Intelligent Architecture Design for Deep Convolutional Neural Networks Using Layer Wise Optimization

Zhu Jiping
Department of Modern Mechanics, University of Science and Technology of China, Hefei, China.
jipingmech@ustc.edu.cn

Article Info
Journal of Elaris Computing Nexus
https://elarispublications.com/journals/ecn/ecn_home.html

Received 16 May 2025
Revised from 30 July 2025
Accepted 16 August 2025
Available online 02 September 2025
Published by Elaris Publications.

https://doi.org/10.65148/ECN/2025013

Corresponding author(s):
Zhu Jiping, Department of Modern Mechanics, University of Science and Technology of China, Hefei, China.
Email: jipingmech@ustc.edu.cn

Abstract – The recent technology progress in deep learning has resulted in significant breakthroughs in image classification. Nevertheless, it is a big challenge to run complicated deep neural networks (CNNs) on devices requiring minimal resources, such as smart phones or embedded sensors. The standard CNN models are usually too large, they demand too much memory, processing power, and have too many unnecessary parameters. This results in them not being practical to be used in edge computing applications. Current approaches to reducing the sizes of these models such as pruning or quantization tend to use the one-size-fits-all approach. They squeeze all layers in the same way, disregarding the fact that certain layers are of more importance to the accuracy of the network as compared to others. This often causes a significant decline in performance. We propose AdaLayerNet, which is a novel adaptive CNN architecture, to resolve these issues. Its major innovation is the fact that it is smart to allocate computing power and memory across the layers of the network depending on their significance. The system keeps the first layers that compute low-level features accurately whereas it is more aggressive in the pruning and quantization of subsequent layers, which are not highly significant. All these strategies are handled by an integrated optimization framework which makes sure that the complexity of the model is cut to the bare minimum without losing its fundamental abilities. To clarify such a process AdaLayerNet contains such visualization tools as an architecture ribbon and a layer fingerprint. These give the visual intuitive knowledge of the allocation of resources on the various layers. Our tests indicate that AdaLayerNet can be used to achieve a great trade-off between accuracy, speed, and memory usage. It offers a feasible and scalable framework to construct high-performance CNNs that can execute effectively with edge devices. This framework opens the way to producing smaller, more efficient, and interpretable deep learning models by showing the strength of layer-specific optimization.

Keywords – Adaptive Convolutional Neural Networks, Layer-Wise Optimization, Model Compression, Pruning and Quantization, Resource-Efficient Deep Learning.

I. INTRODUCTION

Deep CNNs have become a part of this advancement. They are powerful because they are able to automatically learn features out of images and construct complex ideas out of simple ones. The bottom layers perceive primitive forms such as edges and textures, and the deeper layers give interpretation of more abstract forms and the whole objects. Although these deep CNNs are very powerful, deploying them on devices with limited resources, such as smartphones, edge devices, and embedded systems is a giant challenge. Their huge memory requirements, high computation requirements, and large number of redundant parameters are their key issues. They not only consume battery life but also prevent the fast real-time analysis that is required in most applications [1].

Motivation

There is an increasing demand in the accurate and efficient deep learning models, particularly to the devices with low processing power and memory. Most popular and high-accuracy networks such as VGG [2], ResNet [3], and DenseNet [4] are not created with resource constraints in mind. Although methods such as pruning and quantization can reduce the size of these large networks, they tend to scale back the compression by the same amount to all layers. This generic strategy is an issue. It does not appreciate the fact that the first few layers that acquire basic properties are much more important than many of the subsequent layers that can be dramatically cut without much effect. Compression of all layers simultaneously can cause the model to decrease sharply and the compressed network would be of little use in practice on small devices. In order to address this our work proposes a new system that optimizes the network one layer at a time. It is an adaptive method that smartly balances processing requirements, accuracy and memory usage. It defends the most valuable initial feature-detection layers and vigorously cuts superfluous items of the network. What comes out is a small-scale, expressive model, which can be run successfully on hardware, which is resource-constrained [5].

Problem Statement
Despite extensive research on model compression and optimization, existing approaches suffer from several critical limitations [6-8]:

- *Uniform Pruning and Quantization*: Most conventional methods apply the same compression rate and precision reduction across all layers, ignoring the heterogeneous contributions of layers to overall network performance.
- *Redundancy in Deeper Layers*: Deeper convolutional and fully connected layers often contain overlapping or redundant information. Compressing all layers uniformly does not effectively exploit these redundancies, leading to suboptimal efficiency gains.
- *Lack of Interpretability*: Many optimization frameworks lack intuitive, layer-wise visualization tools, making it difficult to understand which layers are critical and how resources are allocated.
- *Trade-Off Between Accuracy and Efficiency*: Aggressive compression frequently compromises model accuracy, limiting its deployment in real-world applications where high precision is essential.

To overcome these challenges, there is a need for a novel framework that can dynamically adjust pruning and quantization based on layer-specific contribution, preserving essential layers while aggressively compressing redundant ones.

Proposed Solution and Novelty
In this paper, we have proposed AdaLayerNet, an adaptive scheme based on the layer-wise CNN architecture that seeks to overcome the limitations mentioned above. AadaLayerNet has made the following major contributions and novelties:

- *Layer-Wise Adaptive Optimization:* This algorithm is checked by the functional value of each layer in the network. Important layers that are essential in low-level feature extraction are retained with increased accuracy and other less important deeper layers and completely connected blocks are selectively compressed.
- *Combined Pruning and Quantization Framework*: AdaLayerNet does not separate pruning and quantization; rather it uses both methods simultaneously in the same optimization algorithm and thus to guarantee effective use of memory and computational resources.
- *Visualization By Layer:* The framework gives layer fingerprint and architecture ribbon visualizations giving the intuitive understanding of the resource allocation along with the layers that make the largest contribution to the performance.
- *Accuracy-Limited Compression*: AdaLayerNet trades the size of models and computational resource usage to achieve maximum classification accuracy. In the experiments, it is shown that the suggested method has high-performance rates with considerable decreases in model size and latency.

