

Article

Rotating Kernel CNN Optimization for Efficient IoT Surveillance on Low-Power Devices

Fei Wang *, Songwei Hou, Jianbang Gao and Dan Wu

School of Electronic Engineering, Xi'an ShiYou University, Xi'an 710312, China;
22211030362@stumail.xsyu.edu.cn (S.H.); gjbang2023@xsyu.edu.cn (J.G.); wudan@xsyu.edu.cn (D.W.)

* Correspondence: 200102@xsyu.edu.cn

Abstract: This study presents an innovative method to optimize convolutional neural networks (CNNs) tailored for the prevalent low-power platforms in IoT and embedded devices, which are integral to smart city infrastructures. Our approach introduces a novel “Rotating Convolutional Kernel” technique, designed to significantly reduce the computational load of CNNs while maintaining high accuracy, an essential feature for the constrained processing capabilities of devices produced by Hisilicon, MediaTek, and Novatek, among others. By leveraging the intelligent video engine (IVE) capabilities inherent in low-end CPUs, our optimized YOLOv3-Tiny model, with fewer than 100 K parameters and specifically fine-tuned for pedestrian and vehicle detection, demonstrates impressive processing speeds of 52 ms/frame on the HI3516EV200 chip and 66 ms/frame on the MSC313E chip, with minimal accuracy compromise. Despite a substantial reduction in parameter count compared to that in MobileNetV2, our model’s top-1 accuracy only slightly decreases by 2.6%, showcasing the effectiveness of our optimization technique. Our findings highlight the potential and applicability of our method in enhancing the performance and utility of IoT and embedded devices within smart cities. By achieving an optimal balance between computational efficiency and detection accuracy, our approach offers a promising avenue for advancing the capabilities of low-power devices in urban surveillance, traffic management, and other smart city applications, thereby contributing to the development of more intelligent, efficient, and responsive urban environments.

Keywords: convolutional neural network; intelligent video engine; resource-limited settings



Citation: Wang, F.; Hou, S.; Gao, J.; Wu, D. Rotating Kernel CNN Optimization for Efficient IoT Surveillance on Low-Power Devices. *Electronics* **2024**, *13*, 2862. <https://doi.org/10.3390/electronics13142862>

Academic Editor: Stefanos Kollias

Received: 5 July 2024

Revised: 17 July 2024

Accepted: 19 July 2024

Published: 20 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The advent of high-definition video technology, coupled with advancements in smart analytics, cloud computing, and big data, has significantly transformed security systems from passive defense mechanisms into proactive, intelligent early warning systems. This evolution has not enhanced security measures but also broadens their applications across various sectors, fostering a smarter lifestyle and offering intelligent solutions to a wider array of industries and individuals. The shift towards an integrated cloud-edge industrial ecosystem, characterized by the architectural progression of video surveillance to include front-end intelligence, back-end cloud processing, and a three-tiered structure comprising edge nodes, edge domains, and cloud centers, marks a significant paradigm shift in the security sector. This new trend underscores the importance of adaptive, real-time surveillance solutions capable of meeting the complex demands of modern security environments [1,2].

However, despite these technological strides, a significant portion of the security market is still dominated by non-intelligent edge nodes, which are constrained by limited computing resources and face significant performance challenges. This limitation hinders the deployment of intelligent algorithms on such devices, necessitating innovative solutions to achieve high-performance analytics on low-power devices. Presently, about 80% of security front-end devices in the market rely on solutions from providers like Hisilicon, MediaTek, and Novatek, which often lack the computational power required for advanced

video analysis. These devices are predominantly used for basic video surveillance and recording, with only a minor fraction capable of conducting simple video analysis functions, which suffer from high false alarm rates and limited utility [3,4].

In response to these challenges, this study introduces an innovative approach to optimize convolutional neural networks (CNNs) for resource-constrained security devices. Central to our methodology is the development of a “Rotating Convolutional Kernel”, which significantly reduces computational demands while maintaining high accuracy. This approach, combined with max-filtering and direct memory access (DMA) for efficient max-pooling, maximizes the performance of the intelligent video engine (IVE) in low-end CPUs. Our tailored YOLOv3-Tiny model, specifically optimized for pedestrian and vehicle detection, demonstrates remarkable processing speeds and minimal accuracy degradation on low-resource platforms.

By addressing the computational limitations of current security devices, this research aims to bridge the gap between the growing demand for intelligent security solutions and the capabilities of existing hardware. The implications of this work extend beyond the security industry, offering insights into the design of efficient neural network architectures for a variety of applications in resource-constrained environments.

The remainder of this paper is organized as follows: Section 2 introduces relevant research and methods. Section 3 presents the units and architecture of the proposed network. Section 4 covers the related experimental content. Finally, conclusions are provided in Section 5.

