A Novel Fusion Pruning-processed Lightweight CNN for Local Object Recognition on Resource-Constrained Devices

Can Xie, Xiaobing Zhai, Haiyang Chi, Wenxiang Li, Xiaolin Li, Yuyang Sha, and Kefeng Li, *Member*, *IEEE*

***Abstract*—**In recent years, smartphones and smart educational products have become important catalysts for the development of consumer electronics. However, deploying large convolutional neural networks (CNNs) models on resource-constrained devices like smartphones is impractical due to the lack of high-performance central processing units (CPUs), graphics processing units (GPUs), and large-capacity memory. To address these challenges, we propose a new Fusion Pruning (FP) method aimed at compressing and accelerating large CNN models by leveraging L2 norm and equivalent transformation of the receptive field. To validate the feasibility of the proposed method in language education, we have developed an application deployed on smartphones. This application enables offline object recognition without the need for additional hardware platforms like Jetson Xavier NX. Additionally, we have also explored the performance of the FP method in the field of household robotics, obtaining satisfactory results. Experimental results demonstrate that the proposed method can accomplish tasks on resource-constrained devices.

***Index Terms*—Convolutional neural networks, Fusion pruning, Lightweight, Local object recognition, Mobile devices**

I. INTRODUCTION

CONVOLUTIONAL neural networks (CNNs) that are deployed on smartphones unlock a variety of novel applications empowered by deep learning, including object recognition-based second language learning [1]. With the rapid development of smartphones and broadband network technologies, the acquisition of multimedia information has become increasingly available [2] [3], resulting in a greater diversification of intelligent education formats. Among these formats, images represent the most intuitive and easily accessible learning materials [4] [5]. Intelligent recognition of image information by resource-constrained devices can be convenient for use in smart education systems. Directing learners to employ object recognition in real-world scenarios via smartphones, which then transforms these results into textual information and corresponding audio, stimulates greater interest in language learning and also enhances learning efficiency compared to traditional methods.

Most state-of-the-art CNNs have a large number of parameters and require extensive floating-point operations. Consequently, deploying large-scale CNNs necessitates multiple servers with GPUs, referred to as the "Education Cloud" in this paper. Learners connect to the education cloud via resource-constrained devices [6] such as smartphones, leveraging network signals (5G/Internet/Local net). They upload the captured raw images to the education cloud for recognition. As depicted in Fig. 1, transmitting a substantial volume of user images (Image 1, Image 2, ..., Image n) over communication networks is likely to result in congestion and increased transmission latency.

Although many object recognition applications based on typical lightweight CNN architectures [7] have been developed for use on mobile devices [8], these models still rely on broadband mobile networks to upload a large amount of image data to cloud servers for recognition, which consumes a significant amount of network resources. Furthermore, network latency can have a detrimental impact on user experience due to the following factors: (1) the transmission of large amounts of image data congests the network, resulting in bandwidth limitations, information loss, and a decrease in service quality; (2) wireless signals are susceptible to interference from weather conditions [9], obstacles, and electromagnetic sources, consequently leading to connection disruptions; (3) limited infrastructure and wireless network coverage restrict availability in certain geographical areas.

This paragraph of the first footnote will contain the date on which you submitted your paper for review, which is populated by IEEE. This work was supported by Macao Polytechnic University (RP/FCA-14/2023), The Youth Project of Science and Technology Research Program of Chongqing Education Commission of China (KJQN202103112) & The Science and Technology Research Program of Chongqing Education Commission of China (KJQN202003113). (*Corresponding author: Kefeng Li*). Can Xie, Xiaobing Zhai, and Haiyang Chi contributed equally as co-first authors.

Can Xie, Xiaobing Zhai, Haiyang Chi, Wenxiang Li, Yuyang Sha and Kefeng Li are with Faculty of Applied Science, Macao Polytechnic University, Macao SAR (e-mail: alexxiecan100@163.com; zhaixbwh@163.com; chy@kmu.edu.cn; 944033818@qq.com; syyshayuyang@163.com; kefengl@mpu.edu.mo)

Xiaolin Li is with Chongqing University of Posts and Telecommunications, Chongqing, China (e-mail: lixiaolin@cqupt.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org

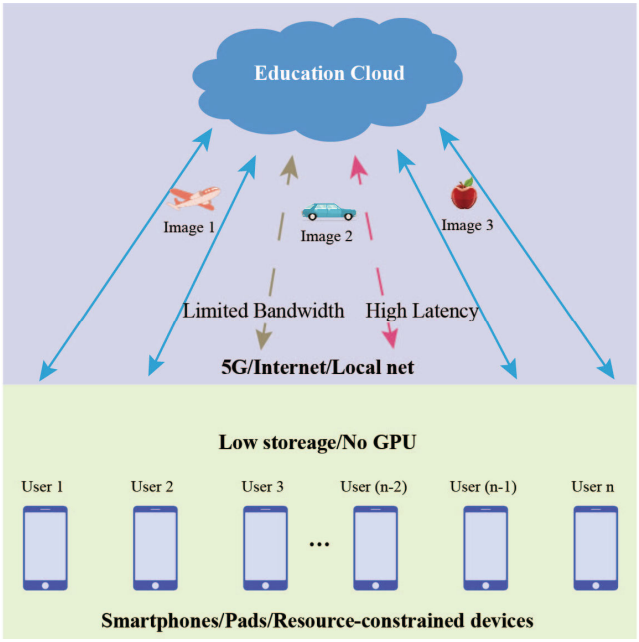

Fig. 1. Current image recognition APPs on smartphones rely on mobile networks to connect to cloud servers and request classification results remotely.

Object recognition is widely recognized by industry [10] as a topic that is important in the field of computer vision, and CNNs are effective at extracting image features for object recognition [11]**.** When implementing object recognition on mobile and embedded devices, the speed of neural network model inference is a crucial metric. YOLO FLNet [12] is a lightweight algorithm specifically designed for individual detection in open scenes that cleverly reduces model parameters and floating-point operations per second (FLOPs). However, there is an urgent need to address the application of target recognition technology in resource-constrained devices, to facilitate the integration of CNNs into intelligent educational systems.

This study addresses the aforementioned problem by propos ing a novel approach that includes removing redundant convol utional kernels and merging convolutional layers. This approac h reduces the model's parameter count and floating-point opera tions while minimizing the loss in accuracy. The resulting fina l model can be deployed on resource-constrained devices such as smartphones.

One effective approach for deploying CNNs on smartphones is to reduce the size of the model [13]. The classical model compression methods can be divided into four categories: parameter pruning and quantization, low-rank factorization, structure pruning, and knowledge distillation [14]. Parameter pruning and quantization include converting weights into binary values and compressing weight bit numbers through Huffman coding [15]. However, manual adjustments may not reliably filter out performance-sensitive parameters, potentially decreasing the model accuracy. Singular value decomposition (SVD) is a well-known low-rank factorization algorithm that distinguishes contributions by sorting singular values [16]. Nevertheless, matrix multiplication consumes a large amount of floating-point computing resources, and its approximation rules make the final evaluation challenging. Although structure pruning (specifically filter pruning methods) can reduce memory costs, it may encounter dimension matching issues in multibranch networks and does not offer a solution for reducing FLOPs [17]. The related knowledge distillation method includes training a compressed network with distillated knowledge from a large network, but only supports training models from scratch.