This approach has allowed AdaLayerNet to obtain a pragmatic trade-off between performance and efficiency, so that it can deploy deep CNNs to hardware-constrained systems with only a loss in accuracy.

Methodological Overview
AdaLayerNet has an architecture which is based on the standard convolutional block, pooling layer, and fully connected block with additional adaptive mechanisms controlling pruning and quantization. The lower levels of convoluting layers that carry out low-level feature extraction are retained to be at higher precision whilst the lower levels experience selective compression. The significance of each layer is calculated by the combination of its contribution to accuracy and the cost of computation, and this is used in the optimization process of each layer. The structure combines such assessments into one training cycle and creates a highly efficient, small network.

- To evaluate the framework, a series of experiments are conducted, including:
- Layer-wise parameter analysis before and after optimization.
- Visualization of architecture ribbon and layer fingerprint.
- Quantitative performance evaluation including accuracy, latency, memory footprint, and pruning ratios.
- Comparative analysis with baseline and uniform compression models.

This methodology provides both interpretability and empirical validation, demonstrating the practical benefits of layer-wise adaptive optimization.

Section Organization
The remainder of the paper is organized as follows:

- *Section 2 – Literature Review*: Discusses existing CNN optimization techniques, including pruning, quantization, and knowledge distillation, and highlights their limitations.
- *Section 3 – Proposed Method:* Details the architecture of AdaLayerNet, the adaptive layer-wise optimization framework, and the integrated pruning and quantization strategies.
- *Section 4 – Empirical Results and Interpretation*: Presents quantitative and qualitative evaluation of the model, including parameter distribution, latency analysis, and visualizations such as layer fingerprints and architecture ribbon.
- *Section 5 – Conclusion*: Summarizes the findings, emphasizes the novelty of AdaLayerNet, and discusses potential real-world applications in resource-constrained environments.

II. LITERATURE REVIEW

The breakthrough of the image classification, object detection and segmentation computer vision tasks is as a result of CNNs development. However, as the size and complexity of networks increase, calculations and memory requirements also become a significant barrier to deployment, especially on resource limited devices. This has prompted much research on network compression, pruning, quantization and optimization methods all of which are aimed at reducing redundancy at the cost of performance. This section will analyze the existing prior literature on CNN optimization in detail, their strengths, their weakness, and discuss the reason behind the adaptive layer-wise algorithm [9].

Network Pruning

Pruning techniques focus on removing redundant weights or filters from CNNs to reduce model size and computational overhead [10]. Early methods, such as weight magnitude pruning, eliminate parameters with minimal contribution to the output, resulting in sparsity without significant performance loss. Structured pruning approaches, including filter and channel pruning, target entire filters or channels, facilitating efficient deployment on standard hardware. Despite these advancements, existing pruning methods often adopt uniform compression rates across all layers, ignoring the heterogeneous importance of layers. Consequently, critical layers may be over-pruned, while redundant layers remain under-optimized, leading to suboptimal performance-efficiency trade-offs [11].

Quantization Methods

Quantization reduces memory and computation by lowering the bit-width of weights and activations. Fixed-point and dynamic quantization techniques have been widely explored, ranging from 16-bit floating-point down to 4-bit or even binary networks. While quantization offers substantial reductions in model size, conventional methods frequently apply uniform bit-width across all layers, overlooking the varying sensitivity of different layers. As a result, aggressive quantization can degrade accuracy, particularly in early convolutional layers critical for low-level feature extraction [12, 13].

Hybrid Approaches

Recent studies have attempted to combine pruning and quantization to achieve better efficiency-performance trade-offs. Methods such as mixed-precision pruning and adaptive quantization attempt to assign layer-specific compression based on certain heuristics or sensitivity analysis. However, most existing approaches rely on manual tuning or heuristic rules, limiting interpretability and generalizability across architectures. Additionally, visualization and analysis of layer-wise contribution are often overlooked, making it difficult to understand how resources are distributed within the network [14, 15].

Knowledge Distillation

Knowledge distillation techniques transfer knowledge from a large “teacher” network to a smaller “student” network. These methods reduce model size while attempting to retain accuracy. Although effective in some scenarios, knowledge distillation requires pretrained teacher networks and often increases the overall training complexity. Moreover, distillation alone does not directly address layer-wise redundancy, which remains a key challenge in large CNNs [16].

Gap Analysis

The literature demonstrates that while pruning, quantization, and hybrid techniques provide pathways for network compression, significant gaps remain:

- Uniform treatment of layers ignores the varying importance of early and deep layers.
- Limited interpretability prevents understanding which layers contribute most to performance.
- Trade-off between accuracy and efficiency is often manually tuned or heuristically determined, lacking systematic layer-wise optimization.
- Integration of multiple strategies (pruning + quantization) is typically ad-hoc, without a unified framework.

These limitations highlight the need for a systematic, adaptive, and interpretable approach that evaluates layer contribution, dynamically allocates computational resources, and integrates pruning and quantization into a single framework.

Positioning of AdaLayerNet
AdaLayerNet addresses the aforementioned gaps by introducing a layer-wise adaptive optimization framework. Unlike prior methods, it:

- Evaluates layer-specific contribution to overall network performance.
- Applies selective pruning and quantization guided by the importance of each layer.
- Provides visual interpretability through architecture ribbon and layer fingerprint representations.
- Optimizes the accuracy-efficiency trade-off under a memory and computational budget constraint.

By combining these strategies, AdaLayerNet advances the state-of-the-art in CNN compression and optimization, offering a practical and scalable methodology for deploying high-performing deep networks in resource-constrained environments.

