Compact CNN: Multi-Objective Optimization for Architecture Search

Wassim Kharrat[1], Khadija Bousselmi[2], and Ichrack Amdouni[1]

[1] *RIADI, National School of Computer Science ENSI, University of Manouba, Tunisia*
[2] *LISTIC, University of Savoie Mont Blanc, France*

Abstract. Convolutional Neural Network (CNN) architectures have achieved remarkable success in various image analysis tasks. However, designing these architectures manually remains both labor-intensive and computationally expensive. Neural Architecture Search (NAS) has emerged as a promising approach for automating and optimizing network design. Among NAS methods, gradient-based techniques stand out for their ability to reduce computational costs while maintaining competitive performance. Nevertheless, the architectures they produce can still be demanding in terms of model size and inference time. To address this challenge, we propose a comprehensive image-analysis pipeline that combines the PC-DARTS algorithm with post-training 16-bit quantization and structured pruning. Experimental results show that our pipeline achieves an accuracy of 99.10% and an Intersection over Union (IoU) of 72.03%, while reducing the model size by up to 54%, making it well-suited for deployment in resource-constrained environments.

Keywords: Neural Architecture Search, Convolutional Neural Networks, Compression, Quantization, Pruning, Segmentation.

1 Introduction

Convolutional Neural Networks (CNNs) have become central to image classification and a wide range of deep learning tasks, with architectures such as AlexNet, VGG-Net, and Inception serving as foundational models [**1**, **2**]. Traditionally, designing these networks relied on manual engineering and trial-and-error, a process that is both time-consuming and computationally expensive [**4**, **5**]. Given the vast search space of possible configurations, exhaustive exploration is impractical [**3**].

Neural Architecture Search (NAS) was introduced to automate this process, enabling the discovery of architectures tailored to specific tasks and datasets [**3**, **4**, **5**]. NAS methods generally fall into three categories: reinforcement learning, evolutionary algorithms, and gradient-based techniques [**6**, **7**, **8**]. Gradient-based NAS is particularly attractive due to its lower computational cost and competitive accuracy. By relaxing the discrete search space into a continuous one, it allows the use of gradient descent over architectural parameters, as pioneered in DARTS [**9**] and further refined in P-DARTS [**11**], PC-DARTS [**10**], λ-DARTS [**12**], and EM-DARTS [**13**].

Although NAS automates CNN design, the resulting architectures are often over-parameterized. Model compression methods—such as structured pruning, quantization, and knowledge distillation—can reduce model size and computational complexity while preserving accuracy [**15**, **16**, **17**].

David C. Wyld et al. (Eds): NATL, MLTEC, SIGEM, SEAPP, CSEA, FUZZY, DMDBS – 2025
pp. 99-118, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.152208

In this work, we present an end-to-end pipeline that combines gradient-based NAS with a hybrid post-training compression strategy, aiming to produce compact, high-performance models suitable for resource-constrained environments.

The remainder of this paper is organized as follows: Section 2 introduces core concepts; Section 3 reviews related research; Section 4 details the proposed approach; Section 5 presents experimental results; Section 6 discusses the findings; and Section 7 concludes with directions for future work.

2 Basic Concepts

In this section, we introduce fundamental concepts essential for understanding the subsequent discussions. We present the main components and history of Convolutional Neural Networks (CNNs), Neural Architecture Search (NAS) with an emphasis on Differentiable Architecture Search (DARTS) and its variants, and conclude with several model compression techniques.

2.1 Convolutional Neural Networks (CNNs)

Deep learning has emerged as the dominant solution in artificial intelligence, obtaining great results in various scenarios like image classification, object detection, and super-resolution [**18**, **19**, **20**]. Among its architectures, Convolutional Neural Networks (CNNs) are particularly effective for computer vision due to their ability to learn hierarchical feature representations.

A typical CNN is composed of several key components:

- **Convolutional layers**: Apply learnable filters to extract local spatial patterns, producing feature maps.
- **Activation layers**: Offers non-linearity (e.g., ReLU, Sigmoid, Tanh) to model complex relationships.
- **Pooling layers**: Downsample feature maps (e.g., max or average pooling) to reduce computational cost and retain salient information.
- **Fully connected layers**: Aggregate extracted features and map them to the output space, commonly used for classification tasks.
- **Output layer**: Gives the final prediction, often using a softmax function to generate class probabilities.

The first CNN, LeNet [**21**], established the foundation by combining convolution, pooling, and backpropagation, demonstrating strong performance on handwritten digit recognition. Significant progress followed with AlexNet [**22**], which increased network depth, adopted ReLU activations, and leveraged GPU acceleration.

Subsequent architectures further refined CNN design. Network-in-Network (NiN) introduced MLP convolutions and global average pooling to reduce parameters. VGGNet [**23**] emphasized depth through stacks of 3×3 convolutions and max-pooling layers. ResNet [**24**] addressed vanishing gradient issues with residual (skip) connections, enabling the training of very deep networks.

Inception networks [**25**] captured multi-scale features using parallel convolutions of different sizes, with Inception-V3 [**26**] improving efficiency through factorized and asymmetric convolutions. Xception [**27**] extended this approach with depthwise separable convolutions, reducing computation without sacrificing accuracy. HRNet [**28**] preserved high-resolution feature representations via parallel branches that exchange information at multiple scales.

Collectively, these developments illustrate CNNs' evolution towards greater depth, efficiency, and flexibility, establishing them as the backbone of modern computer vision.

2.2 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) automates the design of neural networks, replacing manual trial-and-error with algorithmic optimization to discover efficient, high-performing architectures.

A typical NAS framework consists of three main components:

- **Search Space**: Defines the set of candidate architectures and operations. This can be global, covering the entire network, or local, as in cell-based approaches [**14**], where an optimal cell is learned and then repeated to form the full model.
- **Search Strategy**: Determines how the search space is explored:
  - *Reinforcement learning*: Models the search as a sequential decision process, rewarding architectures that achieve high performance [**8**].
  - *Evolutionary methods*: Iteratively evolve populations of architectures using genetic operators such as mutation and crossover [**7**].
  - *Gradient-based methods*: Relax the discrete search space into a continuous one, representing architectures as a directed acyclic graph (DAG) with weighted candidate operations, optimized via gradient descent [**9**].
- **Performance Estimation**: Approximates model quality without fully training each candidate, significantly reducing computational cost.