In this study, we introduce an innovative method denoted fusion pruning (FP) to address some of the challenges in deploying large CNNs on resource-constrained devices. The model processed using the FP method is compact, non-redundant, and efficient. This approach not only reduces the number of parameters but also enhances the inference speed. Our main contributions are as follows:

(1) The proposed approach differs significantly from traditional compression methods and compact models by introducing the novel utilization of L2 norm to prune redundant convolutional kernels. This technique effectively reduces the number of parameters without significantly compromising the recognition accuracy of the model.

(2) This study found a strong correlation between the number of convolutional layers and the model's FLOPs. Therefore, the utilization of receptive field equivalency for convolutional layer fusion to reduce FLOPs can enhance the inference speed of the model.

(3) Traditional methods typically only address model size or computational speed. The proposed approach, following the aforementioned two-step strategy, enables optimization in both model size and computational speed. We designed algorithmic experiments based on publicly available datasets and subsequently developed a simple app to simulate its application in offline object recognition in the field of smart education. The results indicate that the proposed method exhibits satisfactory performance in both algorithmic aspects and practical applications.

(4) The experimental deployment of smartphones indicates that the proposed method does not require high-performance AI servers or specialized hardware platforms like Jetson Xavier NX to perform recognition tasks. Local model recognition also eliminates the necessity of utilizing communication network resources. This discovery provides new insights into the application research of neural networks in areas such as wearable devices, drones, and autonomous driving.

(5) To validate the performance of our theoretical findings, we developed a simple mobile APP to deploy our model. The experimental results demonstrate that the APP's size and the inference speed of our method are well-suited for resource-constrained devices, offering new insights into the deployment and application of neural networks on such devices.

TABLE I and TABLE II list the mentioned abbreviations in alphabetical order and the used notations, respectively.

TABLE I
LIST OF ABBREVIATIONS (ALPHABETIC)

Abbreviation	Meaning
APP	Application
BN	Batch Normalization
CIFAR-10	Canadian Institute for Advanced Research-10
CIFAR-100	Canadian Institute for Advanced Research-100
CNNs	Convolutional Neural Networks
CPUs	Central Processing Units
FP	Fusion Pruning method
FLOPs	Floating-point Operations Per Second
GPUs	Graphics Processing Units
LR	Learning Rate
PTQ	Post-Training Quantification
QAT	Quantitative Perception Training
SVD	Singular Value Decomposition
VGG-16	Visual Geometry Group-16
VGG-19	Visual Geometry Group-19

TABLE II
SUMMARY OF THE NOTATION ADOPTED IN THE TEXT

Notation	Meaning
W	Italic capital letters denote matrices
w	Lowercase letters represent vectors
$\|\| \bullet \|\|$	L2 norm
$U\Sigma V^T$	Multiplication operation of three matrices U, Σ, V^T
I	Italic capital letter "I" represents identity matrix
$diag(\bullet)$	It represents the construction of diagonal matrices
$min(\bullet)$	The min operation represents finding the minimum value
$\frac{\partial J}{\partial_{W^{[l]}}}$	It represents taking the partial derivative of the weight w
RF_l	RF represents the receptive field of the l-th layer

II. RELATED WORK

A. Parameter Pruning and Quantization

The most intuitive approach for model simplification and acceleration is to eliminate redundant parameters and quantify the datatype. Parameter pruning removes parameters that are not important for CNN learning. Pruning algorithms can be categorized into two methods: static and dynamic methods. Static pruning occurs during non-training processes, while dynamic pruning takes place in real time during program execution.

Static pruning [18] refers to the pruning process that occurs before inference (rather than after training). After each model training, all the parameters and network results are saved as a file. The program trims the saved results to generate a new model, using the last saved parameters as the initialization parameters for the new training process. Mainstream static pruning algorithms typically include three steps: (1) fully train a model and select redundant parameters, (2) prune the redundant parameters, and (3) retrain a new model [19]. Retraining is an essential component of static pruning. The accuracy of the pruned model decreases, but retraining can compensate for this loss of accuracy. However, in a static pruning process, if the parameter selection is inappropriate, useful information will be deleted, and this change will continue. Even after retraining, the model's generalization ability is affected [20]. In contrast, dynamic pruning does not have a retraining stage. Before training, it is necessary to determine the importance of neurons, channels, and layers and select important parameters for training to avoid the limitations of static pruning. Previous studies have shown that dynamic pruning requires additional learnable connections during network training, objectively increasing the initial network's structural complexity [21]. Additionally, the extent of such dynamic pruning [22] affects the related hardware design. Moreover, dynamic pruning also needs to be completed during the training process, which inevitably consumes additional storage memory and computational resources and increases the complexity of algorithm design and hardware cost. Finally, this mechanism lacks a loss compensation mechanism, thereby affecting the final generalization accuracy.

Quantization [23] is commonly utilized for weight quantization. Han [19] comprehensively employed quantization, clustering, and sharing techniques to process weights, reducing the storage capacity from 27 MB to 6.9 MB. Post-training quantization involves quantifying the weights using the trained model and subsequently reoptimizing the model [24]. Kullback–Leibler divergence (also known as Kullback–Leibler divergence or information divergence) was used to accelerate the network without sacrificing accuracy for various established models. Quantization technology is typically divided into two directions: 1) quantitative perception training (QAT) and 2) post-training quantification (PTQ). Sun [25] proposed a method to train a comprehensive quantization model that supports diverse bit widths (e.g., from 8 bits to 1 bit) to accommodate online quantization and bit width adjustment. Nagel [26] introduced state-of-the-art algorithms to mitigate the impact of quantization noise on network performance while maintaining low-bit weights and activations. Recent research [27] endeavors have delved into the realm of extreme quantization techniques, transforming floating-point arithmetic into binary gates, a paradigm shift that has been documented in [28]. Typically, the utilization of fewer than 32 bits often results in notable performance deterioration. A prevalent approach to mitigate this issue involves substituting floating-point operations with their integer counterparts, often necessitating an additional training phase to compensate for the subsequent reduction in accuracy [29]. This underscores the intricate interplay between quantization methods, which indeed facilitate parameter reduction, and the more intricate training strategies they inherently rely upon to maintain performance standards.

