of over-parameterization, which demands significant memory, computational resources, and additional training/testing time. Here is where the idea of pruning came to life in the field of neural networks, where several optimization tools were introduced in the past 10 years.

One of the important tools proposed was by Molchanov et al. (2016)[?]. This paper introduced the problem in a very interesting way, where the authors decided that pruning would be filter-based (through heuristic selection criteria), meaning that after training the CNN model until full convergence, there would be an estimation of every filter’s importance level, that is evaluated using first-order Taylor series expansion of the loss function [?]. The obtained importance level will be significant enough to know which filter would be redundant, i.e., have less impact on the output layer, as it captures the magnitude of each filter’s contribution to the loss. This also involves per-layer normalization of the criterion to obtain global scaling in the model. While this method involves accurate identification of irrelevant filters, it requires consistent checking and manual supervision. Pruning will not stop unless the developer finds the perfect moment to do so, that is, whenever there is a significant accuracy drop, which might include fine-tuning steps to recover accuracy.

The fact that CNNs contain redundant filters [?] that can be pruned after training. Molchanov et al. proposed a powerful tool to optimize this matter; however, it is done at a post-training level and requires manual supervision. This led to a significant breakthrough by Liu et al. (2017) [?] with network slimming (NS). They converted the problem of pruning into a training objective by making channel importance an adjustable parameter within the loss function. This method introduces Batch Normalization (BN) filters in the pruning process by defining the corresponding scaling parameter  $\gamma$ , as an indicator of channel importance. This parameter is determined using  $\ell_1$  regularization, saying that  $\gamma$  values of channels that are relatively small need to be removed as they are unimportant [?]. However, NS relied on standard stochastic gradient descent (SGD). Since the  $\ell_1$  term is non-smooth, SGD struggled to push the  $\gamma$  values to zero, often leaving them as small, non-zero values. This point is important: because SGD cannot produce exact zeros, the training stage of NS does not by itself identify which channels should be removed. As a result, NS necessarily becomes a three-stage process. After the  $\ell_1$ -regularized training, the authors propose to perform a separate, manual thresholding step to decide which “almost-zero”  $\gamma$  values correspond to prunable channels. Then, after these channels are removed, a final fine-tuning stage is required to recover accuracy lost by this heuristic cut. Together, these extra stages compensate for the fact that SGD alone cannot enforce sparsity in  $\gamma$ .

The inefficiency inherent in network slimming’s three-stage process [?] was directly addressed and solved by Bui et al. (2024) [?] with proximal network slimming. The advancement proposed by the authors keeps the foundational  $\ell_1$  regularization on the  $\gamma$  parameter of each channel [?], but it involves the usage of a proximal gradient method to automatically set the previously found small  $\gamma$  values to exactly zero, using what they called soft-thresholding, thus performing pruning as part of the optimization. The network emerges from the sparse, accurate, and structurally compact single training run, eliminating the need for manual thresholding and making post-pruning fine-tuning optional. The advantage is that the process is being done throughout the training, and it is completely automatic (does not require manual supervision like in [?]), and most importantly,

it guarantees convergence of the model through the *Kurdyka–Łojasiewicz (KL)* condition, which ensures that the optimization process will eventually stabilize. This stability is crucial because it confirms that the pruning happens in a consistent and reliable way during training, rather than oscillating or requiring manual stopping criteria.

## 4 Mathematical Background

The pruning methods discussed earlier rely on understanding how CNN parameters contribute to the network’s loss and how sparsity can be induced through optimization. This section introduces the mathematical notation and concepts needed to formalize these ideas, providing the foundation to understand the proposed methods.

We use  $\mathbb{R}$  to denote the set of real numbers. Real-valued  $n$ -vectors and  $m \times n$  matrices are denoted  $\mathbb{R}^n$  and  $\mathbb{R}^{m \times n}$ , respectively. The vector of all ones is denoted by  $\mathbf{1}$ , and the identity matrix (of dimension clear from context) by  $I$ . For any vector  $v \in \mathbb{R}^n$ , we write  $v_i$  for its  $i$ -th component. The Euclidean ( $\ell_2$ ) norm of  $v$  is  $\|v\|_2$ , and the  $\ell_1$ -norm is  $\|v\|_1 = \sum_i |v_i|$ . The Frobenius norm of a matrix  $W \in \mathbb{R}^{m \times n}$ , often used to measure the magnitude of convolutional filters, is  $\|W\|_F = \sqrt{\sum_{i,j} W_{ij}^2}$ . These norms are used in pruning methods such as Network Slimming [?], where the  $\ell_1$  norm encourages sparsity in the scaling factors  $\gamma$ , and in Taylor-based filter importance estimation [?], which relies on gradient magnitudes. For two sets  $(A, B) \subseteq \mathbb{R}^n \times \mathbb{R}^n$ , the distance between them is  $\text{dist}(A, B) := \inf_{a \in A, b \in B} \|a - b\|_2$ . This notion is helpful in understanding how proximal optimization, as used in Proximal Network Slimming [?], moves parameters toward sparse solutions. We denote by  $x \in \mathbb{R}^d$  an input sample and by  $y$  its corresponding target label. A neural network parameterized by weights  $W$  and scaling factors  $\gamma$  defines a mapping  $h(x; W, \gamma)$ . The empirical training loss over a dataset  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$  is written as

$$L(W, \gamma) = \frac{1}{N} \sum_{i=1}^N \ell(h(x_i; W, \gamma), y_i), \quad (1)$$

where  $\ell(\cdot, \cdot)$  is a differentiable loss function such as cross-entropy. This loss function forms the basis for first-order Taylor pruning [?] and also serves as the objective for Network Slimming [?] and Proximal Network Slimming [?]. A regularized optimization problem then seeks

$$\min_{W, \gamma} L(W, \gamma) + \lambda \|\gamma\|_1, \quad (2)$$

where  $\lambda > 0$  controls the trade-off between accuracy and sparsity. In Network Slimming [?], the  $\ell_1$  term encourages unimportant channels to shrink, while in Proximal Network Slimming [?], the optimization ensures some of these channels reach exactly zero. Gradients  $\nabla_W L$  and  $\nabla_\gamma L$  denote first-order derivatives of the loss with respect to model parameters and are used in stochastic gradient-based optimization. We use the notation  $\langle a, b \rangle := a^\top b$  for the standard inner product and “ $\circ$ ” for element-wise (Hadamard) multiplication. When referring to computational complexity, the number of floating-point operations is denoted as “FLOPs.”

## 5 Biography

Anis Zebiane is a PhD student in the Industrial and Systems Engineering (ISE) department at Lehigh University. Born and raised in Lebanon, he recently completed his Bachelor of Engineering in Computer and Communication Engineering at Notre Dame University in Lebanon, during which time he was placed constantly on the Dean's Honor List. During his time in collage, Anis developed through course projects a keen interest in conducting cutting-edge research. The research that he conducted focused on signal processing and networking systems, as well as on machine learning tools such as data sampling and regression models. Starting late 2024, he began focusing more on machine learning techniques such as anomaly detection and predictive analysis, diving deeper into various modeling techniques that further enhanced his interest in conducting cutting-edge machine learning research. Based on that, he decided to pursue a doctoral degree at Lehigh University. He believes that emerging machine learning models require significant optimization to enhance their performance beyond their current capabilities. He is currently a Rossin Fellow for the Fall 2025 semester and will proceed in the next semesters to become a Research Assistant in the ISE department at Lehigh.