As illustrated in Figure 1, the search strategy explores the space, evaluates candidate architectures A via a performance estimator, and iterates until convergence to an optimal design.

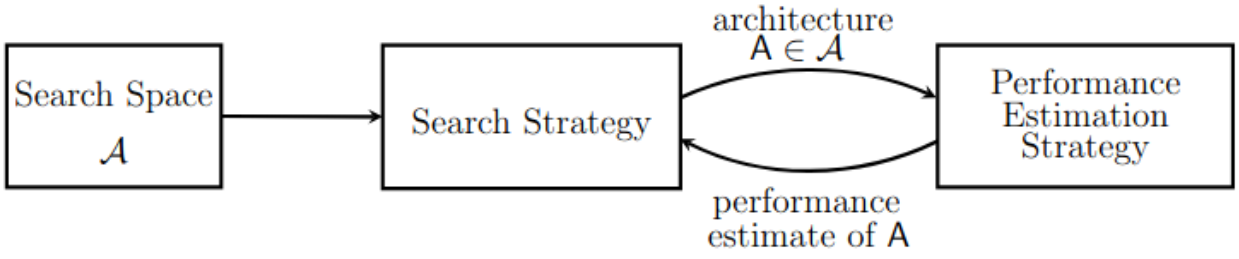

Figure 1: General workflow of a Neural Architecture Search (NAS) pipeline, showing the search space exploration, performance evaluation, and iterative optimization [**5**]

While all NAS approaches can yield high-performing models, reinforcement learning and evolutionary strategies are computationally expensive—sometimes requiring 2000–3000

GPU-days [**7**, **8**]. Gradient-based NAS methods are far more efficient, leveraging continuous relaxation and differentiable optimization to achieve faster convergence with lower computational resources [**9**, **10**, **11**, **12**, **13**].

2.3 Differentiable Architecture Search (DARTS)

DARTS is a gradient-based NAS method that represents architecture design as a bi-level optimization problem within a cell-based search space [**9**]. It jointly learns:

- the architecture parameters α, which define the network structure
- the weights of the operations ω, which correspond to the candidate operations.

The optimization is expressed as:

$$\min_{\alpha} \mathcal{L}_{\text{val}}(w(\alpha), \alpha) \tag{1}$$

$$\text{s.t. } w(\alpha) = \arg\min_{\omega} \mathcal{L}_{\text{train}}(w(\alpha)) \tag{2}$$

With $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{train}}$ respectively being the validation loss and training loss. By relaxing the discrete search space into a continuous one, DARTS allows gradient-based updates of α and ω, greatly reducing computational cost as opposed to discrete NAS methods.

During the search phase, the network consists of 8 cells: six *normal* cells for feature extraction and two *reduction* cells for downsampling. In the evaluation phase, 20 cells are stacked (18 normal and 2 reduction), with reduction cells positioned at one-third and two-thirds of the network depth.

Several extensions improve DARTS:

- **P-DARTS** [**11**] reduces the search-to-evaluation gap by progressively increasing the number of cells during search ($5 \to 11 \to 17$), producing more stable architectures.
- **PC-DARTS** [**10**] lowers computational cost by applying candidate operations to only a subset of input channels, concatenating transformed and untransformed features. This allows larger batch sizes and more stable gradients.
- **λ-DARTS** [**12**] addresses the over-selection of skip connections by introducing a regularization term that encourages balanced operation selection across layers:

$$\Lambda_{\pm}(\omega, \alpha) = \frac{1}{\binom{L}{2}} \sum_{\ell < \ell'} \frac{\ell g^T \ell' g}{|\ell g^T \ell' g|}, \qquad g = \nabla_{\ell_p} \mathcal{L}(\omega, \alpha) \tag{3}$$

  The bi-level optimization objective becomes:

$$\min_{\alpha} \mathcal{L}^{val}(w(\alpha), \alpha), \quad \text{s.t. } w(\alpha) = \arg\min_{\omega} \left[\mathcal{L}^{train}(\omega, \alpha) - \lambda \Lambda^{train}(\omega, \alpha)\right] \tag{4}$$

  where λ controls the regularization strength.
- **EM-DARTS** [**13**] introduces stochastic "edge mutation," randomly selecting operations on some edges instead of always using weighted sums, improving exploration and reducing the risk of premature convergence.

These refinements enhance DARTS' efficiency, stability, and robustness, enabling the discovery of competitive neural architectures.

2.4 Compression Techniques

Model compression reduces the computational demands of deep neural networks while maintaining predictive performance [**15**, **16**]. The most commonly used techniques are pruning, quantization, and knowledge distillation.

2.4.1 Pruning

Pruning removes redundant or low-impact components from a trained model, resulting in a smaller, faster network [**15**, **16**, **17**]. It is typically classified into two types:

- **Unstructured pruning**: Eliminates individual weights irrespective of their location, achieving high sparsity but generating irregular patterns that are difficult to accelerate on standard hardware.
- **Structured pruning**: Removes entire filters, channels, or layers, producing hardware-friendly models, though potentially incurring higher accuracy loss if not carefully applied.

Several studies have explored pruning techniques to reduce neural network size. Han et al. [**33**] proposed removing all weights below a threshold from a trained model, followed by retraining; this achieved compression factors of 9× for AlexNet and 13× for VGG16 on ImageNet. Guo et al. [**34**] introduced Dynamic Network Surgery, combining iterative pruning and "splicing," allowing pruned connections to be restored if useful, achieving 108× compression on LeNet and 17.7× on AlexNet. Carreira-Perpiñán et al. [**35**] formulated pruning as an optimization problem, minimizing loss under pruning-cost constraints. Shiheb et al. [**37**] presented an unstructured pruning framework for Mamba models, combining weight magnitude, gradients, and global parameter allocation to remove up to 70% of parameters without significant accuracy loss, though sparse matrices complicate deployment [**36**].

