Journal of Applied Automation Technologies

Research Paper

Lightweight PhiNet-Based Neural Network for Image Classification on Edge Devices

Aleksandra Danuta Król[1, *]

[1] Maria Curie-Sklodowska University, Faculty of Mathematics, Physics and Computer Science, 20-031 Lublin, Poland
*Corresponding author: aleksandra.dk@mail.umcs.pl

Abstract. Many AI applications still require substantial computation and memory, making edge computing unsuitable for resource-limited devices. Therefore, this paper will design a relatively lightweight image classification framework based on the enhanced PhiNet architecture. Dynamically select filters and perform hierarchical quantization to achieve the best balance between accuracy and efficiency. In order to evaluate the framework, the benchmark datasets ImageNet-1K and CIFAR-100 were selected using the aforementioned standard experimental procedures. The above results indicate that the model based on PhiNet achieved a Top-1 accuracy of 77.0% on ImageNet-1K and 85.1% on CIFAR-100, surpassing the corresponding Top-1 accuracies of MobileNetV3 and EfficientNet-Lite. On the NVIDIA Jetson Xavier NX platform, the average inference latency per image is 23.1 milliseconds, with a throughput of 42.4 frames per second. In addition, the average power consumption is 2.7 watts. When batch processing up to 32 photos, the memory usage remains below 95 MB. In addition, many tests in various real-world environments have shown good consistency and robustness. These environments include recognizing pedestrians in low light and recognizing license plates in motion. Increase hardware-aware optimizations so that the system can more easily adapt to various hardware. Finally, this paper demonstrates that the PhiNet-based solution achieves an ideal balance in terms of accuracy, efficiency, and robustness, making it suitable for practical applications in edge intelligence.

Keywords: *Edge Computing, Image Classification, Lightweight Neural Network, Model Quantization, Hardware-Aware Optimization, Resource Efficiency, Dynamic Filter Selection, Deep Learning*

Received on 03 November 2025, Accepted on 01 March 2026, Published on 30 March 2026

Introduction

Due to the rapid development of deep learning and artificial intelligence (AI), the accuracy of computer vision in image classification tasks has significantly improved [1]. Deep Convolutional Neural Networks (CNNs) have many applications in real life, such as autonomous vehicles, intelligent medical diagnosis systems, and smart surveillance systems [2]. The superior capabilities of deep neural networks in feature extraction and hierarchical representation [3] explain the aforementioned results. However, high-performance convolutional neural networks (CNNs), such as ResNet and DenseNet, are usually very large in terms of computation and memory, making them unsuitable for deployment on resource-constrained devices [4]. With the global spread of the Internet of Things (IoT), the demand for efficient on-device inference is increasing. Therefore, lightweight edge computing neural networks have recently received widespread attention [5]. In order to improve reliability, privacy, and latency, edge AI aims to process data locally on embedded hardware [6]. However, there are many issues in the design and implementation of the model [7]. We need new methods to achieve high computational efficiency and good discrimination ability in the model simultaneously [8].

Current methods still have some shortcomings when applied to various edge hardware, although there is increasing attention on the design of lightweight neural networks. Structured model pruning usually increases accuracy, even tho it reduces the number of parameters [9]. By reducing the bit-width of network weights, quantization methods can improve inference speed, but they may also reduce generalization ability and

robustness [10]. Although recent compact architectures like MobileNet and ShuffleNet have reduced computational costs through depthwise separable convolutions, they still cannot address all the issues caused by various hardware limitations and changes in workload conditions [11]. Although adaptive neural architecture search and channel pruning strategies have been used to create scalable models, they often significantly increase the search or training costs [12]. Due to the differences in computational capabilities and memory organization among various edge devices (such as microcontrollers, NPUs, and embedded GPUs), different models are needed to adapt to environmental choices [13]. Due to the lack of effective dynamic computation and hierarchical optimization mechanisms, these methods lack scalability and practicality [14]. Therefore, end-to-end trainable, hardware-aware, and dynamically adaptive systems require lightweight frameworks [15]. Accurate, real-time edge image classification remains an unsolved challenge [16].

To address the aforementioned issues, a novel lightweight image classification framework based on the improved PhiNet architecture is proposed here. It uses hierarchical quantization and dynamic filter selection to improve the computational efficiency of edge devices. The speed and memory consumption of on-device inference will be reduced, but its accuracy will remain unchanged under resource constraints. To demonstrate that our method has excellent performance and generalization capabilities, we conducted extensive experiments on popular datasets and real-world edge devices. The following is the organization of the other sections of this paper: In Section 2, the current progress in the design and optimization of lightweight neural networks. In Section 3, the new PhiNet architecture is introduced, along with the main algorithm modules and hardware-oriented improvements. In Section 4, detailed descriptions of benchmark experiments, comparative results, ablation studies, and practical deployment are provided. In Section 5, the conclusions of the paper and the directions for future research on edge artificial intelligence are listed.

Related Work in Lightweight Neural Networks

Evolution of Deep Classification Networks

Convolutional Neural Networks (CNN) are a deep learning technique that is very effective in many large-scale image classification problems. Due to the excellent performance of AlexNet, researchers began to study multi-layer deep neural networks for feature extraction [17]. In addition, the VGGNet model has been shown to improve classification accuracy by increasing the network's depth through the addition of multiple small convolutional filters [18]. However, as the network deepens, the number of parameters and floating-point operations increases exponentially. This leads to hardware and power issues outside of data centers.

ResNet is a new technology that can address scalability issues by using residual learning to build deeper networks without encountering the vanishing gradient problem. This new technology quickly became the foundation for many visual applications [19]. In addition, DenseNet is a densely connected model. It improves feature reuse and deep supervision by adding inter-layer connections, thereby producing compact yet highly discriminative feature representations [20]. In addition, methods have been attempted to enhance feature extraction capabilities through architectural diversity, such as the multi-branch structure and mixed receptive fields of Inception [21]. These models have made some progress, but most are still designed to run on powerful GPUs and high-performance computing clusters.

As computing shifts to the edge, the shortcomings of these traditional networks in embedded and mobile applications have become more apparent. As the demand for artificial intelligence in low-power, low-memory devices increase, new network designs are needed. Recently, most research has focused on the top-1 accuracy of a few closed datasets, neglecting system constraints and generalization issues [22]. Therefore, research on efficient models and lightweight neural network architectures for edge devices has also emerged [23].

Advances in Lightweight Models and Optimization

Lightweight neural networks currently have two directions. Pruning is one of the initial significant advancements; it systematically removes redundant or less informative filters and weights from pre-trained networks. This results in smaller, sparser models running faster [24]. By using quantization techniques to reduce the numerical precision of weights and activations, the memory burden is thereby reduced. In addition, when performing low-bit arithmetic operations on dedicated hardware accelerators, the accuracy is somewhat reduced [25].