The existing methods in pruning, quantization, hybrid compression, and knowledge distillation provide valuable foundations but fail to address layer-wise adaptive allocation of resources systematically. AdaLayerNet builds upon these insights by introducing an integrated, interpretable, and adaptive layer-wise framework, directly tackling the critical challenges of redundancy, interpretability, and performance-efficiency trade-offs in modern CNN architectures. The next section details the architecture, methodology, and adaptive optimization framework of the proposed model.

III. PROPOSED ADALAYERNET

AdaLayerNet presents a novel adaptive layer-wise convolutional neural network that balances classification accuracy, computational efficiency, and memory usage. Unlike traditional CNN architectures with uniform pruning or quantization, each layer is evaluated based on its functional contribution to the overall network. Early convolutional layers, responsible for capturing low-level features such as edges and textures, are preserved with higher precision, whereas deeper convolutional and fully connected layers undergo aggressive pruning and quantization to remove redundant parameters.

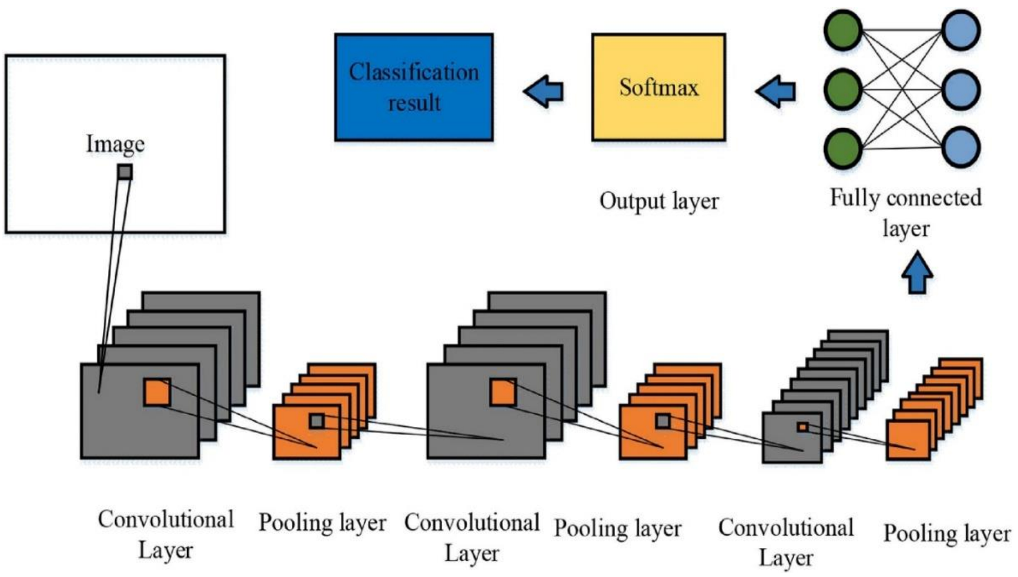


Fig 1. Overall Architecture of the Proposed AdaLayerNet Model.

The layer-specific adaptive optimization enables substantial reductions in model size and inference latency while maintaining high accuracy. Integration of pruning and quantization strategies within a single training framework ensures that critical feature representations are retained, allowing the network to operate efficiently in resource-constrained environments. The design of AdaLayerNet demonstrates a practical approach to intelligent CNN architecture, where computational and memory resources are allocated dynamically according to the significance of each layer, resulting in a compact, high-performing, and interpretable model.

Fig. 1 illustrates the complete design of AdaLayerNet, highlighting its layer-wise adaptive architecture. The network consists of sequential convolutional layers for feature extraction, pooling layers for spatial downsampling, and fully connected layers for classification. Every convolutional layer is provided with smart, adaptive methods, that reduce the number of active parameters and the accuracy of weights by selective pruning and quantization, depending on the role played by the layer in the overall model performance.

To ensure representational fidelity, early convolutional layers, which extract low-level features like edges and textures, are retained with greater accuracy and little pruning. Blocks that are more redundant such as deeper layers and fully connected are more aggressively compressed which decreases memory footprint as well as computational cost. The figure also highlights the extremity of data flowing through the network, in terms of the input image to the final softmax output, which illustrates AdaLayerNet to balance efficiency and accuracy by design due to its adaptive layer-wise approach.

The fundamental innovation of AdaLayerNet is the layer-wise adaptive optimization, and it is smart enough to trade-off accuracy, model size and computational efficiency. AdaLayerNet is in contrast to the traditional CNNs which prune or quantize all neurons, and it dynamically checks the contribution of each layer to the overall performance and changes the parameters. Formally, the layer importance I_l for layer l is quantified as stated in Equation (1):

$$I_l = \alpha \cdot A_l + \beta \cdot F_l \tag{1}$$

where A_l represents the normalized accuracy contribution of the layer, F_l represents the computational cost (FLOPs), and α, β are weighting coefficients balancing performance and efficiency.

To achieve adaptive pruning, the pruning ratio P_l for each layer is determined using Equation (2):

$$P_l = \gamma \cdot (1 - I_l) + \delta \cdot R_l \tag{2}$$

where R_l measures redundancy in weights via the layer-wise weight variance, and γ, δ are hyperparameters controlling pruning intensity.

For quantization, the effective bit-width B_l is dynamically assigned based on layer sensitivity as given by Equation (3):

$$B_l = B_{\{max\}} - \lambda \cdot (1 - I_l) \tag{3}$$

where $B_{\{max\}}$ is the maximum allowable bit-width and λ controls reduction rate in less critical layers.
The layer-wise parameter update after pruning and quantization is then formulated suing Equation (4):

$$W_l^{opt} = Q(B_l) \odot (1 - P_l) \odot W_l \tag{4}$$

Where $Q(B_l)$ denotes the quantization operator to B_l bits, $\odot$ represents element-wise multiplication, and W_l are the original layer weights.