Structured pruning instead removes entire components (filters or feature maps). Li et al. [**38**] pruned CNN filters via the L1 norm, cutting VGGNet/ResNet inference cost by 30%. He et al. [**39**] proposed "soft" L2-based pruning, where removed filters can be reactivated, improving ResNet inference speed by 30% on ImageNet. Ye et al. [**40**] used BatchNorm scaling factors to prune ResNet filters on CIFAR-10, reducing model size by 39% with minimal accuracy loss. Li et al. [**41**] applied reinforcement learning (TD3) to adaptively prune filter groups using a multi-objective reward balancing accuracy, FLOPs, and parameter count.

Structured pruning accelerates inference by fully removing filters but may degrade accuracy if applied too aggressively.

2.4.2 Quantization

Quantization reduces the quality of the representation of model parameters , reducing memory usage and accelerating inference [**15**, **16**]. While networks are typically stored in 32-bit floating-point format, many tasks tolerate lower precision (e.g., 16-, 8-, or 4-bit) with minimal accuracy loss. This enables simpler arithmetic operations, lower power consumption, and efficient deployment on resource-constrained devices. Many studies have explored quantization as a way to reduce model size. Zhou et al. [**43**] and Chen et al. [**44**] focused on weight quantization, which lowers memory requirements with little impact on inference cost. Jacob et al. [**42**] proposed quantizing both weights and activations, enabling full 8-bit

(INT8) inference; for instance, an INT8 ResNet loses only 2% accuracy on ImageNet while cutting memory use by 4×. Other work investigates even lower precisions [**45**, **46**]. Zhou et al. [**47**] showed that 4-bit quantization can halve memory consumption with negligible accuracy loss, whereas going below 4 bits (e.g., 2-bit) causes a sharp performance drop.

In short, quantization is an effective way to shrink models and ease deployment on resource-limited devices, but overly aggressive precision reduction can significantly degrade performance.

2.4.3 Knowledge Distillation

Knowledge Distillation transfers the learned behavior of a large *teacher* model to a smaller *student* model [**15**], preserving accuracy while reducing memory requirements. The main paradigms include:

- **Offline**: A pre-trained teacher provides soft targets or intermediate features to guide the student.
- **Online**: Teacher and student are trained simultaneously, enabling dynamic knowledge exchange.
- **Self-distillation**: A single model uses its deeper layers to supervise its shallower layers, simplifying training while retaining the benefits of distillation.

Early work in this area dates back to Buciluǎ et al. [**48**], who introduced the concept of knowledge transfer. Their method trained a small model on data labeled by a larger one, allowing the student to approximate the teacher's performance with shorter training time than learning directly from the raw data. The term knowledge distillation was later popularized by Hinton et al. [**49**]. Ba et al. [**50**] proposed using logits—the outputs of the softmax layer—as the transfer signal, with the distillation loss measuring the gap between teacher and student predictions. Romero et al. [**51**] extended this idea by transferring intermediate feature representations, aligning the student's internal outputs with those of the teacher. This approach achieved a 10.4× compression on CIFAR-10 while improving accuracy.

The choice of what to distill must be paired with an appropriate strategy. Offline distillation trains the teacher first and then transfers its knowledge [**49**, **51**], while online distillation trains teacher and student jointly [**52**]. Self-distillation uses a single model that plays both roles [**53**].

Overall, knowledge distillation is a promising technique for building lightweight neural networks that retain strong classification performance while significantly reducing computational cost.

These techniques are critical for creating compact, energy-efficient networks suitable for real-time and low-power applications.

3 Related Work

Neural Architecture Search (NAS) has been extensively investigated in the literature [**7**, **8**, **9**], with numerous algorithms exploring different families of optimization techniques. This work

focuses primarily on gradient-based NAS methods. Among these, the DARTS algorithm [**9**] is a foundational approach that applies bi-level optimization within a one-shot architecture framework [**14**].

Several studies have addressed the limitations of the original DARTS formulation. For instance, P-DARTS [**11**] tackles the discrepancy between the search and evaluation phases. PC-DARTS [**10**] reduces computational cost by constraining the number of channels used during the search process. Furthermore, λ-DARTS [**12**] and EM-DARTS [**13**] enhance the fairness of operation selection—achieved through layer alignment in the former and edge mutation in the latter.

To the best of our knowledge, no other work tries to combine NAS algorithms with compression techniques aimed at reducing the size of the resulting model. LMD-DARTS [**30**] is the only method that integrates a differentiable architecture search (NAS) algorithm with a form of pruning. In this approach, pruning is not applied to the final network to reduce its size; rather, it is embedded directly into the search process. During the optimization of the architecture, a learned sampling mechanism progressively eliminates operations deemed less relevant, thereby shrinking the search space and accelerating convergence. The goal is to make the NAS procedure more efficient, not to compress the resulting model. Nevertheless, LMD-DARTS demonstrates the potential of coupling NAS techniques with pruning strategies, hinting at opportunities for future work aimed at jointly optimizing model size and performance.

4 Proposed Approach

This section presents our work. The study begins with a comparative evaluation of five Neural Architecture Search (NAS) algorithms. We then describe the adopted search space, followed by an overview of the model compression techniques and their configurations.

4.1 Search Strategy

We compare five gradient-based NAS algorithms: DARTS, P-DARTS, PC-DARTS, λ-DARTS, and EM-DARTS. Based on the results, the algorithm with the best performance will be used in the final experiments.

4.2 Search Space

Our search space follows the original DARTS design [**9**, **14**]. The candidate operations in the operation set O include: 3×3 and 5×5 depthwise separable convolutions, 3×3 and 5×5 dilated depthwise separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero operations (applied with stride 1 where applicable).

Each cell contains seven nodes separated into 2 input nodes that pass their outputs to 4 intermediate nodes. This is then followed with 1 output node. The output node aggregates the outputs of all intermediate nodes via depthwise concatenation. The final network is constructed by stacking multiple cells, with reduction cells positioned at one-third and two-thirds of the network depth, and all other cells being normal. Figure 2 illustrates the structure of a single cell.