2. Related Works

The rapid progress in deep learning, particularly through convolutional neural networks (CNNs), has shown remarkable success across various domains, including audio and video processing. CNNs have the unique ability to learn from extensive datasets and generalize their findings to new, unseen data, making them particularly suited for complex security and surveillance tasks. The development of various network architectures, such as AlexNet [5], VGGNet [6], GoogLeNet [7], ResNet [8], and DenseNet [9], with increasing complexity and depth has been instrumental in advancing the field. However, the enhanced performance of these networks comes at the cost of increased network parameters and computational demands, necessitating efficient network design and optimization to make them feasible for deployment in resource-constrained environments.

Addressing the efficiency challenges in CNNs involves a multifaceted approach, including the development of lightweight models, model compression techniques, and hardware optimization strategies. Lightweight models such as SqueezeNet [10], MobileNet [11–13], and ShuffleNet [14,15] aim to reduce computational demands through innovative convolutional approaches, making them more suitable for deployment on devices with limited computational resources. Model compression techniques, including quantization and pruning, strive to maintain network performance while reducing the model size and computational complexity. Hardware optimization techniques seek to maximize the utilization of available CPU or GPU resources, with inference frameworks like NCNN [16], MNN [17], and TNN [18] enhancing convolutional operations at the instruction level. These optimizations have made it possible to apply advanced network models on mobile and edge devices, extending the reach of deep learning technologies to a wider array of applications [19].

However, the application of sophisticated CNN algorithms on security front-end devices without neural processing units (NPUs) and with limited CPU capabilities remains a significant challenge. The development of efficient CNN algorithms suitable for such platforms is crucial for extending the application of deep learning technologies across the security industry. This involves not only optimizing the algorithms themselves but also considering the unique constraints and requirements of security surveillance applications. The recent literature has emphasized the importance of model efficiency and deployment strategies that cater to the constraints of edge devices, highlighting the ongoing efforts

to make deep learning technologies more accessible and effective in resource-constrained settings [20–22].

By addressing these challenges, the security industry can harness the full potential of deep learning to enhance surveillance capabilities, reduce false alarms, and provide more intelligent solutions that meet the evolving needs of diverse industries and individuals. The continued development and optimization of CNNs for low-resource environments represent a key area of research that holds significant promise for the future of security and surveillance technologies.

3. Approach

3.1. Rotating Convolutional Unit

In contemporary neural network architectures such as AlexNet [5] and VGGNet [6], a sequential arrangement of convolutional layers with varying parameters predominates, whereas frameworks like ResNet [8] and DenseNet [9] incorporate shortcut connections to enhance network depth and precision. Conversely, lightweight architectures including MobileNets and ShuffleNets [11–15] minimize parameter volume via group convolutions and the extensive employment of 1×1 convolutions, presenting a paradigm shift towards efficiency. Despite the success of these diverse structural designs in achieving remarkable performance, the significant parameter count necessitates considerable computational resources, rendering them less feasible for deployment in security devices with limited processing capabilities.

The most straightforward solution to this issue is to reduce the number of convolutional kernels to decrease the parameter count, but this reduction can lead to a decrease in accuracy, presenting a contradiction. To address this challenge, this paper fully leverages intelligent video engine (IVE) resources to design a convolutional kernel structure called “**Rotating Convolution**” (the IVE is a simple acceleration engine provided by many video processing chips that can speed up straightforward video processing algorithms). By rotating the same convolutional kernel (at 90, 180, and 270 degrees), this approach significantly reduces the number of parameters without decreasing the number of kernels. The specific methodology is as follows:

In the intelligent video engine (IVE) implementations of chip solutions from Hisilicon, MediaTek, and Novatek, a 5×5 Sobel operator is available that enables the simultaneous extraction of 16-bit horizontal and vertical edges from an 8-bit image. This operator features a customizable 5×5 convolutional kernel, with padding set to 2 and a stride of 1. Leveraging this capability, we can utilize the hardware resources to concurrently compute convolutions in two directions using a single kernel. The operator is defined in Equations (1) and (2),

$$H(x, y) = \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x + i, y + j) * K(i, j), \quad (1)$$

$$V(x, y) = \sum_{-2 < j < 2} \sum_{-2 < i < 2} I(x + i, y + j) * K(j, i), \quad (2)$$

where $H(x, y)$ and $V(x, y)$ represent the convolutions in the horizontal and vertical directions, I represents the input 8-bit image or data, and K denotes the 5×5 convolution kernel. $*$ represents convolution operation. K_{ij} denotes the kernel parameter at the i -th row and j -th column, while K_{ji} represents the corresponding parameter in the symmetric direction.