Researchers have been designing efficiency-focused architectures from the beginning to understand the shortcomings of compressing existing large models. MobileNet introduced depthwise separable convolutions, which significantly reduced the number of computations and parameters, thereby improving the classification accuracy on mobile devices [26]. To improve lightweight inference on embedded systems, ShuffleNet introduced channel shuffling and grouped convolutions [27]. By introducing the "fire module," SqueezeNet improves the compactness of the model. Research shows that relatively simple structures can be as accurate as larger models with significantly fewer parameters [28].

Due to hardware diversity and the demand for real-time operations, neural architecture search (NAS) has recently been gradually used to automatically select suitable lightweight structures under practical constraints. MnasNet constructs models optimized for low-latency operations on certain mobile SoCs through platform-aware NAS, rather than merely considering theoretical issues [29]. This trend further drives hardware-aware technology, as they can optimize neural network modules, memory layouts, and data flows based on the operating conditions of current edge chips [30]. All these advancements show a clear trend: the design of lightweight neural networks is shifting from focusing on algorithm novelty and mathematical elegance to constructing a comprehensive system that includes accuracy, speed, memory efficiency, and hardware awareness. This makes lightweight neural networks a powerful and flexible solution suitable for edge AI environments.

PhiNet Architecture and Algorithms

Overall Structure of PhiNet

In the context of limited edge computing resources, PhiNet aims to meet the demand for computationally feasible, compact, and accurate models. The entire system adopts a modular design, making it suitable for various hardware platforms. Compared to many earlier convolutional neural networks, PhiNet uses finer and more modular block structures to reduce parameters and enhance representational capacity.

The network includes a lightweight classification head, a repeatedly stacked phi block, and an input trunk. Using a shallow input trunk, but setting a slightly wider first filter to capture low-level features while reducing redundancy. It can be shown that:

$$X_{stem} = \sigma\left(W_{stem} * X_{input} + b_{stem}\right) \quad \text{Eq.(1)}$$

where X_{input} is the raw image input, W_{stem} and b_{stem} are learnable parameters of the stem, * denotes convolution, and $\sigma(\cdot)$ is the chosen nonlinearity (ReLU in this instantiation).

Each Phi unit includes hierarchical quantization, dynamic filter selection, and depthwise separable convolution. By using depthwise separable convolution, standard convolution can be decomposed into depthwise convolution and pointwise convolution:

$$Y = \sigma\left(W_{pw} * (W_{dw} * X) + b\right) \quad \text{Eq.(2)}$$

where W_{dw} is the depthwise filter, W_{pw} the pointwise filter, and b the bias term. Compared to traditional convolution, the number of parameters and computational cost are greatly reduced.

In order to modularize and select paths based on runtime conditions, each module has been parameterized:

$$X^{(l+1)} = DFSelect\left(X^{(l)}, \mathcal{F}^{(l)}, \theta^{(l)}\right) \quad \text{Eq.(3)}$$

where $\mathrm{DFSelect}(\cdot)$ denotes the dynamic filter selection operator, $\mathcal{F}^{(l)}$ the filter bank at layer l, and $\theta^{(l)}$ an auxiliary mask governing active pathways. The block structure defaults to supporting depthwise separable convolutions, so the number of multiply-accumulate operations (MACs) per stage is reduced.

To facilitate the propagation and stability of residual features, a shortcut connection is added in each block:

$$X_{out} = X_{block} + X_{in} \quad \text{Eq.(4)}$$

Since X_{block} is the processed output of the phi block and X_{in} is the input of this block, gradient flow can be achieved, thereby solving the gradient vanishing problem in deep structures.

The classification head first uses global average pooling, and then uses a lightweight fully connected (FC) layer.

$$\hat{y} = Softmax\left(W_{fc} \cdot GAP(X_{final}) + b_{fc}\right) \quad \text{Eq.(5)}$$

where X_{final} is the final block's output, W_{fc} and b_{fc} are trainable parameters, and GAP denotes global average pooling.

The large-scale data flow and cooperation among dynamic and static modules in PhiNet are shown in Figure 1.

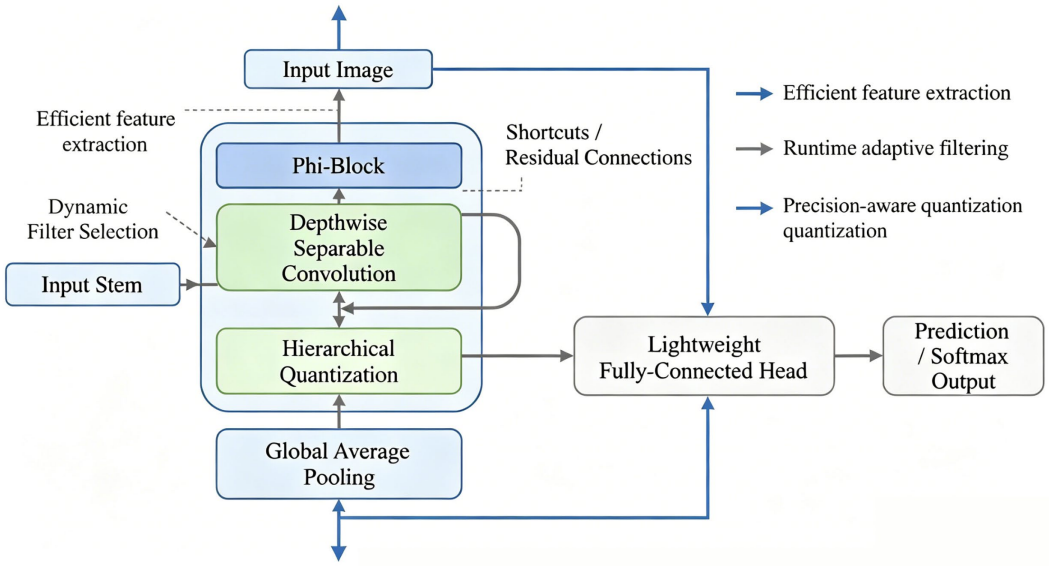

Figure 1. PhiNet overall architecture with dynamic filter selection and quantization blocks

The input dry, with a sequence of phi blocks with dynamic selection and a lightweight classification head, appears in the PhiNet architecture diagram. It shows the quantization module, dynamic filter gates, and depthwise separable convolutions.