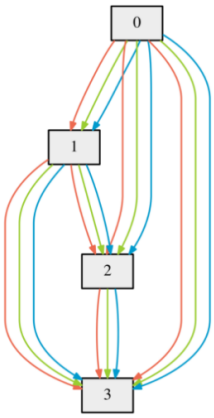

Figure 2: Structure of a cell in the DARTS search space. Each directed edge represents a candidate operation from O, and the output node concatenates all intermediate outputs [**9**]

For the final architecture, we stack a total of 20 cells separated into 18 normal cells used in tandem with 2 reduction cells, placed at one-third and two-thirds of the network depth.

4.3 Compression Methods

4.3.1 Individual Compression Techniques

To reduce computational complexity and model size, we evaluated several state-of-the-art compression strategies, each configured with carefully selected hyperparameters to balance compression rate and predictive performance.

- **Pruning:** Structured pruning was applied to convolutional layers at three sparsity levels: 40%, 60%, and 80%, exploring a range from conservative to aggressive compression:
  - 40%: moderate compression with minimal risk of accuracy loss.
  - 60%: balanced reduction, achieving substantial parameter savings while maintaining performance.
  - 80%: aggressive pruning to evaluate the maximum feasible compression before performance degrades significantly.

  Filters were ranked by their L1-norm, which reflects their contribution to the output—higher L1-norm filters are preserved. To prevent severe accuracy loss, pruning was not applied to:
  - Depthwise convolution layers, as they are lightweight and crucial for feature representation.
  - Layers with few output channels, where aggressive pruning could cause underparameterization.

  These sparsity levels allow a systematic exploration of the trade-off between compression and performance.
- **Quantization:** We adopted uniform 16-bit quantization to reduce memory footprint and accelerate inference while maintaining sufficient numerical precision. This bit-width offers a practical balance between storage efficiency and computational accuracy:

  - Compared to 32-bit floating-point, 16-bit reduces memory usage by roughly half and lowers bandwidth demands, enabling faster inference on resource-limited hardware.
  - Unlike lower-precision formats (e.g., 8-bit), 16-bit preserves a wide dynamic range, accurately representing both small and large weights, minimizing underflow, overflow, and rounding errors.
  - Modern accelerators (GPUs, TPUs) provide optimized support for 16-bit operations, allowing higher throughput without compromising training or inference stability.

  Overall, 16-bit quantization strikes a balance between compression and precision, making it a practical choice for resource-constrained deep learning deployments.

- **Knowledge Distillation:** To transfer knowledge from a high-capacity teacher model to smaller student networks, we investigated four configurations varying along two dimensions:
  - **Initial number of channels:** 16 or 36
  - **Number of layers:** 8 or 14

  These choices explore the trade-off between model width (channels) and depth (layers), affecting representational capacity and computational cost:
  - **Init. Channels = 16:** Lightweight, reducing model size and inference cost while preserving essential feature extraction.
  - **Init. Channels = 36:** Wider configuration to capture more complex patterns, closer to the teacher's capacity.
  - **Layers = 8:** Shallow architecture for low latency and memory usage, matching the minimal depth from the NAS search phase.
  - **Layers = 14:** Deeper model enhancing hierarchical feature representation, potentially improving accuracy at slightly higher computational cost.

  Combining these dimensions yields four student architectures, allowing systematic evaluation of how width and depth affect knowledge transfer.

  Distillation uses two complementary loss components:
  - **Hard Loss:** Standard loss comparing student predictions to ground-truth labels.
  - **Soft Loss:** Encourages the student to mimic the teacher's softened outputs, obtained by scaling logits with a temperature T.

  The overall objective, applied to update only the student weights, is:

$$\mathcal{L} = \alpha \cdot \text{loss}_{soft} + (1 - \alpha) \cdot \text{loss}_{hard}, \tag{5}$$

  where α balances imitation of the teacher and learning from ground truth, with values chosen empirically.

4.3.2 Hybridization Process

In a second phase, we explore the combination of pruning and quantization to maximize model compression while minimizing accuracy loss. Pruning reduces the number of active weights, and quantization further compresses their numerical representation.

The optimal configuration (e.g., pruning sparsity level and quantization bit width) is determined based on the results of the individual techniques, allowing selection of the best trade-off between model size and predictive performance.

We will analyze the effectiveness of each method individually and in combination to evaluate the performance of combining these techniques, aiming to reduce model size while keeping accuracy and generalization.

5 Experiments and Results

In this section, we present the results of applying various NAS algorithms on the CIFAR10 dataset. We then report the performance of the best-performing algorithm on the ISSLIDE dataset. Finally, we discuss the impact of different compression techniques on the selected model.

5.1 CIFAR10 Dataset

CIFAR10 serves as a benchmark dataset to evaluate which DARTS variant performs best on our hardware setup. It consists of 60,000 images of size 32×32 pixels [**31**], distributed across 10 classes with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images.

5.2 Algorithm Performance on CIFAR10

We applied the five NAS algorithms to CIFAR10 and evaluated them using the following performance metrics:

- **Accuracy**: The proportion of correct predictions out of all predictions, providing an overall measure of model performance across classes.
- **Recall**: Measures the ability of the model to correctly identify all positive samples, calculated as the ratio of true positives to total actual positives.
- **F1-Score**: Balances precision and recall by taking their harmonic mean. Particularly useful for imbalanced datasets, it penalizes extreme values and provides a more reliable performance measure than accuracy alone.
- **Number of Parameters**: The count of learnable parameters, indicating the model's size and computational complexity during training and inference.
- **Training Time**: The duration required to update the model, influenced by hyperparameters such as epochs, batch size, and learning rate.

Table 1 summarizes the results obtained for the different NAS algorithms.