Rotational convolution ingeniously modifies the depthwise convolution approach by utilizing 5×5 convolutional filters, which are uniformly sized but strategically rotated to different orientations (0° , 90° , 180° , 270°), as delineated in Equations (3) and (4). Each convolutional kernel shares identical parameters, yet their positioning varies according to the angle of rotation, enhancing the model’s capability to capture features from multiple orientations with a consistent parameter set. This methodology not only amplifies the receptive field pertinent to feature extraction but also optimizes the exploitation of IVE

resources. Furthermore, it effectuates a substantial reduction in the convolutional filter parameters, diminishing them to a mere quarter of those necessitated by conventional convolutional approaches. The architectural configuration of this novel convolution unit is delineated in Figure 1. In the figure, (a) and (b) show two rotational convolution kernels with shortcuts, and (c) displays the full rotational convolution architecture. The next section will provide a detailed comparison of these structures through ablation studies.

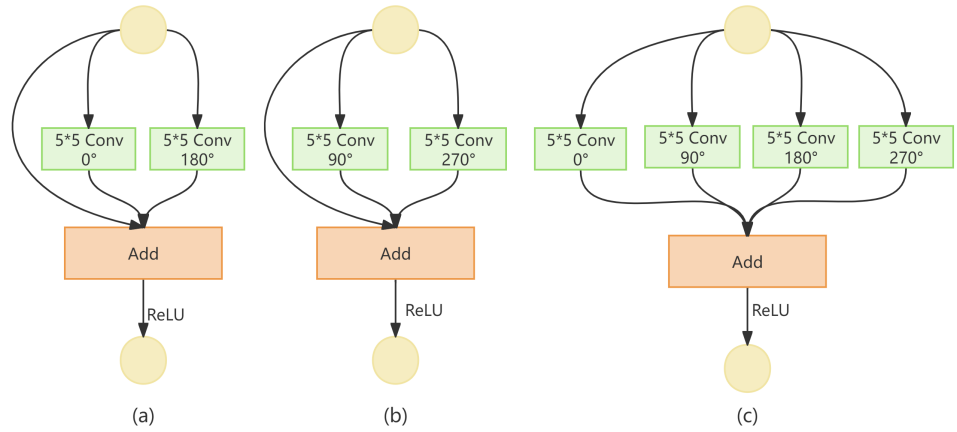


Figure 1. Rotating convolutional unit. (a,b) Unit rotating with shortcut; (c) Full-rotating convolutional unit.

$$K_0 = \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} & k_{04} \\ k_{10} & k_{11} & k_{12} & k_{13} & k_{14} \\ k_{20} & k_{21} & k_{22} & k_{23} & k_{24} \\ k_{30} & k_{31} & k_{32} & k_{33} & k_{34} \\ k_{40} & k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix}, K_{180} = \begin{bmatrix} k_{44} & k_{43} & k_{42} & k_{41} & k_{40} \\ k_{34} & k_{33} & k_{32} & k_{31} & k_{30} \\ k_{24} & k_{23} & k_{22} & k_{21} & k_{20} \\ k_{14} & k_{13} & k_{12} & k_{11} & k_{10} \\ k_{04} & k_{03} & k_{02} & k_{01} & k_{00} \end{bmatrix} \quad (3)$$

$$K_{90} = \begin{bmatrix} k_{40} & k_{30} & k_{20} & k_{10} & k_{00} \\ k_{41} & k_{31} & k_{21} & k_{11} & k_{01} \\ k_{42} & k_{32} & k_{22} & k_{12} & k_{02} \\ k_{43} & k_{33} & k_{23} & k_{13} & k_{03} \\ k_{44} & k_{34} & k_{24} & k_{14} & k_{04} \end{bmatrix}, K_{270} = \begin{bmatrix} k_{04} & k_{14} & k_{24} & k_{34} & k_{44} \\ k_{03} & k_{13} & k_{23} & k_{33} & k_{43} \\ k_{02} & k_{12} & k_{22} & k_{32} & k_{42} \\ k_{01} & k_{11} & k_{21} & k_{31} & k_{41} \\ k_{00} & k_{10} & k_{20} & k_{30} & k_{40} \end{bmatrix} \quad (4)$$

3.2. Pooling Layer Using MaxMum Filters and DMA

To minimize computational overhead with a 5×5 kernel, stride = 1, and padding = 2, we necessitate feature dimensionality reduction via a pooling layer. In the absence of native pooling functions in IVEs such as Hisilicon's, we approximate 2×2 max pooling through max filtering and DMA copying, detailed in Equations (5) and (6),

$$I_{out}(x, y) = \max_{i, j \in (-1, 1)} (I(x + i, y + j)) \quad (5)$$

$$F_{out}(x_1, y_1) = I_{x_1=x/2, y_1=y/2}(x, y) \quad (6)$$

where I_{out} represents the feature map after 3×3 max filtering, and F_{out} denotes the feature map following alternate row copying by DMA, thereby approximating max pooling.

3.3. Network Architecture

The structure of the network is built on rotational convolutions as mentioned in the previous section. By defining the network in such simple terms, we are able to easily explore network topologies to find a good network with less parameters. The architecture is defined

in Table 1. “Type” represents the operator type, “Filter Shape” indicates the size of the convolutional kernel, “InputSize” denotes the size of the feature map input to the current layer, and “Parameters” refers to the parameters that need to be trained (excluding bias). All rotational convolution units are followed by a batchnorm [23] and ReLU nonlinearity with the exception of the final 1×1 conv layer which, has no nonlinearity and feeds into a softmax layer for classification. Counting rotational convolution and pooling layers as separate layers, the proposed method has 23 layers, maintaining a compact structure with only 570 k parameters.