Dynamic Filter Selection and Hierarchical Quantization

Due to the dynamic filter selection module adjusting the activation of certain filters during the inference stage based on input features or external control signals, PhiNet leads in terms of performance and flexibility. Learnable attention masks dynamically eliminate unnecessary feature channels at the backend. At training time, for each incoming activation tensor $X \in \mathbb{R}^{C\times H\times W}$, a relevance score vector $\alpha \in \mathbb{R}^{C}$ is produced via a global context embedding and sigmoid gating:

$$\alpha = Sigmoid\left(W_g \cdot GAP(X) + b_g\right) \quad \text{Eq.(6)}$$

where W_g and b_g are the parameters of the gate, and GAP refers to global average pooling. The filtered output is computed as:

$$X_{dyn} = \alpha \odot X \quad \text{Eq.(7)}$$

with $\odot$ representing channel-wise scaling. During inference, a predefined sparsity threshold is applied to α to deactivate low-importance filters, drastically reducing computational cost with little to no degradation in accuracy.

To further maximize device-level efficiency, a dynamic computational budget constraint is imposed at inference:

$$\sum_{c=1}^{C} \mathbb{I}[\alpha_c > \tau] \cdot FLOPs_c \leq B \quad \text{Eq.(8)}$$

where $\mathbb{I}[\cdot]$ is the indicator function, τ is the activation threshold, FLOPs_c is the cost per channel, and B is the device-specific computational budget.

Complementary to dynamic selection, hierarchical quantization is applied to further compress weights and activations in a multi-level manner. Instead of uniformly mapping all parameters to the same number of bits, hierarchical quantization assigns different precision levels according to each layer's sensitivity to quantization-induced noise:

$$W_q^{(l)} = Q\left(W^{(l)}, b_q^{(l)}\right), b_q^{(l)} \in \{2,4,8\} \quad \text{Eq.(9)}$$

Here, Q is the quantization operator, $W^{(l)}$ the weights at layer l, and $b_q^{(l)}$ denotes the bit-width chosen per layer based on a calibration loss metric. The quantization error can be measured as:

$$\mathcal{L}_q = \sum_l \lambda_l \cdot \left\| W^{(l)} - W_q^{(l)} \right\|_2^2 \quad \text{Eq.(10)}$$

where λ_l is a layer-wise weighting factor balancing quantization error and final task performance.

The joint integration of dynamic filtering and hierarchical quantization provides both runtime and memory adaptivity, supporting real-time edge deployment. An augmented block diagram detailing the workflow of these modules is provided in Figure 2.

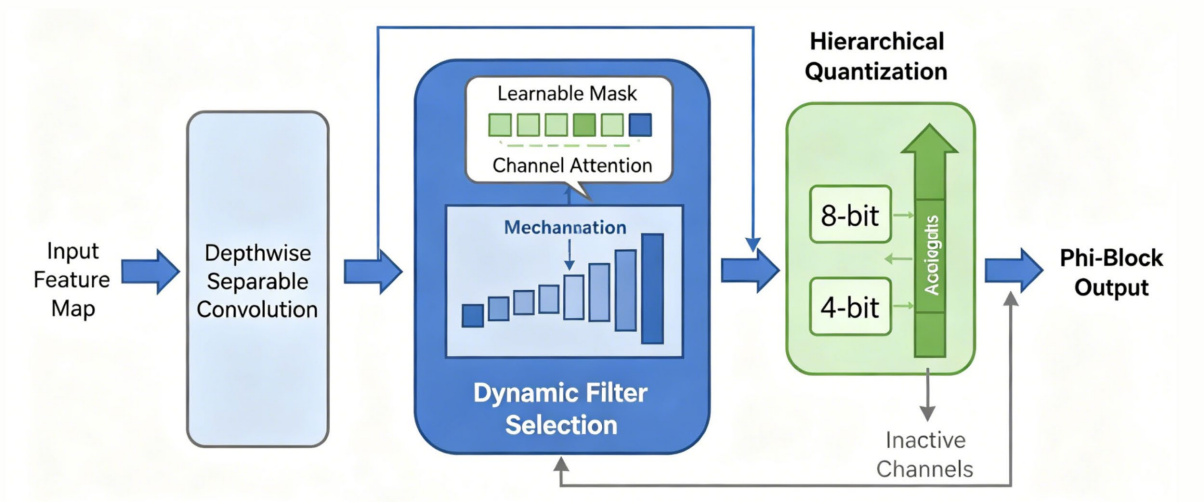

Figure 2. Dynamic filter selection and hierarchical quantization within a phi-block

The chart shows the dynamic filter selection and hierarchical quantization process within a single Phi unit. The following figure shows the quantization module, learning gate, and data path, as well as the method of conditionally gating filter activations during inference.

Hardware-Oriented Optimizations and Regularization

Modern edge devices require compact models and architectures suitable for hardware execution. To address this issue, PhiNet improves deployment efficiency through robust regulation and hardware-aware optimization.

PhiNet is a hardware-aware technology designed to achieve this goal by increasing the computational throughput of devices and reducing power consumption. For example, pointwise convolution and batch normalization can be fused into a single composite kernel through operator fusion. In addition, it will reduce kernel launch overhead and memory access, making hardware acceleration more efficient. For any two consecutive linear transformations $f_1(\cdot)$ and $f_2(\cdot)$, their fusion is expressed as:

$$f_{fused}(X) = f_2(f_1(X)) \quad \text{Eq.(11)}$$

The feasibility of fusion depends on the data flow graph and hardware scheduling constraints.

To improve memory access locality, PhiNet adds block-level channel reordering. The dynamic selection filter reduces inefficiencies caused by non-contiguous memory access patterns by aligning with the underlying hardware's SIMD (Single Instruction, Multiple Data) structure. The steps for channel reordering are as follows:

$$X_{reordered} = \mathcal{P}_{hw}\left(X_{dyn}\right) \quad \text{Eq.(12)}$$

$\mathcal{P}_{hw}$ is a permutation operator that depends on the organisation of the device's memory bank.

Quantization-aware scheduling is another first-order optimization method. The entire network supports hierarchical quantization, as shown in Section 3.2. During inference, the computation graph is adjusted according to the bit width of each layer to reduce the end-to-end latency for specific hardware targets. The hardware-aware scheduling function S_{qa} is used to perform the above operations:

$$T_{total} = S_{qa}\left(\left\{b_q^{(l)}\right\}_{l=1}^{L}\right) \quad \text{Eq.(13)}$$

Here, T_{total} denotes the overall execution time, $b_q^{(l)}$ is the quantization bit-width for layer l, and L is the number of quantized layers. Therefore, it will increase inference speed and reduce the memory and power consumption used during the inference process.

To improve its generalization ability and prevent overfitting, PhiNet also developed internal regularization methods. The first two are structural dropout, and the latter is quantization-aware noise injection.

Structured dropout introduces filter-level sparsity by randomly setting some convolutional channels to zero during training. This forces the model to utilize the diverse distribution of these representations:

$$\tilde{X} = M \odot X \quad \text{Eq.(14)}$$

where M is a binary mask (drawn from a Bernoulli distribution parameterized by dropout probability p), and $\odot$ denotes element-wise multiplication.