B. Low-Rank Factorization

Research on low-rank factorization has focused on simplifying convolutional layers [30]. SVD serves as a crucial method for decomposition. SVD expresses any $m \times n$ matrix as the product of three matrices: an m-rank orthogonal matrix, an $m \times n$ rectangular diagonal matrix containing nonnegative

diagonal elements arranged in descending order, and an n-rank orthogonal matrix. It can be expressed by the following formula:

$$A = U\Sigma V^T \tag{1}$$

A can be converted into the product form of U, Σ, V^T, and only Σ is taken to approximate A, thus simplifying A.

$$I = UU^T \tag{2}$$

$$I = VV^T \tag{3}$$

The column vector U is called the left singular vector, and the column vector V is called the right singular vector.

$$\Sigma = diag\,(\sigma_1, \sigma_2, \ldots, \sigma_p) \tag{4}$$

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq 0 \tag{5}$$

σ_p is the singular value of matrix A.

$$p = min\,(m, n) \tag{6}$$

Low-rank factorization includes using SVD to approximate matrices with a low-rank characteristic matrix, which facilitates network compression [31]. SVD is particularly effective for the fully connected layer, providing a compression ratio of ~3×. However, this technique provides a hierarchical approximation, limiting its ability to perform global parameter compression. The lack of similarity between the logic of different layers makes it challenging for the model to converge. Moreover, te determination of the optimal rank for low-rank approximation remains a challenge, requiring an iterative approach through trial and error. The model compression ratio is dependent on the rank approximation and the tolerable level of performance degradation.

C. Structure Pruning

Structure pruning, especially channel pruning, is an effective model compression technique. This technique simplifies the model by removing redundant channels and reducing the size of the feature maps. However, channel selection poses challenges because complex algorithms that can distinguish the correlation between channels and output feature maps must be designed to ensure proper pruning of channels with different contributions. Luo [32] have attempted to minimize errors in discrete cases by making the parameters more continuous. However, this subjective variation lacks a rational mathematical foundation and cannot accurately determine the actual values of channels with smaller contributions. Other approaches [33] have proposed differentiating the contributions based on the importance of channels, but these importance values may change with variations in datasets or training strategies, leading to decreased model accuracy. Recently, researchers have introduced perceptual pruning techniques [34] to adaptively explore channels. This method adds additional classifiers to the intermediate layers of a CNN to achieve perception. However, this approach increases the complexity of designing intermediate layers and prunes the overall connections between layers, which may result in reduced model accuracy.

D. Knowledge Distillation

Knowledge distillation is a research field within model compression that combines the performance and generalization capabilities of large models with the scale and limited expression features of small models. By leveraging knowledge from a large model to train a smaller model, a small model can achieve comparable performance while reducing the number of parameters while maintaining accuracy [35]. Hinton et al. [36] proposed a method of training a complex network model and then using the output of the complex network along with the real labels of the data to train a small neural network. Therein, the knowledge distillation framework typically involves a complex model, known as the "Teacher" model, and a small model, referred to as the "Student" model.

However, the success of knowledge distillation relies heavily on the quality and capacity of the larger "Teacher" model. If the "Teacher" model is not well trained or lacks sufficient complexity, it may not effectively transfer knowledge to the "Student" model. Additionally, during the transfer process, certain intricate details may be lost during the knowledge extraction, leading to slight performance degradation compared to the original large-scale model.

Furthermore, the network necessitates at least two rounds of training, resulting in a prolonged training period. The knowledge distillation process requires additional computational resources and time to simultaneously train both the "Teacher" and "Student" models, which can result in lower efficiency in resource-constrained environments.

E. Neural architecture search

Recently, neural architecture search (NAS) [37] has emerged as a method for optimizing model structures. Unlike traditional handcrafted neural network architectures, which require manual selection and definition of layer operations, NAS automates the generation of the most suitable network architecture for a specific task from a set of available operations. The process begins with a search phase that explores all possible architectures within the search space. The highest-performing architecture is then identified and retrained from scratch.

NAS algorithms primarily involve three components: the search space, search strategies, and evaluation metrics. The search space is an extensive collection of potential neural network architectures that NAS algorithms can explore and optimize. It includes a wide variety of layer types—such as convolutional, fully connected, and classifier layers—as well as interconnection patterns and parameter initialization methods. Typically, the search space is conceptualized as a directed weighted graph. In this graph, nodes represent distinct neural network architectures, and edges indicate the operations that allow the transition from one architecture to another.

The search strategy in NAS dictates the approach used to navigate the search space and identify the optimal neural network architecture. It encompasses various popular methods, including random search, Bayesian optimization, genetic algorithms, reinforcement learning, and gradient-based techniques. Each method presents unique benefits and challenges regarding efficiency, effectiveness, and scalability.

NAS algorithms often demonstrate superior performance in

image classification tasks compared to object detection and image segmentation. This superiority may stem from the complexity of architectures for image classification, which typically requires a more exhaustive exploration of the search space. However, this specialization can limit the broader application of NAS to other vision tasks. Additionally, within image classification, NAS's generation of resource-efficient architectures, for which it is well-suited, faces constraints such as computational resources, search space design, and evaluation criteria.

Overall, NAS must carefully balance the complexity of the search space, the efficiency of the search strategy, and the rigor of the evaluation metrics to ensure optimal performance across various vision tasks while adhering to practical constraints.

III. PROPOSED METHOD

Traditional approaches may encounter the following challenges when it comes to pruning redundant parameters and accelerating inference:

(1) Inaccurate identification of the contribution of convolutional kernels to feature extraction using heuristic or statistic-based approaches, leading to inadvertent removal of useful information.

(2) The necessity to train auxiliary large models to ensure proper learning of knowledge by the network, which can result in additional computation and storage overhead. Moreover, training large models requires more time and resources, thereby increasing the algorithmic complexity.

(3) Traditional methods often handle parameter pruning and acceleration separately, without simultaneously considering both aspects. This lack of simultaneous consideration fails to exploit the potential synergistic effects of compression and acceleration.

We propose a new fusion pruning method to address the key challenges in deploying large CNNs on embedded devices.

Specifically, we use the L2 norm of the scaling factor γ in batch normalization (BN) layers [38] to identify important channels for pre-training and partial derivative calculations. For the remaining convolutional layers, fusion is achieved by performing equivalent transformations on receptive fields, aiming to reduce both the number of redundant convolutional kernels and the number of convolutional layers. See Fig. 2.

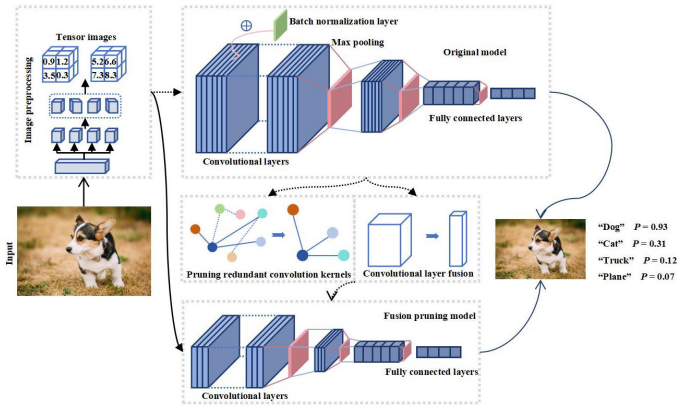

Fig. 2. The schematic diagram of the FP method.