Table 1. The network architecture; “R-Conv” refers to rotational convolution. Four convolutional kernels are grouped together, sharing parameters among them.

Type	Filter Shape	Input Size	Parameters
R-Conv	$5 \times 5 \times 3 \times 16$	$256 \times 256 \times 3$	300
Pooling	-	$128 \times 128 \times 16$	0
R-Conv	$5 \times 5 \times 16 \times 32$	$128 \times 128 \times 16$	3200
Pooling	-	$128 \times 128 \times 32$	0
R-Conv	$5 \times 5 \times 32 \times 32$	$64 \times 64 \times 32$	6400
R-Conv	$5 \times 5 \times 16 \times 32$	$64 \times 64 \times 32$	6400
R-Conv	$5 \times 5 \times 32 \times 32$	$64 \times 64 \times 32$	6400
R-Conv	$5 \times 5 \times 16 \times 32$	$64 \times 64 \times 32$	6400
Pooling	-	$64 \times 64 \times 32$	0
R-Conv	$5 \times 5 \times 32 \times 32$	$32 \times 32 \times 32$	6400
R-Conv	$5 \times 5 \times 16 \times 32$	$32 \times 32 \times 32$	6400
Pooling	-	$32 \times 32 \times 32$	0
R-Conv	$5 \times 5 \times 32 \times 32$	$16 \times 16 \times 32$	6400
R-Conv	$5 \times 5 \times 16 \times 32$	$16 \times 16 \times 32$	6400
R-Conv	$5 \times 5 \times 16 \times 32$	$16 \times 16 \times 32$	6400
Pooling	-	$16 \times 16 \times 32$	0
R-Conv	$5 \times 5 \times 32 \times 32$	$8 \times 8 \times 32$	6400
R-Conv	$5 \times 5 \times 16 \times 32$	$8 \times 8 \times 32$	6400
Pooling	-	$8 \times 8 \times 32$	0
Conv	$1 \times 1 \times 32 \times 32$	$4 \times 4 \times 32$	1024
Reshape	-	$4 \times 4 \times 16$	0
Conv	$1 \times 1 \times 512 \times 1000$	$1 \times 1 \times 512$	512,000
SoftMax	classification	1000	0

3.4. Object Detection Architecture

In the domain of security surveillance applications, the foremost application lies in object detection, with a particular emphasis on the detection of pedestrians and vehicles. The network described in this paper is also applicable to object detection tasks. Building on the network architecture discussed in the previous section, modifications are made to the YOLOv3-tiny [24] network to train models for the detection of vehicles and pedestrians. The architecture of the network is illustrated in Figure 2.

In the figure, R-conv refers to the rotational convolutional layer designed in the preceding section, conv denotes standard convolutional layers, pool represents the pooling layers, and up-sample indicates the up-sampling layers. Unlike the original YOLOv3-tiny network, this network employs a single-channel image input. The terms $n_1, n_2, \dots, n_8$ signify the number of channel outputs for each respective layer. W and H denote the dimensions of the input detection image, which, in this instance, is set to 352×288 . The annotations following the convolutional layers in the figure correspond to the sizes of the resulting feature maps (it is important to note that due to the design of the rotational convolutional layer, the values of n_1 to n_5 must be multiples of 4).

Both the detection and recognition models are trained using fp32 during the training process, and quantization and dequantization layers are added after each layer to enhance the model's accuracy. Moreover, for comparison purposes, we also train an fp32 model with the same network architecture.

4.3. Testing

To assess the efficacy of our models, the recognition model was benchmarked against MobileNetV1 [11], MobileNetV2 [12], ShuffleNetV1 [14], and ShuffleNetV2 [15]. We also evaluated the models under three conditions: utilizing IVE, without IVE (CPU INT8), and employing fp32 precision. The outcomes of these comparisons are detailed in Table 2. The inference framework utilized is NCNN, with the table's timing information reflecting the mean duration for 100 consecutive inferences. CPU utilization is determined by averaging the CPU usage (CPU) percentage over a 10 s interval. Performance comparisons were conducted on the HI3516EV200 (EV200) and MSC313E (313E) platforms.

For the detection model, we trained the yolov3-tiny model with the same datasets for comparison. Similarly to the recognition model tests, we compared the results and performance of models using IVE, without IVE, and using fp32. The comparison results are shown in Table 3.

Table 2. Performance on ImageNet; comparison for different networks.