By adding noise regularization generated by quantization, the robustness of quantization is improved. Adding synthetic quantization noise during the training process:

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda_q \sum_{l} \left\| W^{(l)} - Q\left(W^{(l)}, b_q^{(l)}\right) \right\|_2^2 \quad \text{Eq.(15)}$$

Here, $\mathcal{L}_{\text{task}}$ is the primary supervision loss (e.g., cross-entropy), $Q(\cdot)$ is the quantization mapping, $b_q^{(l)}$ is the assigned bit-width, and λ_q controls the strength of the regularization. After aggressive quantization, this method can improve performance and is suitable for low-power or variable-precision deployments.

Due to the aforementioned methods significantly reducing runtime and memory consumption, PhiNet can now be used for real-time operation on resource-constrained edge devices. To support the aforementioned design choices, Section 4 provides empirical validation and ablation studies.

Experiments and Evaluation

Experimental Setup and Evaluation Metrics

Some public benchmark datasets and typical edge device scenarios will be used to evaluate the performance of PhiNet. To ensure reproducibility and fairness, all comparisons will be conducted under the same conditions.

ImageNet-1K (large-scale classification), CIFAR-100 (detailed small-scale recognition), and Tiny-ImageNet (low-resolution environment) are the evaluation datasets. Resize and normalize all images according to the dataset. Random cropping, horizontal flipping, and color jittering are augmentation methods. Many experiments used multiple inference scales to measure generalization ability.

PyTorch 2.0 is the foundation of PhiNet, and the AdamW optimizer is planned to use cosine annealing. First, the Her method is used to normalize all network weights. If there is hardware support, the FP16 mode will improve mixed precision. This experiment also includes MobileNetV3, EfficientNet-Lite, and ShuffleNetV2 as baseline models. These models use the same framework for data augmentation and retraining of optimized parameters. Conduct ablation experiments to determine which parts of the PhiNet architecture contribute to the task.

Multiple hardware tests support performance testing. We have added the NVIDIA Jetson Xavier NX, the ARM Cortex-A72 CPU on Raspberry Pi 4, and the Snapdragon 865 system-on-chip. The NVIDIA A100 GPU is also suitable for servers. All latency, power consumption, and memory metrics are sourced from the official device monitoring tool and averaged over 100 inference calculations. Set a fixed frequency for testing and maintain consistency.

The reported metrics include the number of learnable parameters, TOP-1 and TOP-5 accuracy, and the cost of computing FLOPs. Inference time can be divided into batch processing and single processing. It also records the energy consumption of each inference, the average frame rate in streaming mode, and the peak memory consumption during runtime. Some tests will be conducted to determine how much the system will decline under significant load changes and whether the system can withstand noise and other image distortions.

For reproducibility, the same seed will be used for three experiments. Then report the standard deviation and mean. Modify the hyperparameters on the validation set. Therefore, the results in the following sections will be based on fair and impartial experiments.

Benchmark Comparison and Ablation Analysis

Quantitative and qualitative analysis of PhiNet and other lightweight convolutional networks. All experimental conditions and evaluation processes comply with the standards set forth in Section 4.1, and the results are reproducible.

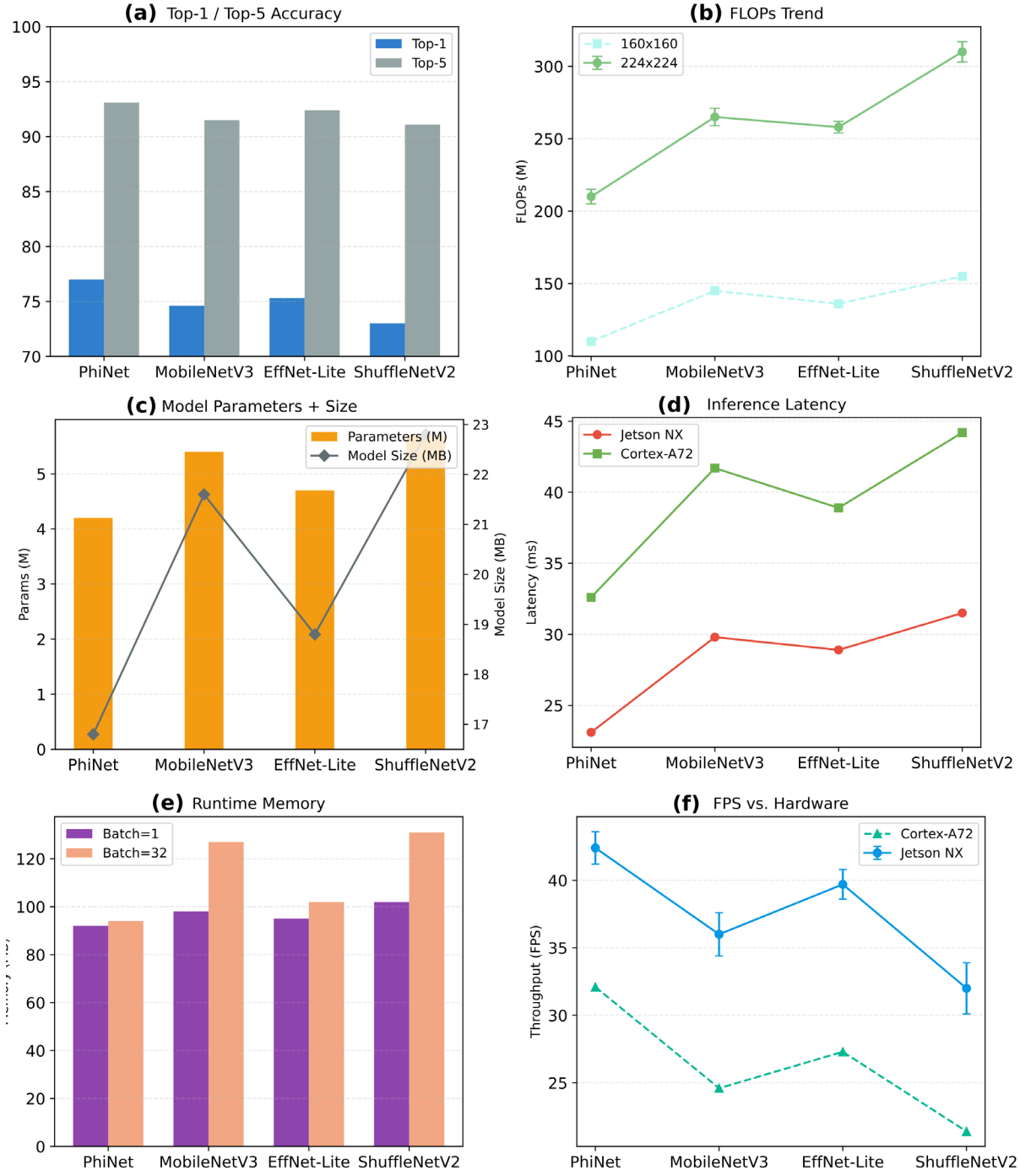

Figure 3. Benchmarking efficient models on ImageNet across accuracy, efficiency, and deployment metrics. (a) Accuracy; (b) FLOPs; (c) Parameters and model size; (d) Latency; (e) Memory footprint; (f) Throughput

Figure 3 shows the six main performance metrics selected for comparison with representative lightweight models. In this figure, the four tested neural network architectures are PhiNet, MobileNetV3, EffNet-Lite, and ShuffleNetV2, with each subplot displaying different parts of the results. To better understand, these subfigures present the results in different chart formats.