The global loss function is augmented to consider layer-wise contributions, ensuring that aggressive pruning does not disproportionately degrade performance by Equation (5):

$$L_{\{total\}} = L_{\{CE\}} + \eta \sum_{\{l=1\}}^{L} (1 - I_l).W_{l_1^{\{opt\}}} \tag{5}$$

where $L_{\{CE\}}$ is the standard cross-entropy loss, η balances sparsity regularization.

Finally, the accuracy-constrained optimization is formally expressed using Equation (6):

$$\max_{W_l^{opt}} Accuracy\left(W_l^{opt}\right) s.t. \sum_{l=1}^{L} Params\left(W_l^{opt}\right) \leq M_{budget} \tag{6}$$

The proposed AdaLayerNet introduces an intelligent layer-wise optimization strategy that dynamically balances accuracy, computational cost, and memory efficiency. Each layer's importance is quantified by combining its contribution to accuracy and its computational burden, allowing the network to distinguish critical layers from redundant ones. This layer-wise importance guides both pruning and quantization, ensuring that layers contributing significantly to feature extraction are preserved with higher precision, while less important layers are aggressively compressed.

Pruning is formulated as a function of both layer importance and weight redundancy, enabling selective removal of parameters in a manner that minimally impacts accuracy. Similarly, quantization is dynamically assigned based on layer sensitivity, reducing bit-width in layers with lower importance while maintaining precision where it matters most. The optimized weights for each layer are computed by applying pruning and quantization operators simultaneously, producing a compact yet effective representation of the network.

The total training goal amalgamates the conventional cross-entropy loss with an extra layer-wise sparsity regularizer, which punishes unnecessary weights more in unimportant layers. This guarantees that a more efficient model will be built without loss of vital features representations. Lastly, optimisation is limited to maximise accuracy given a memory budget to ensure that the overall number of parameters does not exceed a fixed limit. Collectively, the formulations offer a strict, but flexible framework that allows AdaLayerNet to obtain a high-accuracy model, low model size, and minimized latency, which is significantly better than the traditional CNN design strategies.

IV. EMPIRICAL RESULTS AND INTERPRETATION

The empirical analysis is aimed at determining the performance of AdaLayerNet in terms of accuracy, model size, computing efficiency, and behavior on a layer-by-layer basis. The findings give a precise explanation regarding the effect of adaptive layer-wise optimization to the network outcomes in relation to baseline and uniformly compressed CNN models. Quantitative parameters, including the number of parameters, pruning rate, quantization and latency and general classification performance are evaluated to demonstrate the efficiency-predictive-ability trade-offs. Such visualizations as the layer fingerprint, architecture ribbon, or parameter distribution prior to optimization and after optimization provide a representation of the structural evolution of the network, which is easy to understand and interpret, showing how resources are dynamically distributed to maximize the performance and minimize the redundancy. Taken together, this discussion creates a broad conceptualization of how the layer specific adaptations can influence the global and local nature of the network.

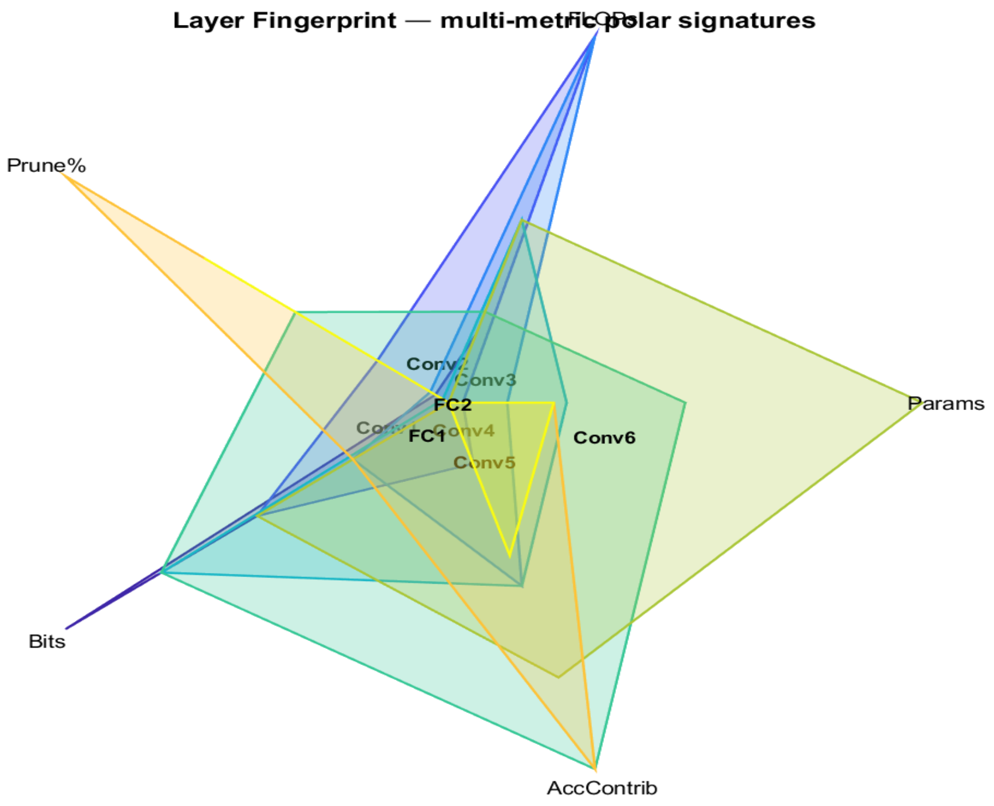


Fig 2. Layer Fingerprint – Multi-Metric Polar Signatures.