Used Algorithm	Accuracy	Recall	F1-score	Nb. of Parameters	Training Time (in S)
DARTS	96.99	96.98	96.85	2.8 M	99 092
P-DARTS	97.20	96.96	96.89	3.34 M	56 456
PC-DARTS	97.15	97.11	96.99	2.9 M	115 510
λ-DARTS	94.66	94.6	94.33	0.7 M	44 893
EM-DARTS	96.92	75.00	75.00	4.81 M	84 824

Table 1: Comparative performance of DARTS and its variants on CIFAR-10

The results indicate that, while the overall performance of the algorithms is broadly similar, PC-DARTS achieves the highest Recall and F1-score, with its Accuracy surpassed only by P-DARTS. Furthermore, PC-DARTS maintains a competitive parameter count, exceeded only by DARTS and λ-DARTS, both of which exhibit lower predictive performance.

Several intrinsic characteristics of PC-DARTS further support its selection:

- By processing only a subset of input channels for each operation, PC-DARTS significantly reduces the computational cost of the architecture search compared to other methods.
- This computational efficiency allows for increased hyperparameter settings—most notably, the batch size, which was raised from 64 to 256 without exceeding hardware limitations. A larger batch size improves training stability by reducing fluctuations in gradient updates.
- The larger batch size also enables more effective processing of diverse datasets, enhancing the algorithm's generalization across a wide variety of images.

Considering these factors, we adopt PC-DARTS for experiments on our final dataset, ISSLIDE, which is introduced in the following section.

5.3 ISSLIDE: InSAR Dataset for Slow Sliding Area Detection

In this work, we apply the PC-DARTS algorithm to the ISSLIDE dataset, a curated collection of satellite imagery designed for segmentation tasks related to terrain displacement. The dataset comprises over 200 documented ground movement events across various regions in France [**32**]. A key strength of ISSLIDE is its high-quality annotations, prepared in collaboration with geomorphological experts, which enhance its suitability for deep learning applications.

The dataset contains 13,230 samples, each acquired over temporal baselines of 6, 12, and 18 days, evenly distributed across these intervals. Each sample consists of three 100×100 pixel grayscale images:

- **Coherence:** Quantifies spatial coherence between two radar acquisitions, ranging from 0 (low coherence, indicating significant scene changes) to 1 (high coherence, indicating temporal stability). High coherence typically corresponds to stable terrain, while low coherence may suggest potential movement or land cover changes.
- **Phase:** Encodes phase information from the interferogram, with values between $-\pi$ and $+\pi$. Variations in grayscale provide insights into surface displacement patterns. Regions with poor coherence may exhibit noise or phase decorrelation, limiting interpretability.

- **Segmentation (Ground Truth):** The segmentation mask serves as the ground truth label for training and evaluation. Pixels labeled as 1 (white) indicate identified terrain movement, while pixels labeled as 0 (black) represent the background. These annotations were validated by geomorphology experts, ensuring accuracy and relevance.

An illustrative example of a sample triplet from ISSLIDE is shown in Figure 3.

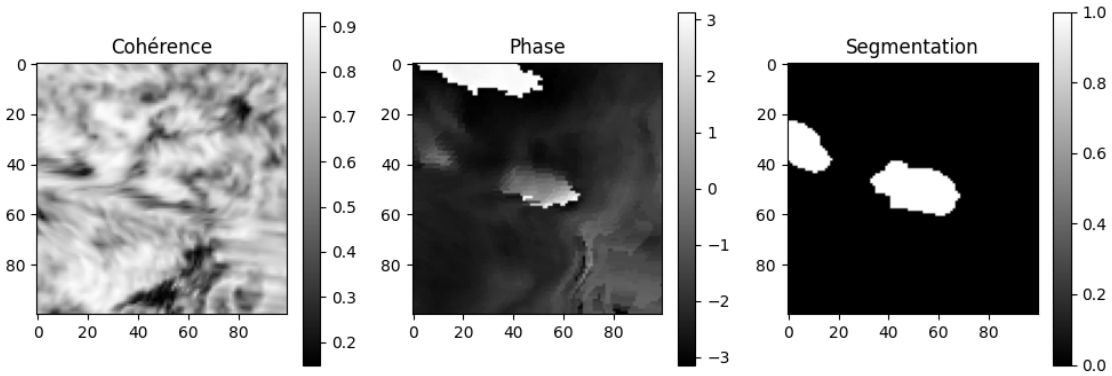

Figure 3: Example sample from the ISSLIDE dataset, showing Coherence, Phase, and Segmentation images

5.4 Results on the ISSLIDE Dataset

Table 2 summarizes the performance of the PC-DARTS algorithm on the ISSLIDE dataset. To further evaluate segmentation performance, we report the Intersection over Union (IoU) metric alongside the already mentioned metrics.

This new metric describes the common part between the predicted segmentation and the ground truth, offering a more precise assessment of pixel-wise classification performance. It is defined as:

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}} \tag{6}$$

Where:

- **Intersection:** the number of pixels in the segmented class correctly predicted.
- **Union:** the total number of pixels present in either the predicted segmentation, the ground truth, or both.

Figures 4 and 5 illustrate the basic building blocks used to construct the final architecture, namely the normal and reduction cells.

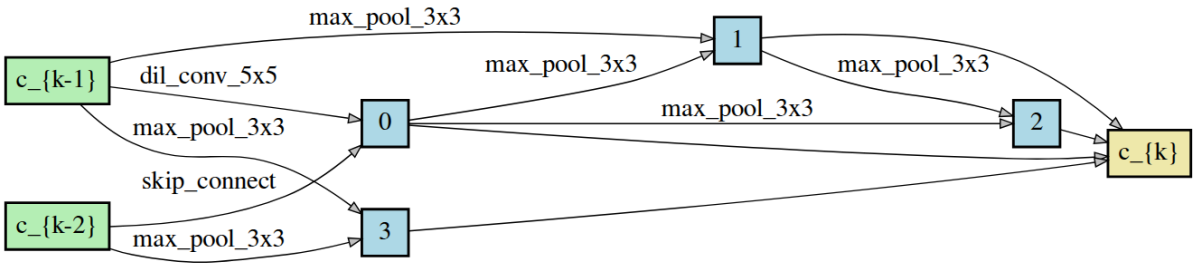

Figure 4: Normal cell proposed after applying the PC-DARTS algorithm on the ISSLIDE dataset