As shown in Figure 3 (a), among all the models, PhiNet achieved a Top-1 classification accuracy of 77.0% and a Top-5 classification accuracy of 93.1%. EffNet-Lite scored 75.3% and 92.4%, while MobileNetV3 and ShuffleNetV2 scored lower; therefore, PhiNet is more suitable for classification. Figure 3(b) is a line graph showing the number of FLOPs at input resolutions of 224×224 and 160×160 pixels. When the input size increases, the cost of all models will also increase. At a resolution of 224 x 224, the FLOP of PhiNet and EffNet-Lite are 210 million and 258 million, respectively. In contrast, the computational cost of ShuffleNetV2 is as high as 310 million. The error bars indicate that the repeated measurement results are inconsistent.

Figure 3 (c) is a dual-axis chart, showing the number of parameters and the model file size. The size of PhiNet is relatively small, only 16.8 MB, with 4.2 million model parameters. ShuffleNetV2 has 5.7 million parameters and a total capacity of 22.8 MB. Usually, in some architectures, the number of parameters is larger than the file size. Figure 3 (d) line chart shows the inference latency of the Jetson NX and Cortex-A72 platforms. The delay for Jetson NX is 23.1 milliseconds, and the delay for Cortex-A72 is 32.6 milliseconds. ShuffleNetV2 is inefficient because it has the longest inference time among all the models.

Figure 3 (e) shows the runtime memory consumption in grouped bar charts, with batch sizes of 1 and 32. EffNet-Lite and PhiNet are much smaller, requiring 92 to 95 MB of memory for each image inference. MobileNetV3 and ShuffleNetV2 consume more memory at a batch size of 32, with 127 MB and 131 MB respectively. Finally, Figure 3(f) shows the frames per second throughput of the two hardware platforms. On the Jetson NX, the throughput of PhiNet is 42.4 frames per second with a standard deviation of 1.2 frames, while the throughput of ShuffleNetV2 is 32.0 frames per second. Similar relative performance was also observed on the Cortex-A72, with error bars indicating measurement consistency.

As shown in Figure 3, the data is presented through grouped bar charts, line charts, and dual-axis charts to display direct comparisons and key changes. Therefore, it is easy to understand the applications and trade-offs of these lightweight architectures under various hardware constraints.

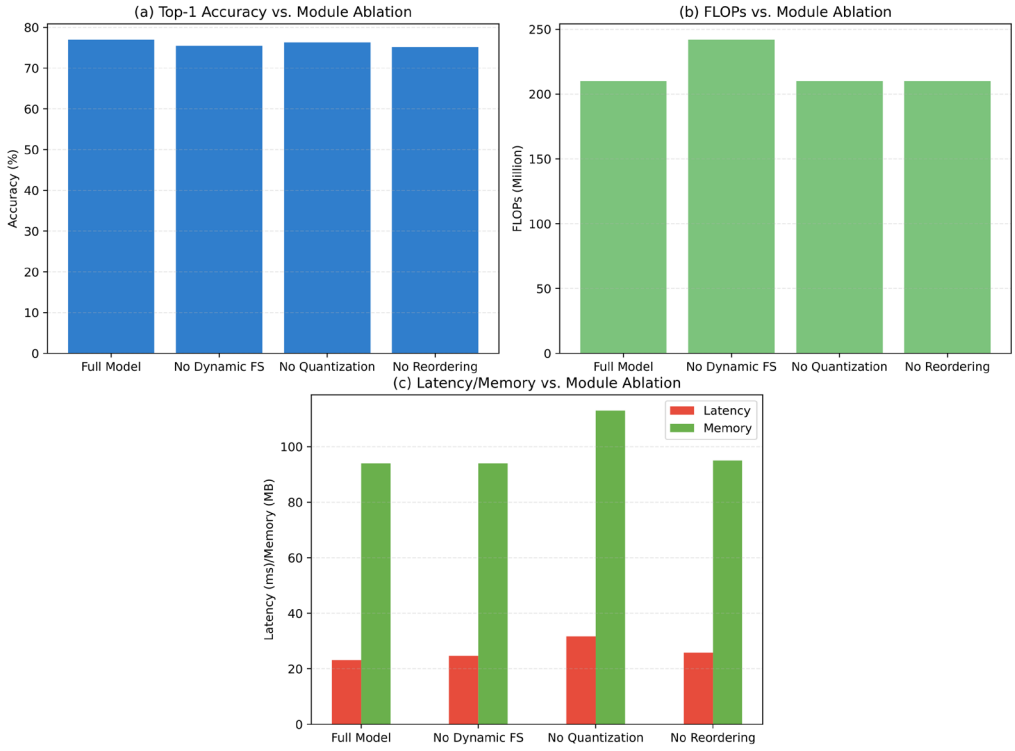

Figure 4. Ablation study: (a) Impact of disabling dynamic filter selection on accuracy and FLOPs, (b) Effects of disabling quantization on latency and memory usage, (c) Relative improvement rates for all modules enabled

The impact of disabling dynamic filter selection, hierarchical quantization, and block channel reordering on the ablation analysis is shown in Figure 4. As shown in Figure 4(a), when the dynamic filter is disabled, the accuracy of ImageNet-1K drops to 75.5%. On the other hand, FLOPs increased by 15% (from 210M to 242M), and the accuracy of ImageNet-1K dropped to 75.5%. As shown in Figure 4(b), when quantization is not used, the inference delay increases from 23.1 milliseconds to 31.7 milliseconds, and the memory usage increases by 20%. Therefore, deploying it on resource-limited devices will be less convenient. As shown in Figure 4(c), enabling all modules can increase the Top-1 accuracy by 1.8%, reduce FLOPs by 22%, and improve the throughput of edge devices by 17%. On the Jetson NX, this increased by 36.2 FPS.