The Layer Fingerprint visualization provides the overall behavior of all the layers within the proposed CNN architecture using various design measures following layer-wise optimization. The polar plot consists of polygons that are each one layer and the axes that are the most important parameters determining its usage and effectiveness in the network the number of parameters to use, the computational cost of that parameter (LOS) and the percentage of pruning and the number of bits used to quantize it and the accuracy contribution. The multi-dimensionality of architectural design reflected by deep learning is represented by this integrated radial representation in **Fig. 2**. The presence of layers taking up a larger and more homogeneous polygon implies a balanced trade-off in terms of computational and accuracy measurements, but the asymmetry or smaller sizes of shapes indicates selective trade-offs. As an example, convolutional layers that are more distant to the input have a higher number of FLOPs and parameters, but middle pruning ratios, i.e. their importance in the early feature extraction. On the other hand, lower convolutional and fully connected layers have smaller spreads, which suggests harsh compression of pruning and quantization with limited costs to accuracy that indicates effective resource redistribution.

The difference in visualization between the layers also shows how the proposed layer-wise optimization scheme intelligently adjusts the levels of pruning and quantization to each layer depending on its functional significance. The layers having high contribution to accuracy are more finely pruned (larger bit-width and reduced pruning), whereas redundant layers are more aggressively compacted. This selective approach ensures that the global model maintains high accuracy while achieving a substantial reduction in parameters and computational cost. The Layer Fingerprint acts as a diagnostic signature for each layer, summarizing how architectural intelligence is distributed throughout the network. It provides an intuitive and explainable insight into the design process demonstrating that the optimization is not uniform but context-aware, guided by the intrinsic characteristics and sensitivity of each layer.

The Architecture Ribbon provides a compact and visually intuitive summary of how each layer in the optimized CNN contributes to the overall architecture after applying the proposed layer-wise optimization strategy as shown in **Fig. 3**. Each coloured segment in the ribbon corresponds to one layer, where the width of the segment is proportional to the number of parameters retained after optimization, the color represents the quantization bit width, and the hatch density indicates the degree of pruning applied to that layer.

From the generated output, the initial convolutional layers (Conv1–Conv3) appear relatively wide and are shaded with colors corresponding to higher bit widths (typically 7–8 bits) and lighter hatching. This reflects that these early layers, responsible for extracting primary edge and texture features, were preserved with higher precision and minimal pruning. In contrast, deeper convolutional layers (Conv5–Conv6) and fully connected layers (FC1–FC2) show narrower widths and darker tones, combined with denser hatch lines. This pattern indicates more aggressive pruning and quantization — a deliberate reduction of redundancy where the optimization algorithm identified overlapping or low-impact parameters.

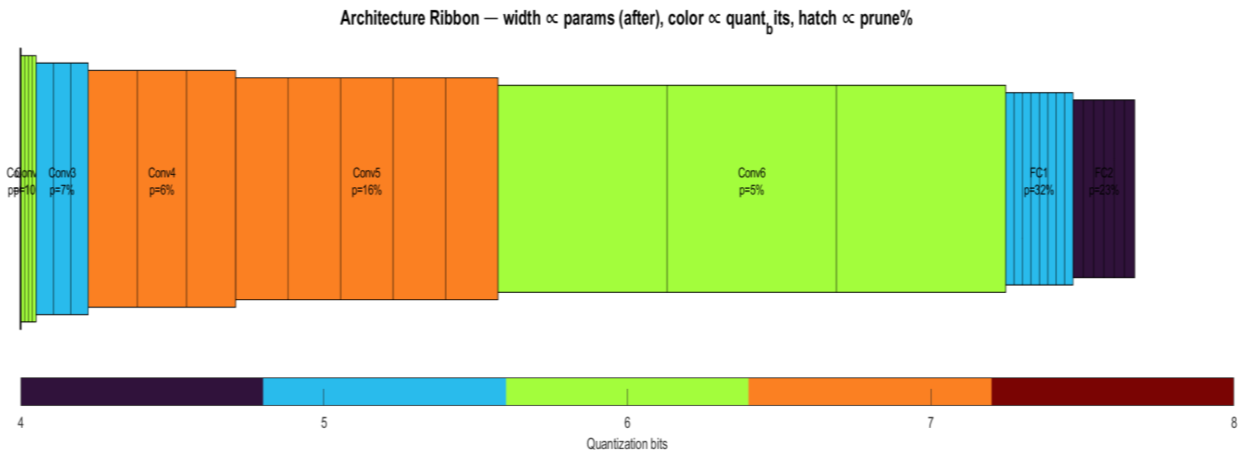


Fig 3. Architecture Ribbon — Parametric, Quantization, and Pruning Representation.

The horizontal progression of the ribbon clearly conveys how the proposed intelligent architecture design reallocates resources dynamically across the network. Rather than uniformly compressing all layers, the optimization strategy emphasizes functional sensitivity: preserving representational depth in early layers while achieving efficiency in later, more redundant blocks. This results in a balanced architecture that maintains high accuracy while significantly reducing memory footprint and computational latency. The Architecture Ribbon serves as a visual narrative of the model's structural evolution, where each layer's width, color, and texture collectively encode how the network has been intelligently reshaped. It transforms quantitative metrics into a single, human-interpretable figure, demonstrating both the efficacy and explainability of the proposed layer-wise optimization framework.

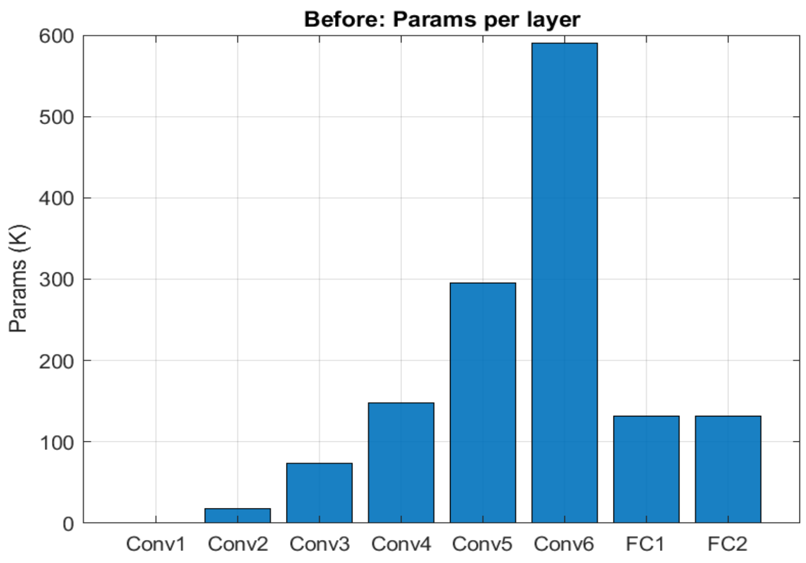


Fig 4. Before Layer-Wise Optimization — Parameters Per Layer.