We append a BN layer to the convolution layer, and the BN layer includes the per-channel parameter γ. Therefore, no additional information needs to be inserted, avoiding redundant computational overhead in the model. Simultaneously, an L2 norm is applied to γ and incorporated into the model training process. During the gradient calculation phase, the partial derivative of γ is computed and used as a change value for updating γ, resulting in its gradual decrease. As the model iterates through the specified number of learning epochs, different values of γ correspond to the importance values of the convolutional kernels. Then, it becomes possible to remove the less important kernels. The remaining kernels are considered "elite" because they are sensitive to feature extraction. This method continues to employ the principle of equivalent receptive field transformation to merge convolutional layers, achieving a reduction in FLOPs.

A. Formulation

To identify redundant filters for pruning, we designed an index (γ) to indicate the relative importance of each convolutional kernel (which is denoted as a filter). These filters can then be removed according to the index without sacrificing accuracy. Accurate representation of the filter importance ensures the quality of model pruning. We first perform regularization using the L2 norm to avoid overfitting. By leveraging the characteristics of the L2-norm, we then obtained an attenuation effect that enables smooth changes in γ during the training process, with the final value indicating the importance of the corresponding filter. The following is the formula for the L2 norm:

When $W \in R^n$,

$$W\ (w^{[1]},w^{[2]},\ldots,w^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} w^i + \frac{\lambda}{2m}\sum_{l=1}^{L}\left|\left|w^{[l]}\right|\right|^2 \quad (7)$$

where W represents the weight of the CNN and is used to represent the γ of the BN layer and λ in (7) is a hyperparameter used to adjust the coefficient of the relationship between the two sides of the plus sign. L represents the L-th layer of the CNN.

The norm expansion form of the L-th layer weight w is represented as follows:

$$\|w^{[l]}\|^2 = \sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}}\left(w_{ij}^{[l]}\right)^2, \quad w^{[l]}:\left(n^{[l]},n^{[l-1]}\right) \quad (8)$$

According to (8), we can calculate the gradient in the back-propagation stage (9):

$$dw^{[l]} = \frac{\partial_J}{\partial_{w^{[l]}}} + \frac{\lambda}{m}w^{[l]} \quad (9)$$

Finally, the gradient is used to update w:

$$\hat{w}^{[l]} = w^{[l]} - \alpha\frac{\lambda}{m}w^{[l]} - \alpha\frac{\partial_J}{\partial_{w^{[l]}}} \quad (10)$$

$$\hat{w}^{[l]} = \left(1 - \alpha\frac{\lambda}{m}\right)w^{[l]} - \alpha\frac{\partial_W}{\partial_{w^{[l]}}} \quad (11)$$

In (10) and (11), α denotes the learning rate (LR). Derivation of these formulas shows that the weight gradually decreases upon subtracting the gradient of the learning rate from the original parameter w. This subtraction results in increasingly

smaller weights over time. We thus select filters with lower importance based on the corresponding channel weight. By leveraging the natural attenuation achieved by learning and backpropagation, the weight can accurately reflect the natural feature distribution in the image information. This process enables the application program to prune redundant information without compromising accuracy. Our approach uniquely leverages the L2 norm beyond regularization. We used this approach to prune convolutional kernels by labeling importance and removing filters with minimal impact on output accuracy. Compared to other pruning techniques, this pruning method does not require sparse libraries or specialized hardware [39].

B. Batch Normalization Layer

The BN layer is known to be important for the optimization of CNNs [40]. In this study, we introduce a BN layer after each convolutional layer to normalize images within a batch. This normalization technique effectively reduces image variations and enhances the convergence speed of the model. The γ parameters in these BN layers are determined using (9), (10), and (11) to identify the significance of the corresponding convolutional kernels. By leveraging this information, we accurately identify and prune less important kernels during pruning.

C. Receptive Field

After accomplishing the task of pruning redundant filters using the L2 norm, the proposed method aims to reduce the number of floating-point computations in the model to enhance inference speed. We carried out receptive field equivalence transformation to fuse convolutional layers and reduce the model's FLOPs.

(1) FLOPs

To provide a more comprehensive explanation of the proposed method, we present a concise overview of FLOPs, which serve as a metric for quantifying the computational efficiency of model inference. Deep neural networks improve performance by augmenting the number of hidden layers; however, this approach concurrently increases the number of FLOPs, resulting in a decline in inference speed. Drawing on the distinctive properties of convolutional operations, FLOPs can be defined as follows:

$$FLOPs = 2 \times C_i \times k^2 \times C_o \times W \times H \quad (12)$$

$$FLOPs = C_i \times k^2 \times C_o \times W \times H \quad (13)$$

where C_i denotes the number of input channels, C_o represents the number of output channels, $C_o \times W \times H$ denotes all features of the feature map, and k represents the length and width of the convolution kernel. Equations (12) and (13) differ in that (13) represents the sum of multiplication and addition operations, referred to as multiply accumulated operations (MACs).

The FLOPs for each convolution kernel are calculated as above, and for n kernels, the total becomes $n \times FLOPs$. Conversely, by reducing the MACs through a decrease in the number of convolution kernels, the inference speed of the model can be significantly accelerated.

(2) Equivalent Transformation of the Receptive Field

Building upon [41], we propose a method for fusing convolutional layers through receptive field equivalent transformation. The receptive field refers to the size of the pixel region on the original input image, which is mapped onto the feature maps outputted by each convolutional layer in the CNN. Typically, convolutional kernels of different sizes have different receptive field sizes. We have observed that convolutional layers with the same receptive field size may vary in quantity. By leveraging this characteristic, we can fuse convolutional layers and thereby reduce the FLOPs. The use of receptive field equivalent transformation significantly reduces the multiplications and additions between the convolution kernels and the image, without compromising the integrity of feature map.

Calculation of the receptive field size:

$$RF_l = RF_{l-1} + (k_l - 1) \times \prod_{i=0}^{l-1} s_i \quad (14)$$

where RF_{l-1} is the size of the receptive field in layer l - 1, k_l is the size of the convolution kernel in layer l, and s_i is the convolution stride in layer i.

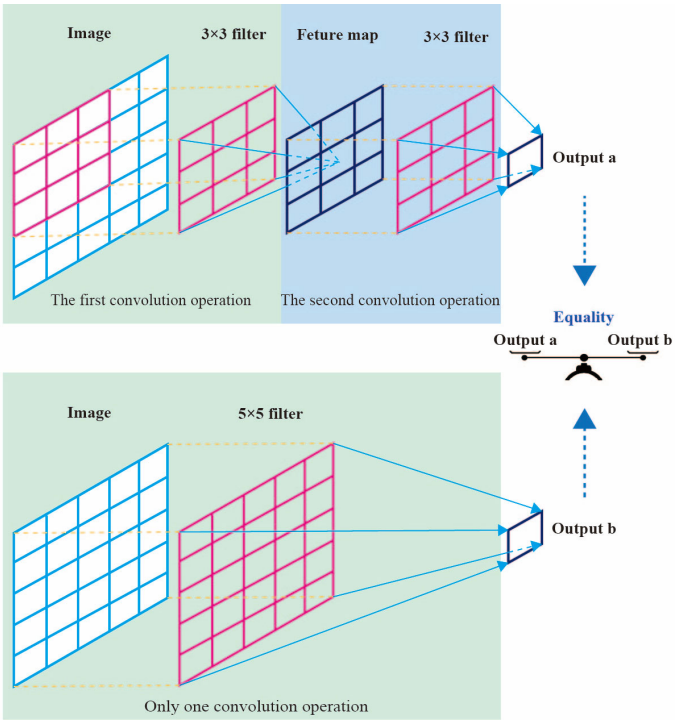

Fig. 3. The receptive field equivalent transformation. Filter in the Fig. 3 is the convolutional kernel.