NetWork	Top 1	Speed (EV200)	CPU (EV200)	Speed (313E)	CPU (313E)
MobileNet-V1	70.6	760	>99%	689	>99%
MobileNet-V2	72.0	476	>99%	438	>99%
ShuffleNet-V1	67.4	248	>99%	230	>99%
ShuffleNet-V2	69.4	242	>99%	224	>99%
Our (CPU FP32)	69.7	721	>99%	680	>99%
Our (CPU INT8)	69.4	924	>99%	814	>99%
Our (IVE INT8)	69.4	123	39%	145	47%

Table 3. Performance on detection datasets; comparison for different networks.

Network	mAP (Pedestrian)	mAP (Vehicle)	Speed (EV200)	CPU (EV200)	Speed (313E)	CPU (313E)
YOLOV3-Tiny	78.02	88.12	2540	>99%	2360	>99%
Our (CPU FP32)	77.87	87.91	672	>99%	651	>99%
Our (CPU INT8)	76.62	87.51	731	>99%	711	>99%
Our (IVE INT8)	76.12	87.04	52	57%	66	62%

Tables 2 and 3 illustrate that the recognition model optimized with IVE only marginally lags behind MobileNetV2 [12] by 2.6% in top1 accuracy, yet boasts an approximately 4-fold increase in processing speed. Notably, leveraging IVE resources cut the CPU usage by nearly half compared to solely CPU-dependent executions.

In the detection model sphere, our model showed a minor mAP decrement in pedestrian and vehicle detection datasets relative to Yolov3-Tiny. Owing to the memory bandwidth limits of the chosen chips, our model's operational speed is roughly 4 times that of Yolov3-Tiny, a result of significantly lower memory bandwidth demands. Post IVE algorithm optimization, the model reached processing speeds of 19 fps on Hisilicon HI3516EV200 and 15 fps on MSC313E, marking about a 91% speed increase over the base model and an 85% increase over models not optimized with IVE, all while maintaining a CPU usage under 70%, thus proving its high practical worth.

Moreover, testing revealed that executing quantized INT8 models with NCNN on these CPUs takes longer than FP32 models, likely due to NCNN's INT8 convolution implementations being memory-intensive.

By integrating the algorithm into cameras with HI3516EV200 and MSC313E chips and conducting over six months of testing across more than 100 cameras in scenarios like road micro-channel checkpoints and residential area access control, the system demonstrated excellent performance in vehicle and pedestrian detection snapshots. Tests on additional chips, including Novatek NT98562, MediaTek SSC335, Hisilicon 3516EV300, and Guoke Micro GK7205V300, affirmed the efficacy of the proposed model and optimization techniques across these platforms.

4.4. Ablation Study

This ablation study meticulously evaluates the distinct components of our proposed model to ascertain their individual and collective impact on performance within resource-constrained environments. The study elucidates the optimization achieved through our novel methodologies and design choices.

4.4.1. Width Multiplier: Scaling Model Complexity

While the base architecture is designed to be compact with low latency, specific use cases or applications may necessitate a smaller and faster model. To facilitate this, we adopt a width multiplier, denoted as α , from the MobileNet [11] to uniformly scale down the model across all layers. This width multiplier, α , ranging between 0 and 1, with typical settings of 1, 0.75, 0.5 and 0.25, is employed to reduce the number of input and output channels at each layer in a proportional manner. $\alpha = 1$ is the baseline network and $\alpha < 1$ are the reduced network. With the application of α , the number of input and output channels is further decreased, making the network more efficient.

In the context of the recognition model, models were trained across a spectrum of width multipliers, α , and subjected to comparative analysis. The results underscore that a decrement in α correlates with a reduction in model parameters and an increment in speed, concomitantly accompanied by a compromise in accuracy, notably so at $\alpha = 0.25$, which registered the most pronounced decline. Subsequent evaluations have ascertained that for diverse applications, either the baseline variant or the models configured with $\alpha = 0.75$ and $\alpha = 0.5$ represent optimal selections.

As shown in Figure 3, the speed-accuracy curves illustrate the trade-offs between different α values, providing a visual representation of the impact on performance. The curves indicate that while reduced α values enhance speed, they also result in lower accuracy, helping to identify the optimal balance for specific applications.