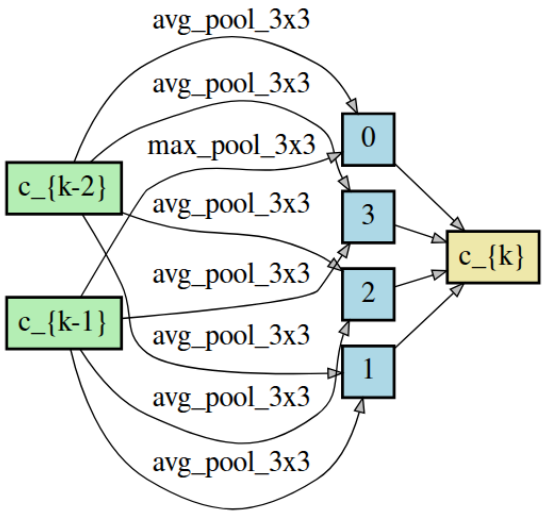

Figure 5: Reduction cell proposed after applying the PC-DARTS algorithm on the ISSLIDE dataset

Used Algorithm	**Accuracy**	**Recall**	**F1-score**	**IOU**	**Nb. of Parame-ters**	**Training Time (in S)**
PC-DARTS	99.44	96.44	93.73	83.01	1.56 M	70 732

Table 2: Performance metrics for PC-DARTS on the ISSLIDE dataset

The results demonstrate the strong generalization and segmentation capabilities of the PC-DARTS architecture on the ISSLIDE dataset. In particular, the model achieves an IoU of 83.01%, indicating that over 80% of the pixels corresponding to terrain displacement regions were correctly identified. These findings highlight the effectiveness and transferability of PC-DARTS for satellite image-based segmentation tasks. Figure 6 illustrates sample segmentation results obtained using the PC-DARTS model.

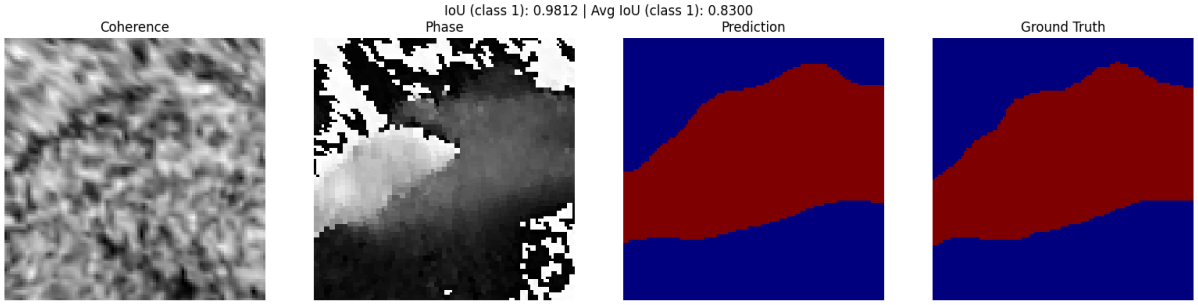

Figure 6: Segmentation output of the PC-DARTS architecture on a sample from the ISSLIDE dataset, illustrating accurate identification of terrain-displacement regions

5.5 Compression Results

We present through the following tables the results of application of the compression techniques presented in section 4.3.

5.5.1 Pruning

Percentage of Pruning	**Accuracy**	**Recall**	**IOU**	**Nb. of Parameters**	**Model Size (en MB)**
0.4	99.37	94.98	80.8	1.44 M	5.9 (Reduction of 9%)
0.6	99.10	90.48	72.04	1.375 M	5.7 (Reduction of 12%)
0.8	97.52	72.14	37.33	1.305 M	5.5 (Reduction of 15%)

Table 3: Performance of the PC-DARTS model on the ISSLIDE dataset under different pruning levels

As shown in Table 3, pruning up to 60% of the filters still maintains acceptable performance, with only a 0.34% decrease in accuracy, a 6% reduction in recall, and roughly a 10% drop in IoU, while also reducing the model size by 12%. These results suggest that a large portion of the filters in the baseline network are redundant or contribute minimally to the final prediction. By removing up to 60% of them, the model retains most of its discriminative power, as the remaining filters preserve the essential feature representations required for accurate segmentation. However, when sparsity becomes too high (e.g., 80%), pruning begins to remove filters that capture fine-grained spatial or boundary information, causing a sharp decline in IoU despite only modest additional storage savings. This trade-off underscores the importance of selecting a pruning threshold that balances compression with the preservation of critical feature structures.

5.5.2 Quantization

Accuracy	**Recall**	**IOU**	**Nb. of Parameters**	**Model Size (en MB)**
99.44	96.43	83.00	1.56 M	3.3 (Reduction of 50%)

Table 4: Performance of the PC-DARTS model after 16-bit quantization

As shown in Table 4, applying 16-bit quantization reduced the model size by approximately 50% while nearly preserving the accuracy, recall, and IoU of the original network. These results indicate that the model's parameters and activations do not require the full precision of 32-bit floating-point representation to maintain high predictive performance. Encoding them in 16-bit format is sufficient to capture the essential information with minimal numerical error, enabling the network to operate more efficiently without compromising its ability to detect fine details or maintain stable boundaries. This demonstrates that half-precision arithmetic is well-suited for resource-constrained environments, where reducing memory footprint and accelerating inference are critical, and highlights that quantization can be effectively lossless or near-lossless when the bit-width is carefully selected.

5.5.3 Knowledge Distillation

Init. Channels	**Nb. of Layers**	**Accuracy**	**Recall**	**IOU**	**Nb. of Parameters**	**Model Size (in MB)**
16	8	97.83	75.83	41.62	115 042	0.5
16	14	97.88	76.59	42.90	218 754	0.98
36	8	97.68	71.91	36.05	557 642	2.3
36	14	97.98	80.58	47.05	1.1 M	4.4

Table 5: Performance of the distilled PC-DARTS model under various configurations of initial channels and network depth

As shown in Table 5, knowledge distillation allowed substantial reductions in the number of parameters and model size — around 91% compared to the original network — while causing only a modest drop in accuracy (approximately 2%). However, the IoU metric reveals a pronounced deterioration in segmentation performance, with the best student configuration achieving only around 47–50%, far below the baseline.