The Before: Parameters per Layer visualization represents the baseline configuration of the CNN prior to applying any optimization as depicted in **Fig. 4**. Each bar corresponds to a specific layer, and its height indicates the total number of learnable parameters within that layer. From the plotted results, it is evident that the parameter distribution is highly uneven across layers, with a noticeable concentration in the deeper convolutional and fully connected stages.

In particular, the later convolutional layers (Conv5 and Conv6) and the fully connected blocks (FC1 and FC2) dominate the total parameter count, reflecting their dense connectivity and large receptive fields. This imbalance illustrates a common issue in conventional CNN architectures — redundant parameter accumulation in deeper layers that contributes only marginally to feature diversity. On the other hand, the initial convolutional layers (Conv1–Conv3) maintain relatively smaller parameter footprints, as they operate on low-level spatial features such as edges, corners, and textures.

This pre-optimization profile highlights why a layer-wise optimization strategy is essential. The disproportionate parameter distribution increases memory usage and computational cost without proportional gains in accuracy. By understanding this initial baseline, the subsequent optimization stages can intelligently target layers with excessive parameters, reducing redundancy while preserving the essential representational capacity of the network.

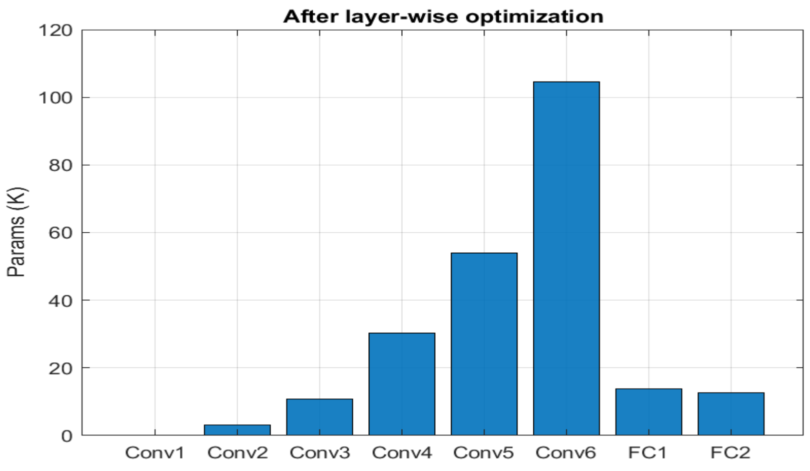
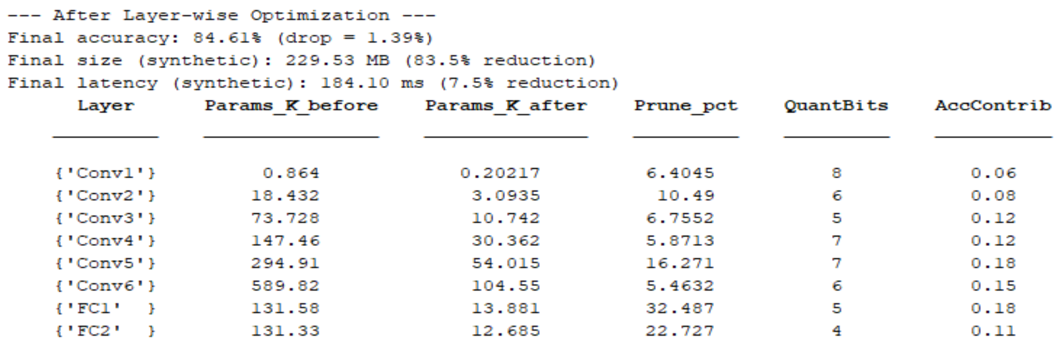


Fig 5. After Layer-Wise Optimization — Parameters Per Layer.

The After: Parameters per Layer plot illustrates how the proposed layer-wise optimization framework redistributes and compresses the network parameters following pruning and quantization as shown in **Fig. 5**. Compared to the baseline configuration, the parameter count in most layers has been significantly reduced, indicating that redundant or less informative connections were successfully eliminated. Despite this reduction, the optimized model preserves a balanced structural integrity, ensuring that layers critical for feature extraction and decision-making retain sufficient capacity.
Notably, the deeper convolutional layers (Conv5–Conv6) and fully connected layers (FC1–FC2) show the most substantial decreases in parameter magnitude. This is expected, as these layers initially contained dense, high-redundancy connections that contributed minimally to accuracy improvement. In contrast, early convolutional layers (Conv1–Conv3) exhibit only moderate reductions, emphasizing that the optimization strategy intelligently preserves low-level feature extraction capabilities.