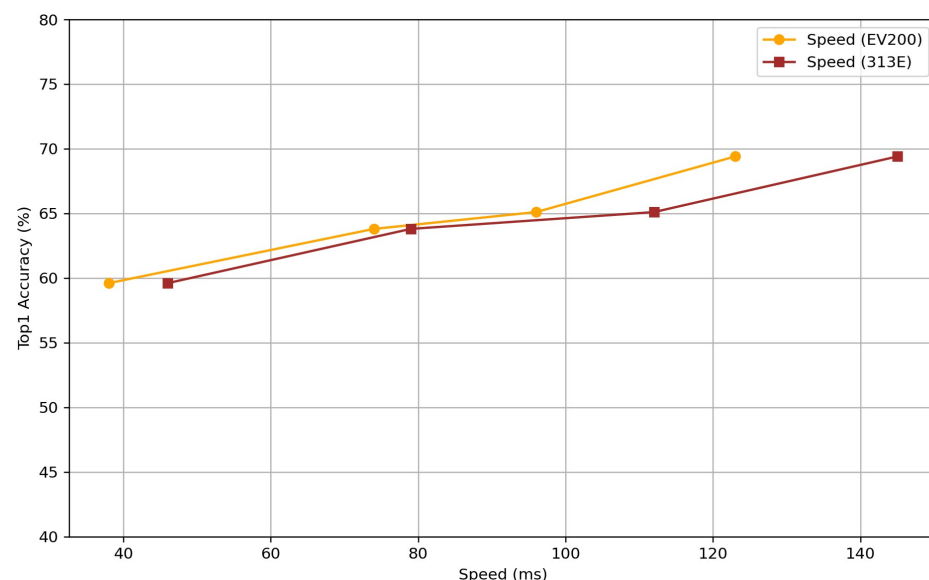


Figure 3. Accuracy vs. speed for different alpha values.

4.4.2. Effectiveness of the Rotating Convolutional Unit

At the heart of our model's efficiency is the "Rotating Convolutional Unit", which employs a single kernel, K_0 , that undergoes rotations to K_{90} , K_{180} and K_{270} . This design significantly reduces parameter count while ensuring comprehensive feature extraction from various orientations. Comparative analysis between models with and without this unit reveals that the incorporation of the rotating convolutional unit markedly decreases computational demand without detrimentally impacting accuracy, highlighting its critical role in the model's efficiency. Comparisons are shown in Figure 4. Several example structures are illustrated in Figure 1.

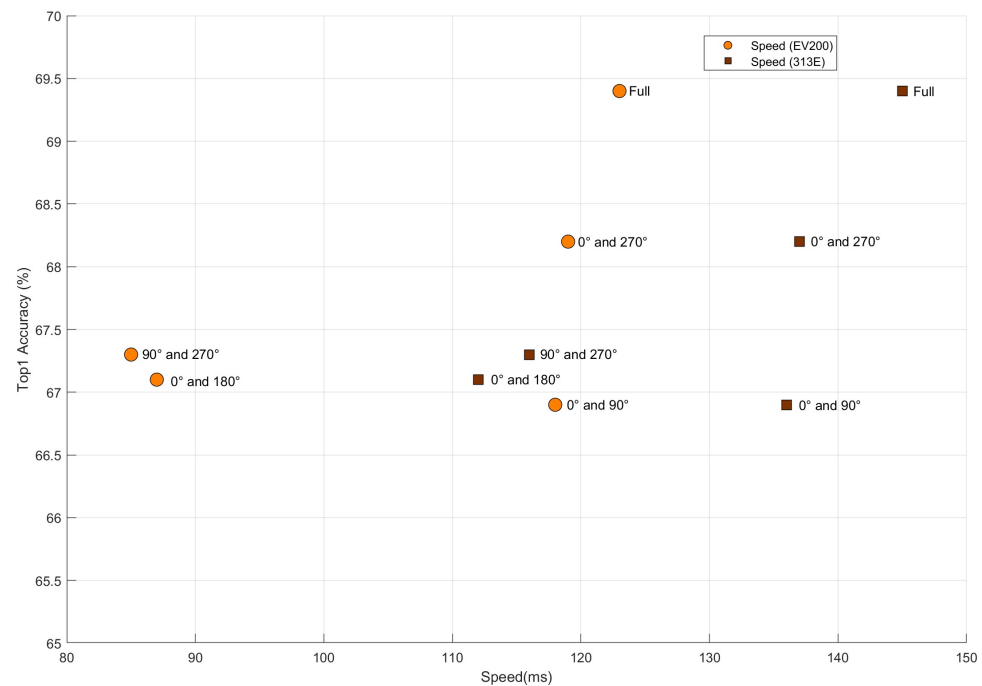


Figure 4. Different rotational convolutional unit architectures. "Full" refers to the utilization of convolutions in all four directions, as depicted in Figure 1c, while others signify the use of shortcut structures along with two types of convolutions, as shown in Figure 1a,b.

It is observed that choosing rotational convolutions at 0 and 180 degrees, and 90 and 270 degrees results in the fastest processing speed. This is attributed to the IVE operator's capability to compute the convolution in the transpose direction in a single calculation.

4.4.3. Backbone Comparisons for Detection Models

Focusing on object detection tasks, particularly pedestrian and vehicle detection, we assessed the efficacy of employing different backbone networks within our model. The optimized model, despite its streamlined architecture and reduced computational footprint, achieved comparable or superior performance in detection accuracy relative to more complex backbones. This outcome attests to our model's effectiveness in real-world surveillance applications, where both efficiency and accuracy are crucial.