These results highlight a key trade-off: while the student network can approximate the teacher's global decision boundaries and preserve high classification accuracy, it struggles to replicate the fine-grained spatial information necessary for dense prediction tasks like segmentation. This limitation is exacerbated by the mismatch between the NAS-optimized teacher architecture and the simpler student models. The PC-DARTS teacher generates rich, high-dimensional feature maps with complex dependencies, whereas the student — especially configurations with only 16 channels and 8 layers — lacks sufficient representational capacity. Even the largest student configuration (36 channels, 14 layers) partially recovers segmentation ability but still falls short of the desired performance.

These findings indicate that, under the tested settings, knowledge distillation alone is insufficient for our objectives. More advanced approaches, such as attention transfer, feature-level alignment, or increasing the student's capacity, would be required to improve IoU while retaining compression benefits.

5.5.4 Hybrid approach

We found that the most effective techniques in our case are, in order of impact: quantization, pruning, and finally knowledge distillation. We therefore explored a hybrid approach that combines the two best techniques — quantization and pruning — to maximize model size

reduction while ensuring that performance remains within acceptable limits. The results of this hybrid strategy are presented in Table 6.

Used Algorithm	**Percentage of Pruning**	**Applied Quantization**	**Accuracy**	**Recall**	**IOU**	**Nb. of Parameters**	**Model Size (in MB)**
PC-DARTS	0.6	16-bit	99.10	90.47	72.03	1.375 M	3.0 (Reduction of 54%)

Table 6: Performance of the PC-DARTS model after combining pruning (60%) and 16-bit quantization on the ISSLIDE dataset

Qualitatively, these results demonstrate the effectiveness of combining pruning and quantization for model compression. The hybrid strategy achieves a 54% reduction in model size while preserving most of the predictive performance: the drop in accuracy is minimal (0.34%), and the reductions in recall (6%) and IoU (10%) remain moderate. This indicates that pruning and quantization complement each other effectively: pruning eliminates redundant filters, while quantization optimizes the numerical representation of the remaining weights without compromising their expressiveness. Overall, this hybrid approach strikes a practical balance between compression and segmentation quality, making it well-suited for resource-constrained environments where maintaining acceptable pixel-wise precision is essential.

6 Discussion

The original PC-DARTS model achieved strong performance on the ISSLIDE dataset, with an accuracy of 90% and approximately 80% of segmented pixels correctly identified. While other evaluated NAS algorithms produced comparable results, PC-DARTS demonstrated superior effectiveness for segmentation tasks involving grayscale input images and shows promise for application to similar remote sensing datasets.

Regarding the compression techniques explored, 16-bit quantization reduced the model size by approximately 50% with negligible impact on accuracy, recall, or IoU. Structured pruning also proved effective, removing redundant convolutional filters and reducing the model size by around 12%, with only minor decreases in accuracy (-0.34%), recall (-6%), and IoU (-10%).

The combination of quantization and pruning—the hybrid approach—achieved the most significant reduction in model size while maintaining a modest performance degradation similar to that of pruning alone. Notably, the order of applying these two techniques did not affect the final outcome, highlighting the robustness and efficiency of post-training compression methods.

In contrast, knowledge distillation did not yield satisfactory results. Reducing the model to the smaller architecture used during the NAS search phase led to a substantial decline in segmentation performance, well beyond acceptable limits. This emphasizes that, for dense prediction tasks such as segmentation, student models must retain sufficient capacity to capture fine-grained spatial information.

Figure 7 provides an overview of the proposed solution pipeline, summarizing the main stages from architecture search to compression and deployment.

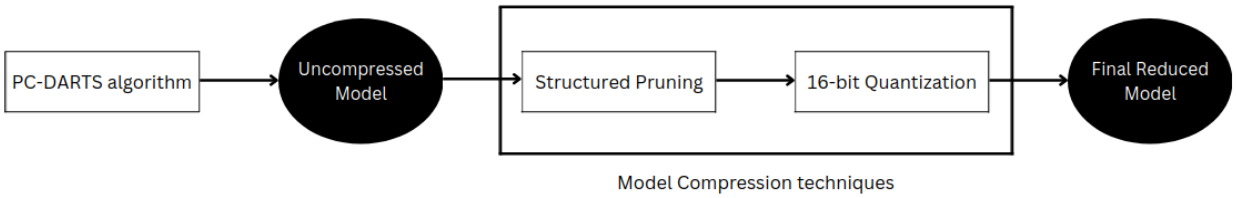

Figure 7: Overview of the proposed solution pipeline, summarizing the main stages from architecture search to compression and deployment.

7 Conclusion

This work investigated various techniques in neural architecture search (NAS), with a particular focus on gradient-based algorithms due to their reduced computational cost. Our main contributions were conducting a comparative evaluation of several gradient-based methods, including DARTS, P-DARTS, PC-DARTS, λ-DARTS, and EM-DARTS. Another contribution was examining multiple model compression techniques—pruning, quantization, and knowledge distillation—to reduce model size while preserving performance. These analyses led to the achievement of our main goal: the design of our proposed pipeline, which integrates NAS with post-training compression to achieve optimized, compact, and high-performing models.

For future work, it would be valuable to explore the transferability of this pipeline to other datasets and application domains, thereby demonstrating its broader applicability. Further, modifying or extending the pipeline could open avenues for more innovative approaches, potentially enhancing both efficiency and predictive performance across diverse tasks.

Refrences

1. Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021).

2. Alom, Md Zahangir, et al. "The history began from alexnet: A comprehensive survey on deep learning approaches." arXiv preprint arXiv:1803.01164 (2018).

3. Ren, Pengzhen, et al. "A comprehensive survey of neural architecture search: Challenges and solutions." ACM Computing Surveys (CSUR) 54.4 (2021): 1-34.