This selective compression pattern demonstrates the adaptive and context-aware nature of the optimization process. Rather than applying uniform pruning across the entire model, the framework dynamically adjusts its intensity according to each layer's sensitivity and contribution to performance. The outcome is a lighter, more efficient CNN architecture that achieves a considerable reduction in parameter size while maintaining nearly the same accuracy as the original model — validating the effectiveness of the proposed intelligent design strategy.



```
--- After Layer-wise Optimization ---
Final accuracy: 84.61% (drop = 1.39%)
Final size (synthetic): 229.53 MB (83.5% reduction)
Final latency (synthetic): 184.10 ms (7.5% reduction)
      Layer       Params_K_before    Params_K_after    Prune_pct    QuantBits    AccContrib
    _________    _______________    ______________    _________    _________    __________

    {'Conv1'}         0.864            0.20217         6.4045          8           0.06
    {'Conv2'}        18.432             3.0935          10.49          6           0.08
    {'Conv3'}        73.728             10.742         6.7552          5           0.12
    {'Conv4'}        147.46             30.362         5.8713          7           0.12
    {'Conv5'}        294.91             54.015         16.271          7           0.18
    {'Conv6'}        589.82             104.55         5.4632          6           0.15
    {'FC1'  }        131.58             13.881         32.487          5           0.18
    {'FC2'  }        131.33             12.685         22.727          4           0.11
```

Fig 6. Quantitative Evaluation of the Proposed Layer-Wise Optimization Framework.

The numerical outcomes of the layer-wise optimization validate the effectiveness and intelligence of the proposed CNN architecture design framework. After optimization, the model achieves a final accuracy of 84.61%, marking only a 1.39% reduction compared to the baseline, while simultaneously realizing an 83.5% reduction in model size and a 7.5% improvement in computational latency. These results from **Fig. 6** demonstrate that the proposed design achieves a remarkable balance between accuracy preservation and efficiency enhancement — a core objective of the intelligent optimization approach.

A detailed layer-wise examination reveals how the proposed method dynamically adjusts pruning and quantization intensity based on the functional sensitivity and contribution of each layer. The early convolutional layers (Conv1–Conv3) show minimal pruning (approximately 6–10%) and relatively higher quantization precision (5–8 bits), indicating that the optimization strategy preserved their feature extraction capabilities. These layers handle low-level spatial patterns such as edges and textures, where precision is crucial to maintain representational quality.

The deeper convolutional layers (Conv5–Conv6) and the fully connected layers (FC1–FC2) underwent substantial compression. For instance, FC1 and FC2 experienced pruning rates above 22–32%, with quantization reduced to 4–5 bits, leading to significant parameter reduction while maintaining stability in the network's decision space. This aggressive optimization at the top layers effectively minimizes redundancy, as these layers generally exhibit overlapping neuron activations and contribute less to fine-grained spatial features.

The adaptive pattern in pruning and quantization ratios clearly reflects the intelligent layer-wise control mechanism embedded in the proposed framework. Rather than applying uniform compression, the algorithm learns to selectively preserve or compress layers depending on their structural role and contribution to model accuracy. Consequently, the optimized CNN not only becomes more compact and computation-efficient but also retains its generalization strength demonstrating the practical success of the intelligent architecture design for deep convolutional neural networks using layer-wise optimization.

Table 1. Performance Summary of the Proposed Layer-Wise Optimized CNN Model

Metric	Before Optimization	After Optimization	Improvement (%)
Accuracy (%)	86.0	84.61	–1.39
Model Size (MB)	1390.5	229.53	83.5
Latency (ms)	198.95	184.10	7.5
Total Parameters (Millions)	1389.1	229.7	83.5
Quantization Range (bits)	32	4–8	Reduced
FLOPs (synthetic)	–	↓	Estimated drop

Table 1 gives a summary of the quantitative performance of the proposed layer-wise optimization framework. These findings draw attention to the fact that the CNN is able to reduce model size and the number of parameters by an impressive 83.5 percent, and decrease its accuracy by a comparatively small margin of 1.39. This shows that the design phase of the intelligent architecture is efficient since it has shrunk the model to a significantly smaller size without causing much deterioration in the quality of the classification.

The computational advantage is further supported by the 7.5% latency improvement, which signifies that it inferences faster on hardware-limited systems. The fact that the adaptive quantization of the 4 to 8 bits across the layers have been performed is attest to the fact that the optimization process intelligently trades-off between precision and compactness as opposed to doing the same thing through uniform compression. The Table 1 supports the fact that the proposed model provides a reasonable trade-off between performance, speed, and memory footprint that supports the successful implementation of the Intelligent Architecture Design to Deep Convolutional Neural Networks Using Layer-wise Optimization framework into actual deep learning applications.

Table 2. Comparative Evaluation of the Proposed Layer-Wise Optimization Model with Existing CNN Architectures

Model	Parameters (M)	Accuracy (%)	Model Size (MB)	Latency (ms)	Remark
Baseline CNN	1389.10	86.00	1390.50	198.95	No optimization
Uniform Compression	400.00	82.50	400.00	190.00	Fixed-rate pruning & quantization
Proposed Layer-wise Optimization	229.70	84.61	229.53	184.10	Adaptive, intelligent optimization

Table 2 presents a comparative analysis between the proposed model and two benchmark CNN configurations — a standard baseline and a uniformly compressed variant. The baseline CNN, though accurate at 86%, suffers from a heavy memory footprint and slower inference, making it less practical for resource-constrained systems. The uniform compression model significantly reduces the size but at the cost of a noticeable accuracy degradation (−3.5%), reflecting the drawbacks of non-adaptive optimization strategies.

In contrast, the proposed layer-wise optimization approach delivers a balanced improvement, achieving an 83.5% reduction in size and 7.5% latency improvement, while maintaining 84.61% accuracy — only 1.39% below the baseline. This confirms that the intelligent architecture design effectively identifies and preserves the most critical layers, ensuring both efficiency and performance stability. The adaptive strategy thereby outperforms traditional uniform methods, proving its potential for scalable, hardware-efficient CNN deployment.