As shown in **Fig. 3**, two consecutive convolution operations are performed on the image using two 3×3 filters. The size of the generated feature map is the same as the result obtained from one convolution operation using a 5×5 filter. This result indicates that the receptive field size of the two convolutional layers is equal. By utilizing this equivalence, we can merge the two 3×3 layers into one 5×5 convolutional layer. This fusion operation reduces the FLOPs in the model and improves the inference speed without significantly affecting the model's

accuracy.

D. Algorithms

We describe the fusion pruning algorithm as follows:

Algorithm The fusion pruning method.

Input: Training images $I=\{i^1,i^2,...,i^n\}$, $n\in Z$, Hyper-parameter λ, number of train_epochs, learning rate $\tau=0.1$, batch_size.

1: **For** epochs **in** train_epochs
  If epochs in [0.5 × train_epochs, 0.75 × train_epochs]
    $\tau\times 0.1$
  End If
  train the model
  backward process
  If $\lambda\neq 0$
    $\hat{\gamma}=\gamma$ add partial derivative of L2-norm of γ
  End If
  Prune the redundant filter
  Calculate the number of parameters, FLOPs, accuracy and loss
  Save model and log files
  End for
2: **Load** model
  If flag
  rewrite the convolution layer of model
    If there are two consecutive 3 × 3 filters
    replace with a 5 × 5 filter
    End If
  Save new_model
  End If
3: Re-train (new model)
  Save log files
End program

IV. EXPERIMENTS

To evaluate the efficacy of our FP method, we performed experiments on different datasets and compared the results to those of classical models. We used VGG-19 as a benchmark model (baseline) and selected several mainstream models for further comparison. Key performance metrics included training accuracy, training loss, test accuracy, test loss, accuracy and number of FLOPs. Our primary datasets were CIFAR-10 and CIFAR-100, which were generated using a Python framework.

A. Experimental Environment

Datasets. We conducted experiments using two benchmark datasets: CIFAR-10 and CIFAR-100 [42], which are commonly used for image classification tasks.

Data augmentation. The pre-processing of data is crucial to the experimental results. To ensure the reproducibility of the experiment, we have disclosed the process of pre-processing the original image, as shown in Fig. 4:

1. Padding all sides of the image, padding = 4.
2. Perform random cropping on the padded image, with a size of (32, 32).
3. Randomly horizontally flip the image with a given probability. (Probability = 0.5)
4. Convert the image into a tensor image.
5. Normalize the tensor image using the mean and standard deviation, with mean values of (0.4914, 0.4822, 0.4465) and standard deviations of (0.2023, 0.1994, 0.2010).

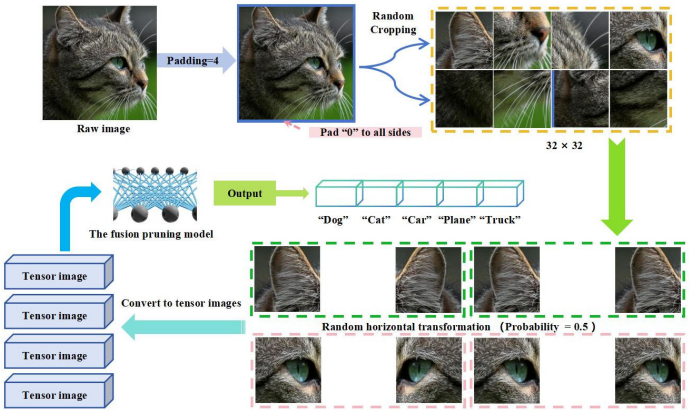

Fig. 4. The flowchart of data argumentation

Hyper-parameters and activation function. In the experiment, the hyper-parameter λ is set to 1×10^{-4}, the momentum is set to 0.9, the initial learning rate (LR) is 0.1, the batch size is set to 64, and the incipient value of epoch is 160. The mish[43] is used as the activation function for the model.

The learning rate decay strategy for epochs is as follows:

- ✓ Epochs ∈ (0,79), LR = 0.1.
- ✓ Epochs ∈ (80,119), LR = 0.01.
- ✓ Epochs ∈ (120,159), LR = 0.001

Hardware platform. The configuration of the hardware platform is as follows: an I5 CPU, 32 GB of RAM, and an RTX3060 12G GPU.

Software environment. The experiment was implemented in Python 3.11.8 and compiled using Anaconda V3 as the platform. The software platform used was Pytorch 2.2.2. Furthermore, we develop the smartphone APP using Android Studio Giraffe (Runtime version: 17.0.6).

B. Comparative analysis on the CIFAR-10 dataset

We applied the FP method to compress and accelerate the baseline model and compared the performance with that of other mainstream models on the CIFAR-10 dataset.

Fig. 5a and 5b curves over training epochs for the baseline model. The loss exhibits substantial fluctuations for the first 80 epochs due to the initial high learning rate. After reducing the learning rate at epoch 80, the loss curve becomes smoother. The accuracy curve corresponds to the loss pattern, with initial jitter followed by a smoother increase after epoch 80. Compared to the baseline, the FP method achieves a lower initial training loss, indicating improved training performance by eliminating redundancies and noise (Fig. 5c). Similarly, the FP accuracy curve was consistent with its loss curve and converged (Fig. 5d). Fig. 5e compares the training and test losses for the baseline and the FP methods. The FP training loss decreases faster than the baseline. The FP and baseline test losses are consistent, with the FP starting from a lower initial value. The accuracy curves coincide, indicating that the FP method preserves model accuracy (Fig. 5f).