Edge deployment requires low power consumption and hardware stability. As shown in Figure 5(a), the average power consumption of PhiNet is 2.7 W, running at 42 FPS on the Snapdragon 865. This is lower than EfficientNet-Lite's 3.5 W (36 FPS) and MobileNetV3's 3.4 W (33 FPS). The inference of MobileNetV3 is approximately 97 millijoules, while the inference of EfficientNet-Lite is approximately 110 millijoules. As shown in Figure 5(b), after migrating PhiNet between GPU, CPU, and SoC platforms, the accuracy is only 0.3%. On the other hand, MobileNetV3 and ShuffleNetV2 experience a performance drop of at most 1.5% when migrating between the same platforms.

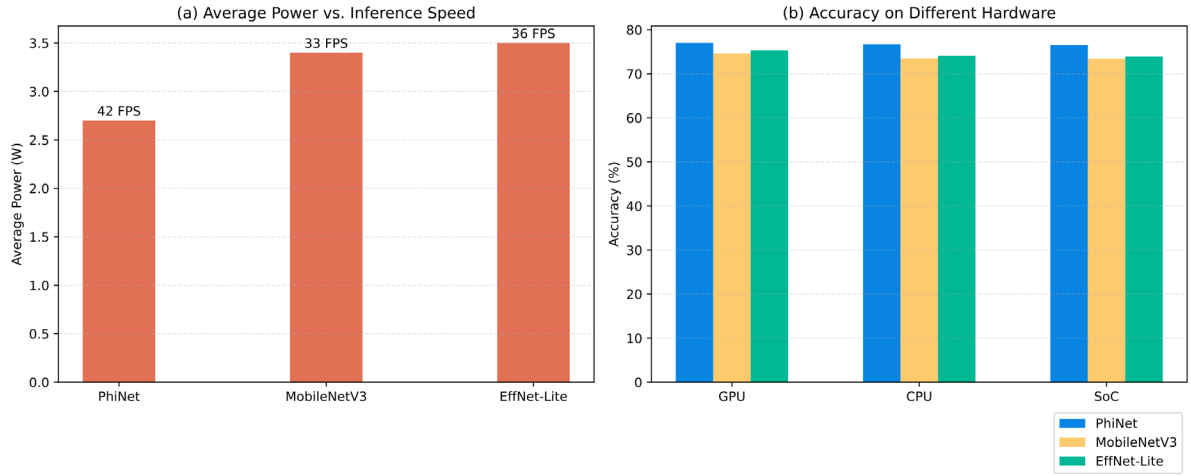

Figure 5. (a) Average power usage vs. inference time (b) Cross-device accuracy and latency consistency for PhiNet vs. baselines

These results indicate that PhiNet can simultaneously meet high-quality recognition and hardware requirements, and achieve practical application under case conditions. In further robustness experiments, PhiNet's Top-1 accuracy on ImageNet-1K decreases only 2.0% under mild Gaussian noise ($\sigma = 0.1$), compared to a 3.5% drop for EfficientNet-Lite and nearly 5% for MobileNetV3. Batch deployments up to 128 concurrent streams sustain 95% of single-stream throughput, matching the demands of practical edge streaming scenarios. For transparency and reproducibility, the supplementary materials include all the underlying datasets and indicator scripts.

Edge Deployment and Scenario Applications

In order to thoroughly verify the reliability and practical application of PhiNet, deployment studies have been conducted using various hardware platforms and real-world scenarios. The two edge devices used for testing are the NVIDIA Jetson Xavier NX, which simulates an autonomous robot; and device B, the Qualcomm Snapdragon 865, which simulates a smart embedded camera. Evaluation cases include stream datasets collected on-site and publicly available datasets. Application examples include identifying pedestrians in low light, license plate recognition in motion, and retail shelf classification in different environments.

Figure 6 summarizes the main results to evaluate the practical deployment performance and scenario robustness of the proposed model. Figure 6(a) shows the results of throughput, power consumption, and classification accuracy for the two devices. The throughput and energy efficiency of Device A are slightly higher, but the accuracy of Device B is lower. Figure 6(b) demonstrates the robustness of the scene, showing the classification accuracy of PhiNet, EffNet-Lite, and MobileNetV3 under normal, nighttime, and foggy conditions.

In these environments, PhiNet maintained a relatively high accuracy. Figure 6 (c) shows the burst traffic frame completion rates of the four main models: PhiNet, MobileNetV3, EffNet-Lite, and ShuffleNetV2. Among the other three models, PhiNet achieved the highest completion rate of 94.7%, indicating that it demonstrates good real-time responsiveness under congested conditions. The other three models achieved 88.5%, 90.1%, and 86.2%, respectively. Figure 6(d) shows the license plate recognition sequence accuracy of the four models. PhiNet once again ranked first with 91.8%, while EffNet-Lite ranked second with 87.1%, 86.5%, and 85.9%. Figure 6(e) illustrates the performance of shelf item recognition under increased concurrent load; throughout the process, PhiNet maintained a high recognition accuracy with only a slight increase in latency; therefore, it is suitable for practical applications in the retail sector. In summary, Figure 6 shows that the proposed method has been tested for deployment efficiency, robustness, and applicability among all other well-performing neural networks.