As shown in Figures 5 and 6, The "Final Model" denotes the completed version of the detection model, whereas "Full-1.0", "Full-0.75", and "Full-0.5" indicate the employment of the recognition backbone network at different α scaling factors. Similar to the preceding section, configurations "0 and 180" and "90 and 270" incorporate only selected rotational convolutions. An examination of the test results reveals that the Final Model achieves enhanced performance in both pedestrian and vehicle detection scenarios. Although the alternative backbone networks also exhibit strong performance capabilities, they are

characterized by longer execution times, suggesting their potential utility as backbones in future, more complex object detection frameworks.

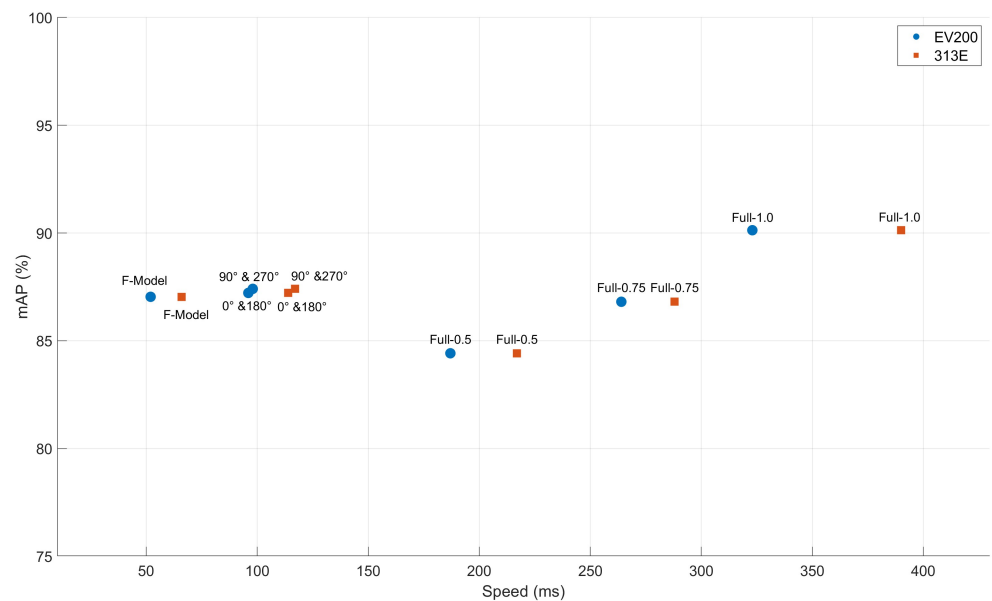


Figure 5. mAP (pedestrian) vs. speed.

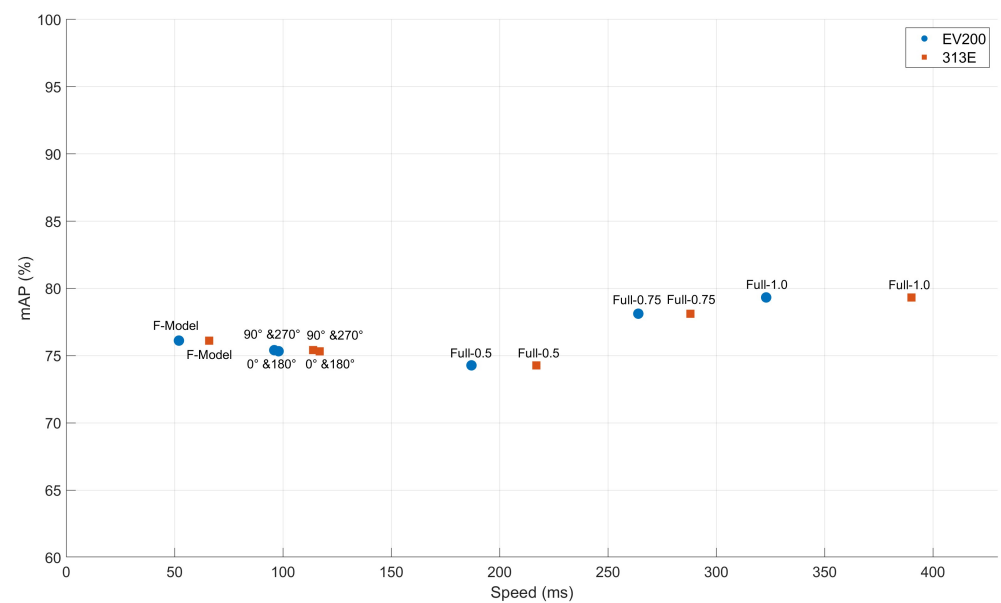


Figure 6. mAP (vehicle) vs. speed.

Through this ablation study, we established the indispensable contribution of the “Rotating Convolutional Unit” to the model’s efficiency, the strategic utility of the width multiplier, α , in tailoring model complexity, and the superiority of a full rotational convolutional approach in feature extraction. These components, individually and collectively, substantiate the model’s optimized performance for deployment on low-resource security devices, paving the way for advanced, efficient surveillance solutions in various applications.