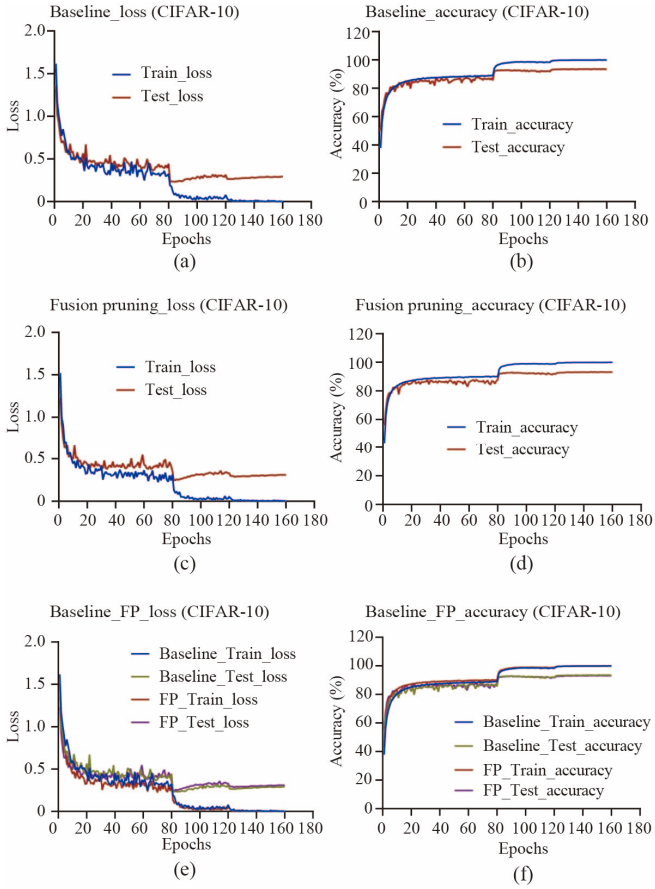

Fig. 5. Performance comparison between the baseline (VGG-19) and fusion pruning (FP) methods using the CIFAR-10 dataset. **(a).** Baseline loss curve. **(b).** Baseline accuracy curve. **(c).** Loss curve of the FP method. **(d).** FP method's accuracy curve. **(e).** Comparative analysis between baseline and FP loss. **(f).** Comparative accuracy analysis between the baseline and FP methods

Furthermore, our FP method achieved a significant balance between parameter reduction and model acceleration without significantly affecting accuracy. This balance greatly improved the overall performance of the cellular neural network model (TABLE III). Compared to the baseline model, our method achieves an accuracy of 92.78%, utilizing only 8.1% of the baseline model's parameters and 37.8% of its FLOPs. The accuracy decrease is only 0.5% compared to that of the baseline model. The slimming network is currently an excellent method that strikes a balance between accuracy, parameters, and FLOPs. However, in comparison to the slimming network, the FP method reduces the number of parameters by 29.1% and the number of FLOPs by 22.8%, albeit with a slight decrease in accuracy of 1%. This finding indicates that when deploying the model on smartphones, it is possible to quickly recognize images using fewer parameters.

TABLE III
COMPARISON OF FUSION PRUNING (FP) AND MAINSTREAM MODELS ON THE CIFAR-10 DATASET

Method	Parameters (M)	FLOPs (×10⁸)	Accuracy (%)	Precision (%)	Recall (%)	F measure (%)
VGG-19	20.04	7.98	93.28	-	-	-
Slimming network	2.30	3.91	93.78	-	-	-
VGGNet (L1)	5.40	2.06	93.40	-	-	-
VGGNet-19 (DFNP)	4.73	2.60	92.55	-	-	-
MobileNet-v1_G2	17.29	2.43	93.09	-	-	-
MobileNet-v1_G4	10.23	1.44	93.14	-	-	-
Densenet161	-	-	93	93	93	93
Googlenet	-	-	87	87	87	87
VGG-16	14.71	3.13	94	94	94	94
GhostNet 1.5×	7.80	3.12	92.99	91.79	92.99	92.16
MobileNet-v2 1×	2.24	3.19	93.91	92.12	92.48	91.90
ShuffleNet-v2 1.5×	2.49	3.03	92.13	91.18	92.13	91.22
Res50-Top10	-	-	92	92	92	92
Res101-Top10	-	-	93	99	99	99
Our FP	**1.63**	**3.02**	**92.78**	**93.26**	**93.24**	**93.24**

C. Comparative analysis on the CIFAR-100 dataset

Fig. 6 shows the loss and accuracy curves of the baseline (Fig. 6a and 6b) and the FP method (Fig. 6c and 6d) on the CIFAR-100 dataset. Fig. 6e and 6f compare the loss and accuracy between the baseline model and the FP method during training. The aligned curves suggest that the FP method did not significantly impact the learning or generalization ability of the baseline model.

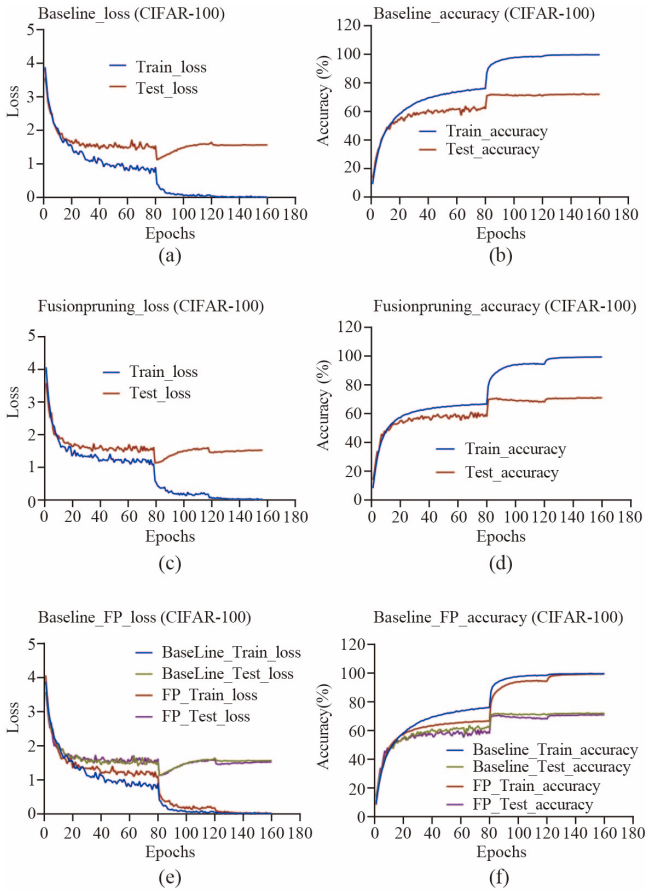

Fig. 6. Performance comparison between the baseline and fusion pruning (FP) methods using the CIFAR-100 dataset. **(a).** Baseline loss curve. **(b).** Baseline accuracy curve. **(c).** Loss curve of the FP method. **(d).** FP method's accuracy curve. **(e).** Comparative analysis between baseline and FP loss. **(f).** Comparative analysis between the baseline and FP accuracy.

TABLE IV compared our method with other mainstream compression models. Compared to the slimming network, our TOP-1 accuracy only decreased by 2.17%, while parameters

decreased by 31% and FLOPs decreased by 17.4%. Compared to other methods, our approach achieved greater improvement in TOP-1 accuracy, number of parameters and number of FLOPs. These results highlighted our method's balanced performance in terms of accuracy, parameters, and FLOPs on the CIFAR-100 dataset.