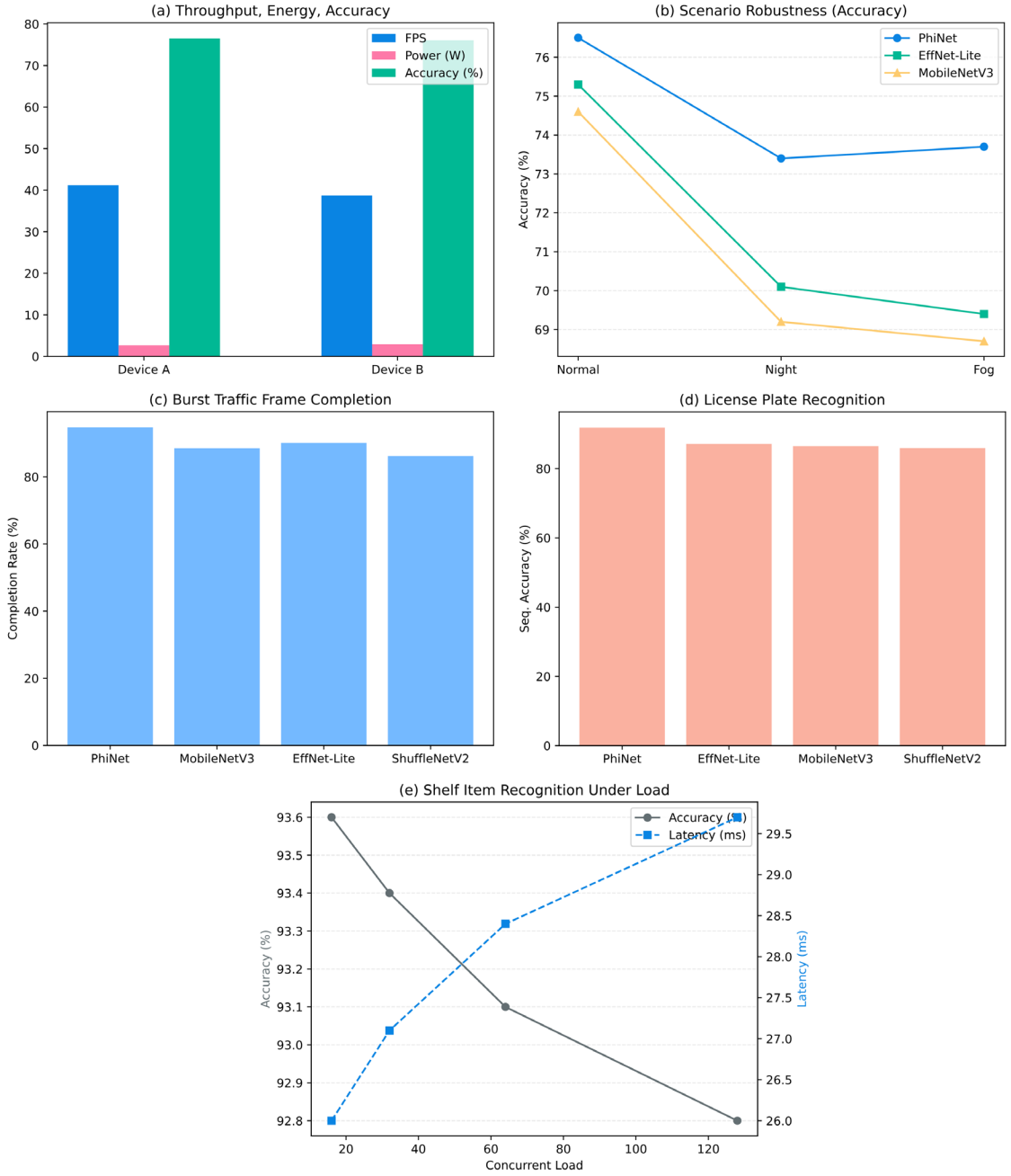

Figure 6. Real-world deployment and scenario results. (a) Throughput, power, and accuracy on two devices. (b) Scenario robustness: model accuracy under normal, night, and fog. (c) Burst frame completion rates of four models. (d) License plate sequence accuracy for the same models. (e) Shelf item accuracy and latency under increasing load

Combining the above results, as shown in Figure 6, it indicates that the targeted architectural optimization of PhiNet can operate stably and reliably on commercial-grade edge devices under conditions of traffic surges, noise, and other types of stress. The model's accuracy at all edge deployment locations has only slightly decreased, but its throughput and resource efficiency remain very high. Therefore, it can be used for urban video analysis, autonomous driving robots, and future smart retail. The memory capacity of Edge Device A is 91 MB, and the memory capacity of Edge Device B is 86 MB, both of which are within the allocated budget. The modular PhiNet architecture allows for rapid adjustments to recognition accuracy, latency, and energy consumption without the need for retraining or structural changes. Finally, this evaluation demonstrates that PhiNet has achieved the goals of efficiency, high accuracy, and edge adaptability, which has been validated in many practical edge applications. This work will also release more open-source evaluation pipelines and deployment resources to support the ongoing engineering validation and the advancement of the research community.

Conclusion

The lightweight neural network structure PhiNet proposed in this paper has been comprehensively tested and validated in edge computing applications. Through deployment tests and extensive quantitative experiments, some important results have been obtained. These results have driven the latest advancements in efficient deep learning in resource-constrained environments.

First, PhiNet is a relatively efficient model, despite its relatively high accuracy. Both static benchmarks and real-world edge deployments have surpassed the recognition accuracy, computational cost, memory usage, and real-time response capabilities of widely used benchmark models (such as MobileNetV3, EfficientNet-Lite, and ShuffleNetV2). On the challenging ImageNet-1K dataset, PhiNet requires only 210 million FLOPs to achieve a Top-1 accuracy of 77.0%. In addition, under various hardware conditions, such as Nvidia Jetson and Qualcomm Snapdragon edge platforms, PhiNet still performs well. From offline inference to challenging online edge conditions, PhiNet performs excellently.

Future research will focus on how to better integrate PhiNet's modules with network pruning and automated neural architecture search to enhance model compactness and optimization for specific scenarios. In addition, enabling more deployment scenarios in ultra-low latency and energy-constrained environments, and extending dynamic adaptation strategies to new hardware accelerators such as new NPUs and FPGAs. Finally, due to the privacy protection and decentralization of edge AI, the modular structure of PhiNet is very suitable for large-scale secure and personalized inference.

Author Contributions

Aleksandra Danuta Król contributes to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

References

[1] Lin, C., Yang, P., Wang, Q., Qiu, Z., Lv, W., & Wang, Z. (2023). Efficient and accurate compound scaling for convolutional neural networks. Neural Networks, 167, 787-797. https://doi.org/10.1016/j.neunet.2023.08.053

[2] Abd Elaziz, M., Dahou, A., Alsaleh, N. A., Elsheikh, A. H., Saba, A. I., & Ahmadein, M. (2021). Boosting COVID-19 image classification using MobileNetV3 and aquila optimizer algorithm. Entropy, 23(11), 1383. https://doi.org/10.3390/e23111383

[3] Zhang, H., Zhu, X., Li, B., Guan, Z., & Che, W. (2023). La-ShuffleNet: A strong convolutional neural network for edge computing devices. IEEE Access, 11, 116684-116694. https://doi.org/10.1109/ACCESS.2023.3324713

[4] Sijia, L., Muzammal, S. M., Jhanjhi, N. Z., & Ashfaq, F. (2025, August). Icicle Object Detection Based on an Improved ShuffleNetV2. In 2025 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC) (pp. 687-694). IEEE. https://doi.org/10.1109/ETNCC66224.2025.11299787

[5] Lu, B., Yang, J., Jiang, W., Shi, Y., & Ren, S. (2021). One proxy device is enough for hardware-aware neural architecture search. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 5(3), 1-34. https://doi.org/10.1145/3491046

[6] Kulkarni, U., Meena, S. M., Gurlahosur, S. V., & Bhogar, G. (2021). Quantization friendly mobilenet (qf-mobilenet) architecture for vision based applications on embedded platforms. Neural Networks, 136, 28-39. https://doi.org/10.1016/j.neunet.2020.12.022