5. Conclusions

This study presents a significant leap forward in the field of computational efficiency for convolutional neural networks (CNNs), particularly in the context of resource-constrained security devices. Our innovative “Rotating Convolutional Kernel” technique

marks a paradigm shift in how CNNs can be optimized to operate within the limited computational capacities of low-end CPUs, commonly found in security devices from Hisilicon, MediaTek, and Novatek.

The introduction of the “Rotating Convolutional Kernel”, coupled with our novel approach to pooling using max-filtering and DMA, not only addresses the pressing need for computational efficiency but also opens up new avenues for real-time, high-accuracy surveillance applications. Our optimized YOLOv3-Tiny model, which operates with remarkable efficiency on platforms like the HI3516EV200 and MSC313E chips, underscores the practical viability of our approach. Achieving processing speeds of 52 ms/frame and 66 ms/frame on these chips with only minimal compromises in accuracy, our model demonstrates that high-performance surveillance is attainable even on devices with stringent resource limitations.

The implications of our work extend beyond the immediate enhancements to security surveillance systems. By significantly reducing the computational demands of CNNs while maintaining high levels of accuracy, our approach paves the way for the broader application of advanced deep learning models in a myriad of contexts where computational resources are limited. This is particularly relevant in the era of the Internet of Things (IoT), where efficient data processing on edge devices is crucial.

Looking ahead, the methodologies developed in this study hold the promise for further refinement and application. The “Rotating Convolutional Kernel”, in particular, presents a versatile tool that can be adapted and extended to other neural network architectures and applications. Future research will explore these possibilities, focusing on enhancing the adaptability and efficiency of CNNs across a wider range of platforms and applications.

In conclusion, our study not only addresses a critical challenge in the deployment of intelligent surveillance systems but also contributes a set of tools and methodologies that have the potential to revolutionize the efficiency of neural networks in resource-constrained environments. As we continue to push the boundaries of what is computationally feasible on low-end devices, we pave the way for a new era of intelligent, accessible, and efficient computing at the edge.

Author Contributions: Methodology, F.W.; Validation, S.H. and J.G.; Writing—review & editing, D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported by the Shaanxi Province Key R&D Program Project (No. 2024GX-YBXM-507) and Innovation and Practical Ability Cultivation Program for Postgraduates of Xi’an Shiyou University (No. YCS23214248).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Smith, A.; Johnson, B. Trends in Security Surveillance: From Passive Defense to Active Warning Systems. *J. Secur. Technol.* **2021**, *34*, 145–159.
2. Wang, L.; Zhang, Y.; Chen, Z. Cloud-Edge Integration in Video Surveillance: Architectures and Applications. *IEEE Trans. Cloud Comput.* **2022**, *10*, 805–820.
3. Davis, M.; Kumar, R. Challenges and Opportunities for Intelligent Algorithms in Resource-Limited Security Devices. *Secur. Commun. Netw.* **2020**, *13*, 1572–1584.
4. Thompson, H.; Li, X. A Survey of Security Device Market Trends: The Dominance of Hisilicon, MediaTek, and Novatek. *Mark. Anal. Q.* **2019**, *6*, 234–247.
5. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Proc. Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [[CrossRef](#)]
6. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
7. Szegedy, C.; et al. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
8. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

9. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
10. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50× Fewer Parameters and <0.5 MB Model Size. *arXiv* **2016**, arXiv:1602.07360.
11. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Wey, T.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
12. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
13. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 17 October–2 November 2019; pp. 1314–1324.
14. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
15. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
16. Li, H.; Chen, Y. NCNN: A High-Performance Neural Network Inference Framework. *J. Comput. Sci. Technol.* **2019**, *14*, 254–266.
17. Wu, S.; Xu, L. MNN: A Universal and Efficient Inference Engine. *Mach. Learn. Appl.* **2020**, *3*, 33–45.
18. Yang, J.; Zhang, P. TNN: Towards Efficient Convolutional Operations in Neural Networks. *IEEE Access* **2021**, *9*, 123456–123467.
19. Choi, Y.; Kim, H. Model Compression Techniques for Efficient Convolutional Neural Networks. *J. Res.* **2021**, *45*, 765–778.
20. Patel, S.; Smith, J. Advancements in Model Efficiency for Edge Device Deployment. *J. Edge Comput.* **2021**, *2*, 134–145.
21. Kim, Y.; Lee, H. Efficient CNN Architectures for Low-Resource Environments. *Int. J. Comput. Vis.* **2022**, *120*, 640–655.
22. Zhao, T.; Wang, L. Deploying Deep Learning in Constrained Settings: Strategies and Innovations. *J. Deep. Learn. Technol.* **2023**, *5*, 47–62.
23. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
24. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
25. Redmon, J. Darknet: Open Source Neural Networks in C. 2013–2016. Available online: <https://github.com/pjreddie/darknet> (accessed on 18 July 2024).
26. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.F. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 248–255.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.