TABLE IV
COMPARISON OF FUSION PRUNING (FP) AND MAINSTREAM MODELS ON THE CIFAR-100 DATASET

Method	Parameters (M)	FLOPs ($\times 10^8$)	Accuracy (%)	Precision (%)	Recall (%)	F measure (%)
VGG-19	20.08	7.98	72.24	-	-	-
Slimming network	5.00	5.01	73.32	-	-	-
DeiT-Ti with CLS	5.72	12.50	64.49	-	-	-
DeiT-Ti without CLS	5.72	12.50	65.93	-	-	-
HVT-S (Conv)	23.54	14.70	69.75	-	-	-
MobileNet-v1_G2	17.38	2.43	72.05	-	-	-
MobileNet-v1_G4	10.32	1.44	72.24	-	-	-
VGG-16 Base	14.71	3.13	72.60	72.60	72.60	72.55
MobileNet-v2	-	-	68.93	68.93	68.93	68.80
Res50-Top10	-	-	74	76	72	74
Res101-Top10	-	-	75	77	73	75
Our FP	**3.45**	**4.14**	**71.15**	**72.24**	**71.73**	**71.83**

D. Comparative analysis on the ImageNet dataset

We conducted experiments on the ImageNet dataset to compare the performance of our method with that of mainstream compact models (Table V). This study explores the possibility of deploying large CNNs on resource-constrained devices, such as smartphones and household robots. The objective is to reduce the number of parameters and FLOPs without significantly compromising the model's inference accuracy. We selected results from mainstream compact models on the ImageNet dataset for comparative analysis and found that our method is competitive in terms of performance on large datasets.

TABLE V
COMPARISON OF FUSION PRUNING (FP) AND MAINSTREAM MODELS ON THE ImageNet DATASET

Method	Accuracy (%)	Precision (%)	Recall (%)	F measures (%)
SqueezeNet	57.64	-	-	-
AlexNet	58.90	-	-	-
MobileNet	69.37	-	-	-
Inception	59.00	61	60	60
Xception	69.90	-	-	-
Our FP	**70.00**	**60**	**62**	**60**

V. DEPLOYMENT ON MOBILE DEVICES

A. Mobile APP Function Description

To further validate the performance of the FP compressed and accelerated model in actual APPs, we developed a simple app through an android studio. Two mobile phones and one tablet were selected as hardware testing platforms for this experiment, and the experimental results are shown in TABLE IX and TABLE X.

The APP for this study was developed using the latest version of Android Studio and programmed in Java. The mobile hardware platform is illustrated in TABLE VI.

TABLE VI
OVERVIEW OF RESOURCES AVAILABLE IN SMARTPHONES

Device	CPU	RAM	Android
OPPO Realme	Qualcomm Snapdragon 778G	8 Gigabytes	V 13
VIVO T1	Qualcomm Snapdragon 778G	8 Gigabytes	V 11
HONOR PAD 6	HUA WEI Kirin 710A	4 Gigabytes	V 10

The APP integrates FP-processed lightweight CNNs for local object recognition (Fig. 7).

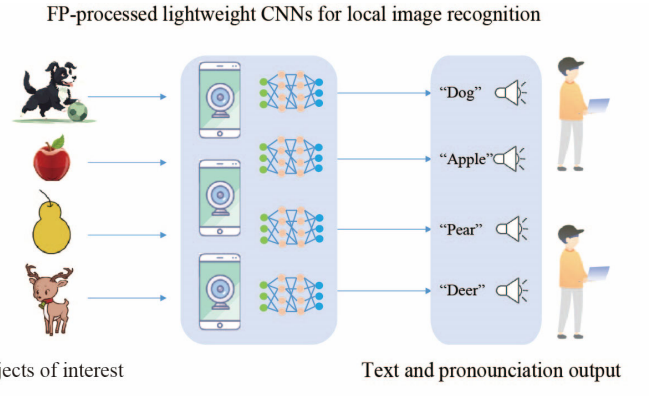

Fig. 7. Android application framework based on fusion pruning method.

As shown in Fig. 7, users learning English can utilize our smart education app on their smart phone or tablet. In particular, they can capture an image of interest using the device's built-in camera. The app performs local image recognition and offline image retrieval directly on the device, eliminating the need to send images back to servers via the mobile network for classification. The results are subsequently converted into text and corresponding audio. The algorithm implements the following strategy to enable offline recognition of local objects. First, a high-performance original model is pretrained on a public dataset. The parameters, models, and labels are saved on smartphones to enable offline recognition. Then, the pretrained models are pruned and fused to generate a lightweight CNN that is deployed on the smartphones or tablets. During offline recognition, the retrieved results are obtained by comparing the inference outputs to the saved labels.

B. Analysis of computational overhead and APP performance

This study utilized a computer equipped with an i5 12400F CPU, 12GB of graphics memory on an RTX3060 GPU, and 32GB of RAM to conduct experiments, aiming to establish a fair hardware environment for analyzing computational overhead.

Parameters and FLOPs are important metrics for measuring model complexity. The larger the number of parameters and FLOPs, the greater the computational overhead, leading to increased memory consumption by the corresponding APP and longer recognition task processing time. In convolutional layers, each convolutional kernel performs operations on input feature maps, involving numerous parameter updates and floating-point calculations. Therefore, the parameters and floating-point operations of the baseline model primarily

concentrate in the convolutional layers. The proposed method also achieves its effects by processing the convolutional layers. TABLE IV also demonstrates similar findings for the proposed model on CIFAR100.

The baseline model generated a final model with 20.04M parameters and 7.98×10^8 FLOPs after completing learning on CIFAR-10, as shown in TABLE III. On the other hand, the FP method had 1.63 parameters and 3.02×10^8 FLOPs, which is only 8.1% and 37.8% of the original model respectively. The accuracy only decreased by 0.5%. This indicates that the proposed method significantly reduces the complexity of the baseline model. In other words, FP achieves comparable recognition accuracy using only 8.1% of the original model's parameters and 37.8% of its FLOPs.

TABLE IX shows the performance of the apps running on three brands of smartphones and tablets. For comparison, we selected the original model to represent a CNN without FP processing. The model size of the original model is 76.5 MB, with inference latencies of 214 ms, 614 ms, and 419 ms on the three devices for the CIFAR10 dataset. In contrast, our FP method reduced the inference latency by 84%, 92%, and 81%, respectively, and the model size by 90%. Additionally, as shown in TABLE X, the FP method improved the size and latency on the CIFAR100 dataset.

Subsequently, we integrated the final models generated by the baseline method and the proposed method into separate mobile APPs to simulate the algorithm in real-world scenarios. The sizes of the two APPs were 76.5MB and 7.51MB respectively (as shown in TABLE IX). This indicates that the models processed through FP have smaller software sizes, which is advantageous for deploying the APP on resource-constrained devices.

Finally, we also evaluated the inference speed of two different apps in recognizing objects. Latency was used to represent the inference speed in the experiment, where a higher delay indicates a slower inference speed, and vice versa. TABLE IX demonstrates that the latency of the proposed method on three hardware platforms is 35ms, 47ms, and 80ms, respectively, which is significantly lower than the baseline model's 214ms, 614ms, and 419ms. This indicates that the proposed method's app is capable of achieving faster image recognition, which is crucial for resource-constrained devices without high-performance CPUs and GPUs. As shown in TABLE X, in CIFAR100, the complexity and computational overhead of the algorithm also exhibit similar comparative results.