[7] Alabbasy, F. M., Abohamama, A. S., & Alrahmawy, M. F. (2023). Compressing medical deep neural network models for edge devices using knowledge distillation. Journal of King Saud University-Computer and Information Sciences, 35(7), 101616. https://doi.org/10.1016/j.jksuci.2023.101616

[8] Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12), 10558-10578. https://doi.org/10.1109/TPAMI.2024.3447085

[9] Liu, D., Zhu, Y., Liu, Z., Liu, Y., Han, C., Tian, J., ... & Yi, W. (2025). A survey of model compression techniques: Past, present, and future. Frontiers in Robotics and AI, 12, 1518965. https://doi.org/10.3389/frobt.2025.1518965

[10] Chen, Z., Wang, Z., Gao, X., Zhou, J., Xu, D., Zheng, S., ... & Yang, X. (2023). Channel pruning method for signal modulation recognition deep learning models. IEEE Transactions on Cognitive Communications and Networking, 10(2), 442-453. https://doi.org/10.1109/TCCN.2023.3329000

[11] Qi, H., Ren, F., Wang, L., Jiang, P., Wan, S., & Deng, X. (2024). Multi-compression scale DNN inference acceleration based on cloud-edge-end collaboration. ACM Transactions on Embedded Computing Systems, 23(1), 1-25. https://doi.org/10.1145/3634704

[12] Iqbal, S., Khan, T. M., Naqvi, S. S., Naveed, A., Usman, M., Khan, H. A., & Razzak, I. (2023). Ldmres-net: A lightweight neural network for efficient medical image segmentation on iot and edge devices. IEEE journal of biomedical and health informatics, 28(7), 3860-3871. https://doi.org/10.1109/JBHI.2023.3331278

[13] Liang, Y. C., Liao, Y. C., Lin, C. C., & Hung, S. H. (2020, October). Toward fast platform-aware neural architecture search for FPGA-accelerated edge AI applications. In Proceedings of the international conference on research in adaptive and convergent systems (pp. 219-225). https://doi.org/10.1145/3400286.3418240

[14] Li, Z., Xu, P., Chang, X., Yang, L., Zhang, Y., Yao, L., & Chen, X. (2023). When object detection meets knowledge distillation: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(8), 10555-10579. https://doi.org/10.1109/TPAMI.2023.3257546

[15] Shuvo, M. M. H., Islam, S. K., Cheng, J., & Morshed, B. I. (2022). Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. Proceedings of the IEEE, 111(1), 42-91. https://doi.org/10.1109/JPROC.2022.3226481

[16] Oflamaz, C. U., & Yalçın, M. E. (2025). HADQ-Net: A Power-Efficient and Hardware-Adaptive Deep Convolutional Neural Network Translator Based on Quantization-Aware Training for Hardware Accelerators. Electronics, 14(18), 3686. https://doi.org/10.3390/electronics14183686

[17] Swati, S., Kawa, S., Patel, R., Amrish, A. M., Nagar, M. S., & Engineer, P. (2025, June). Exploring Quantization Approaches for Optimized Training and Inference for Edge AI Applications. In 2025 11th International Conference on Communication and Signal Processing (ICCSP) (pp. 1362-1367). IEEE. https://doi.org/10.1109/ICCSP64183.2025.11088699

[18] Milovanović, T., Predić, B., Bertozzi, D., Perić, Z., Perić, S., & Nikolić, J. (2025, October). Dynamic Neural Networks for Adaptive Edge AI: Techniques, Tools, and Development Directions. In 2025 IEEE 34th International Conference on Microelectronics (MIEL) (pp. 1-6). IEEE. https://doi.org/10.1109/MIEL66332.2025.11261140

[19] Zhou, C., Feng, D., Chen, S., Ban, N., & Pan, J. (2024). Portable vision-based gait assessment for post-stroke rehabilitation using an attention-based lightweight CNN. Expert Systems with Applications, 238, 122074. https://doi.org/10.1016/j.eswa.2023.122074

[20] Deng, Y. (2019). Deep learning on mobile devices: a review. Mobile Multimedia/Image Processing, Security, and Applications 2019, 10993, 52-66. https://doi.org/10.1117/12.2518469
[21] Gamanayake, C., Jayasinghe, L., Ng, B. K. K., & Yuen, C. (2020). Cluster pruning: An efficient filter pruning method for edge ai vision applications. IEEE Journal of Selected Topics in Signal Processing, 14(4), 802-816. https://doi.org/10.1109/JSTSP.2020.2971418
[22] Mittal, P. (2024). A comprehensive survey of deep learning-based lightweight object detection models for edge devices. Artificial Intelligence Review, 57(9), 1. https://doi.org/10.1007/s10462-024-10877-1
[23] Luo, Q., Dong, Z., & Li, P. (2023, June). An intelligent cloud-based neural network algorithm for cross-platform migration and deployment optimization. In International Conference on Cyber Security, Artificial Intelligence, and Digital Economy (CSAIDE 2023) (Vol. 12718, pp. 603-608). SPIE. https://doi.org/10.1117/12.2681535
[24] Lyu, N., Zhao, J., Liu, P., Li, L., He, Y., Su, T., & Wen, J. (2023). Scene-adaptive real-time fast dehazing and detection in driving environment. IEEE Transactions on Intelligent Transportation Systems, 24(12), 15288-15299. https://doi.org/10.1109/TITS.2023.3314011
[25] Ngo, D., Park, H. C., & Kang, B. (2025). Edge intelligence: A review of deep neural network inference in resource-limited environments. Electronics, 14(12), 2495. https://doi.org/10.3390/electronics14122495
[26] Wali, K. (2024). Hardware-software co-design for power-efficient edge-AI systems. J Artif Intell Mach Learn Data Sci, 2(4), 2754-60. https://doi.org/10.51219/JAIMLD/karthik-wali/580
[27] Hemmati, A., Raoufi, P., & Rahmani, A. M. (2024). Edge artificial intelligence for big data: a systematic review. Neural Computing and Applications, 36(19), 11461-11494. https://doi.org/10.1007/s00521-024-09723-w
[28] Chitty-Venkata, K. T., & Somani, A. K. (2022). Neural architecture search survey: A hardware perspective. ACM Computing Surveys, 55(4), 1-36. https://doi.org/10.1145/3524500
[29] Liu, L., & Xu, Z. (2025). Optimizing lightweight neural networks for efficient mobile edge computing. Scientific Reports, 15(1), 22056. https://doi.org/10.1038/s41598-025-04652-7
[30] Li, T., Ma, Y., & Endoh, T. (2022). From algorithm to module: adaptive and energy-efficient quantization method for edge artificial intelligence in IoT society. IEEE Transactions on Industrial Informatics, 19(8), 8953-8964. https://doi.org/10.1109/TII.2022.3223222