4. Salmani Pour Avval, Sasan, et al. "Systematic review on neural architecture search." Artificial Intelligence Review 58.3 (2025): 73.

5. Elsken, Thomas, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey." Journal of Machine Learning Research 20.55 (2019): 1-21.

6. Ali, Sarwat, and M. Arif Wani. "Gradient-based neural architecture search : A comprehensive evaluation." Machine Learning and Knowledge Extraction 5.3 (2023)

7. Real, Esteban, et al. "Regularized evolution for image classifier architecture search." Proceedings of the aaai conference on artificial intelligence. Vol. 33. No. 01. 2019.

8. Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

9. Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search." arXiv preprint arXiv:1806.09055 (2018).

10. Xu, Yuhui, et al. "Pc-darts: Partial channel connections for memory-efficient architecture search." arXiv preprint arXiv:1907.05737 (2019).

11. Chen, Xin, et al. "Progressive darts: Bridging the optimization gap for nas in the wild." International Journal of Computer Vision 129 (2021): 638-655.

12. Movahedi, Sajad, et al. "Λ-DARTS: Mitigating Performance Collapse by Harmonizing Operation Selection among Cells." arXiv preprint arXiv:2210.07998 (2022).

13. Liu, Huaxiong, et al. "EM-DARTS : Preventing Performance Collapse in Differentiable Architecture Search with The Edge Mutation Mechanism." (2025).

14. Bender, Gabriel, et al. "Understanding and simplifying one-shot architecture search." International conference on machine learning. PMLR, 2018.

15. Garbay, Thomas. Zip-CNN. Diss. Sorbonne Université, 2023.

16. Li, Zhuo, Hengyi Li, and Lin Meng. "Model compression for deep neural networks: A survey." Computers 12.3 (2023): 60.

17. Zhang, Chengming, et al. "Clicktrain: Efficient and accurate end-to-end deep learning training via fine-grained architecture-preserving pruning." Proceedings of the 35th ACM International Conference on Supercomputing. 2021.

18. Zhao, Xia, et al. "A review of convolutional neural networks in computer vision." Artificial Intelligence Review 57.4 (2024): 99.

19. Alzubaidi, Laith, et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." Journal of big Data 8 (2021): 1-74.

20. Alom, Md Zahangir, et al. "The history began from alexnet: A comprehensive survey on deep learning approaches." arXiv preprint arXiv:1803.01164 (2018).

21. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

22. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

23. Simonyan, Karen, and Andrew Zisserman. " deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556(2014)

24. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

25. Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

26. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

27. Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

28. Wang, Jingdong, et al. "Deep high-resolution representation learning for visual recognition." IEEE transactions on pattern analysis and machine intelligence 43.10 (2020): 3349-3364.

29. Liu, Kai, et al. "Low-bit model quantization for deep neural networks: A survey." arXiv preprint arXiv:2505.05530 (2025).

30. Li, Zhongnian, et al. "LMD-DARTS: Low-Memory, Densely Connected, Differentiable Architecture Search." Electronics 13.14 (2024): 2743.

31. Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.

32. Bralet, Antoine, et al. "ISSLIDE : A new InSAR dataset for Slow SLIding area DEtection with machine learning." IEEE Geoscience and Remote Sensing Letters 21 (2024) : 1-5

33. Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems 28 (2015).

34. Guo, Yiwen, Anbang Yao, and Yurong Chen. "Dynamic network surgery for efficient dnns." Advances in neural information processing systems 29 (2016).

35. Carreira-Perpinán, Miguel A., and Yerlan Idelbayev. "“learning-compression” algorithms for neural net pruning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

36. Gale, Trevor, Erich Elsen, and Sara Hooker. "The state of sparsity in deep neural networks." arXiv preprint arXiv:1902.09574 (2019).

37. Shihab, Ibne Farabi, Sanjeda Akter, and Anuj Sharma. "Efficient unstructured pruning of mamba state-space models for resource-constrained environments." arXiv preprint arXiv:2505.08299 (2025).

38. Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).

39. He, Yang, et al. "Soft filter pruning for accelerating deep convolutional neural networks." arXiv preprint arXiv:1808.06866 (2018).

40. Ye, Jianbo, et al. "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers." arXiv preprint arXiv:1802.00124 (2018).

41. Li, Qianxi, and Wenhui Zhang. "Structured Pruning Based on Reinforcement Learning for CNN." Academic Journal of Computing & Information Science 8.7 (2025): 79-86.

42. B. Jacob, S. Kligys, B. Chen et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference", en, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT : IEEE, juin 2018

43. Zhou, Aojun, et al. "Incremental network quantization: Towards lossless cnns with low-precision weights." arXiv preprint arXiv:1702.03044 (2017).

44. Chen, Wenlin, et al. "Compressing neural networks with the hashing trick." International conference on machine learning. PMLR, 2015.

45. Park, Eunhyeok, and Sungjoo Yoo. "Profit: A novel training method for sub-4-bit mobilenet models." European conference on computer vision. Cham: Springer International Publishing, 2020.

46. Esser, Steven K., et al. "Learned step size quantization." arXiv preprint arXiv:1902.08153 (2019).

47. Zhuo, Shaojie, et al. "An empirical study of low precision quantization for TinyML." arXiv preprint arXiv:2203.05492 (2022).

48. Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. 2006.

49. G. Hinton, O. Vinyals et J. Dean, Distilling the Knowledge in a Neural Network, arXiv :1503.02531 [cs, stat], mars 2015

50. Ba, Jimmy, and Rich Caruana. "Do deep nets really need to be deep?." Advances in neural information processing systems 27 (2014).

51. A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta et Y., Bengio, FitNets : Hints for Thin Deep Nets, arXiv :1412.6550 [cs], mars 2015.

52. Zhang, Ying, et al. "Deep mutual learning." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

53. Zhang, Linfeng, et al. "Be your own teacher: Improve the performance of convolutional neural networks via self distillation." Proceedings of the IEEE/CVF international conference on computer vision. 2019.