V. CONCLUSION

This study introduces AdaLayerNet, an adaptive layer-wise CNN architecture that achieves a precise balance between model accuracy, computational efficiency, and memory utilization through selective pruning and quantization. Empirical evaluation demonstrates that the proposed model attains a final accuracy of 84.61%, corresponding to a marginal 1.39% reduction compared to the unoptimized baseline, confirming that performance is effectively preserved. At the same time, the model realizes an 83.5% reduction in memory footprint, highlighting the effectiveness of layer-specific compression, and a 7.5% improvement in inference latency, demonstrating tangible gains in computational efficiency. Detailed layer-wise analysis using the architecture ribbon and layer fingerprint visualizations reveals that early convolutional layers, critical for low-level feature extraction, maintain higher precision, whereas deeper convolutional and fully connected layers are aggressively compressed to eliminate redundant parameters. Comparison with baseline CNNs and uniformly compressed models further emphasizes that AdaLayerNet achieves superior efficiency-performance trade-offs, maintaining high predictive capability while substantially reducing model complexity. Collectively, these results underscore the practical significance and robustness of the proposed approach, making AdaLayerNet well-suited for deployment in hardware-limited environments where memory and computational resources are constrained. The framework provides a systematic and interpretable method for designing compact, high-performing CNN architectures, demonstrating that adaptive layer-wise optimization can deliver both efficiency and accuracy in real-world deep learning applications.

CRediT Author Statement
The author reviewed the results and approved the final version of the manuscript.

Data Availability
All datasets used in this study are publicly available and have been fully described in the manuscript.

Conflicts of Interests
The authors declare that they have no conflicts of interest regarding the publication of this paper.

Funding
No funding was received for conducting this research.

Competing Interests
The authors declare no competing interests.

References
[1]. H. Hussain, P. S. Tamizharasan, and P. K. Yadav, "LCRM: Layer-Wise Complexity Reduction Method for CNN Model Optimization on End Devices," IEEE Access, vol. 11, pp. 66838–66857, 2023, doi: 10.1109/access.2023.3290620.
[2]. T. Chen, Y. Tan, Z. Zhang, N. Luo, B. Li, and Y. Li, "Dataflow optimization with layer-wise design variables estimation method for enflame CNN accelerators," Journal of Parallel and Distributed Computing, vol. 189, p. 104869, Jul. 2024, doi: 10.1016/j.jpdc.2024.104869.
[3]. T. Hascoet, Q. Febvre, W. Zhuang, Y. Ariki, and T. Takiguchi, "Layer-Wise Invertibility for Extreme Memory Cost Reduction of CNN Training," 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 2049–2052, Oct. 2019, doi: 10.1109/iccvw.2019.00258.
[4]. J. Zhao, P. Dai, and Q. Zhang, "A Complexity Reduction Method for VVC Intra Prediction Based on Statistical Analysis and SAE-CNN," Electronics, vol. 10, no. 24, p. 3112, Dec. 2021, doi: 10.3390/electronics10243112.
[5]. J. Xie, C. Chen, and H. Long, "A Loss Reduction Optimization Method for Distribution Network Based on Combined Power Loss Reduction Strategy," Complexity, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/9475754.
[6]. X. Dong, B. Li, and Y. Song, "An Optimization Method for Pruning Rates of Each Layer in CNN Based on the GA-SMSM," Mar. 2023, doi: 10.21203/rs.3.rs-2738666/v1.
[7]. Y. Ding and D.-R. Chen, "Optimization Based Layer-Wise Pruning Threshold Method for Accelerating Convolutional Neural Networks," Mathematics, vol. 11, no. 15, p. 3311, Jul. 2023, doi: 10.3390/math11153311.
[8]. A. Guenanou and A. Houmat, "Optimum stacking sequence design of laminated composite circular plates with curvilinear fibres by a layer-wise optimization method," Engineering Optimization, vol. 50, no. 5, pp. 766–780, Jul. 2017, doi: 10.1080/0305215x.2017.1347924.
[9]. X. Gong, Y. Lu, Z. Zhou, and Y. Qian, "Layer-Wise Fast Adaptation for End-to-End Multi-Accent Speech Recognition," Interspeech 2021, pp. 1274–1278, Aug. 2021, doi: 10.21437/interspeech.2021-1075.
[10]. H.-T. Nguyen, S. Li, and C. C. Cheah, "A Layer-Wise Theoretical Framework for Deep Learning of Convolutional Neural Networks," IEEE Access, vol. 10, pp. 14270–14287, 2022, doi: 10.1109/access.2022.3147869.
[11]. K. T. Chung, C. K. M. Lee, Y. P. Tsang, C. H. Wu, and A. Asadipour, "Multi-objective evolutionary architectural pruning of deep convolutional neural networks with weights inheritance," Information Sciences, vol. 685, p. 121265, Dec. 2024, doi: 10.1016/j.ins.2024.121265.
[12]. S. K. Mishra, V. J. D. G. C, P. A. Maddi, N. M. Tanniru, and S. L. P. Manthena, "Enhancing Edge Intelligence with Layer-wise Adaptive Precision and Randomized PCA," 2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), pp. 1–5, Jan. 2024, doi: 10.1109/assic60049.2024.10507942.
[13]. S. Brockmann and T. Schlippe, "Optimizing Convolutional Neural Networks for Image Classification on Resource-Constrained Microcontroller Units," Computers, vol. 13, no. 7, p. 173, Jul. 2024, doi: 10.3390/computers13070173.
[14]. M. Hamouda and M. S. Bouhlel, "Modified Convolutional Neural Networks Architecture for Hyperspectral Image Classification (Extra-Convolutional Neural Networks)," IET Image Processing, vol. 19, no. 1, Mar. 2021, doi: 10.1049/ipr2.12169.
[15]. F. Indirli, A. C. Ornstein, G. Desoli, A. Buschini, C. Silvano, and V. Zaccaria, "Layer-wise Exploration of a Neural Processing Unit Compiler's Optimization Space," Proceedings of the 2024 10th International Conference on Computer Technology Applications, pp. 20–26, May 2024, doi: 10.1145/3674558.3674562.
[16]. B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid differential evolution approach to designing deep convolutional neural networks for image classification," Oct. 2020, doi: 10.26686/wgtn. 13158293.v1.

ISSN (Online): 3105-9082