C. Fusion pruning-processed model for home robots

The model processed through fusion pruning is also applicable to consumer electronics domains such as household robots. Household robots capture images through onboard cameras and are analyzed by controllers to autonomously display anthropomorphic behavior. The FP model can be easily deployed into the hardware systems of household robots without the need for large-scale CPUs, GPUs, or storage systems. We simulated the performance of the proposed method using a publicly available dataset named RoboCup@Home-Objects [44], sourced from the world robotics competitions. The dataset comprises a large-scale collection of 196 k images, encompassing 180 classes of everyday household items encountered in routine domestic chores, including *Cleaning stuff, Containers, Cutlery, Drinks, Food, Fruits, Snacks, and Tableware*. Fig. 8 shows the structure of the RoboCup@Home-Objects dataset.

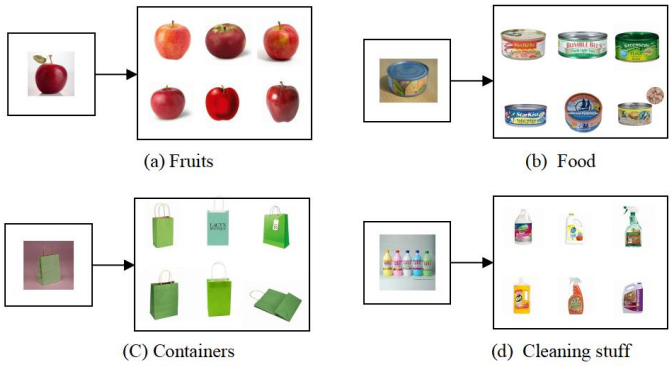

Fig. 8. Examples of different categories from the RoboCup@Home-Objects dataset.

We conducted experiments using NVidia A100 servers, which consist of 8 GPUs with 80 GB of memory each. Among these GPUs, we utilized 7 for training, compressing, and accelerating the FP method on the RoboCup@Home-Objects dataset. Initially, we employed the FP method to prune the CNN on the ImageNet dataset. Subsequently, we froze the convolutional layer parameters of the model and modified the classifier to accommodate 8 major categories and 180 minor categories for mixed training. The batch size was set to 384, and the learning rate decay strategy was consistent with the experimental settings on the CIFAR datasets. The results, as shown in TABLE VII and TABLE VIII, demonstrated the satisfactory performance achieved by our method.

TABLE VII
COMPARISON OF THE PROPOSED METHOD WITH THE BENCHMARK ON THE ROBOCUP@HOME-OBJECTS

Method	Parameters (M)	FLOPs ($\times 10^8$)	Accuracy (%)	Precision (%)	Recall (%)	F measure (%)
Alexnet@home8	-	-	67.3	-	-	-
Alexnet@home180	-	-	65.9	-	-	-
Googlenet@home8	-	-	73.0	-	-	-
Googlenet@home180	-	-	67.3	-	-	-
Our FP@home8	**133.0**	**116.5**	**81.1**	**80.8**	**80.1**	**80.4**
Our FP@home180			**60.2**	**59.8**	**60.1**	**59.7**

TABLE VIII
METRICS REPORTING FOR SUBCLASS CLASSIFICATION

Method	Category	Accuracy (%)	Precision (%)	Recall (%)	F measure (%)
Our FP@home180	Cleaning stuff	51.3	50.2	51.0	50.3
	Tableware	57.0	56.8	57.3	56.8
	Food	65.3	65.6	65.6	65.4
	Drinks	61.5	60.4	61.0	60.5
	Snacks	61.7	61.2	61.6	61.2
	Fruits	62.5	62.7	62.5	62.4
	Cutlery	62.0	62.1	61.8	61.7
	Containers	60.2	59.5	59.6	59.4

Our approach achieved an accuracy of 81.1% on the Robot World Cup Professional dataset, representing an improvement of approximately 11% over the baseline model. Moreover, the lower number of parameters and FLOPs indicate that the application developed by the proposed method can be effortlessly deployed to household robots without the need for additional hardware platforms.

TABLE IX
COMPARISON OF APP PERFORMANCE FOR LOCAL OBJECT RECOGNITION BETWEEN ORIGINAL UNPRUNED AND FP-PROCESSED CNNS ON MOBILE DEVICES USING THE CIFAR10 DATASET

Device	Original model		Our FP model	
	Size (MB)	Latency (ms)	Size (MB)	Latency (ms)
OPPO Realme	76.5	214	7.51	35
VIVO T1	76.5	614	7.51	47
HONOR PAD 6	76.5	419	7.51	80

TABLE X
COMPARISON OF APP PERFORMANCE FOR LOCAL OBJECT RECOGNITION BETWEEN ORIGINAL UNPRUNED AND FP-PROCESSED CNNS ON MOBILE DEVICES USING THE CIFAR100 DATASET

Device	Original model		Our FP model	
	Size (MB)	Latency (ms)	Size (MB)	Latency (ms)
OPPO Realme	76.7	234	13.7	38
VIVO T1	76.7	196	13.7	49
HONOR PAD 6	76.7	357	13.7	75

VI. Conclusion and Discussion

In this study, we present a novel fusion pruning-processed method for deploying large-scale convolutional neural networks on smartphones. By leveraging the L2 norm, we enhance the precision of parameter selection and improve the model's nonlinear capability rather than blindly eliminating certain parameters. The transformation we applied to the receptive field made it possible to fuse convolutional layers while maintaining the same output features as those of the baseline model. Through empirical experiments, we strongly demonstrate the effectiveness of our method.

Subsequently, we developed a mobile APP for on-device object recognition to validate the actual software capacity and inference time of the proposed model on resource-constrained devices such as smartphones. This application facilitates the conversion of obtained images into text and audio formats, thereby meeting the specific needs of language education. The results have demonstrated that our approach outperforms current CNN compression algorithms in terms of both software size and inference time.

Resource-constrained devices lack powerful processing units such as CPUs, GPUs, and high-capacity storage systems. The proposed method addresses the issues by compressing and accelerating the model in terms of parameters and FLOPs. After deployment on smartphones, the software size is reduced, and the inference time is accelerated, eliminating the need for powerful hardware platform support. This model can be extended to wearable devices, drones, and autonomous driving systems to achieve image classification tasks. For instance, the proposed methodology can be readily deployed onto household robot platforms [44] to achieve autonomous execution of routine domestic tasks. It can also be applied in the field of power-line inspection, where previous research [45] utilized drones to capture images, which were then transmitted to ground servers for training and subsequently deployed on other hardware platforms such as Jetson Xavier NX for autonomous recognition. However, this approach increased the flight payload of the drone and significantly raised the cost of the inspection system. In contrast, the proposed FP method can be directly deployed into the application program of drones, enabling automatic recognition of targets of interest in the power system without the need for developing additional hardware platforms, thus greatly reducing deployment costs. Furthermore, recognition tasks are performed at the frontend by our method, resulting in a reduced frequency of transmitting large-scale images to the cloud server, thereby minimizing data transmission errors and network congestion.

This study may have some limitations, as the receptive fields of convolutional kernels with different sizes may be inconsistent. This inconsistency could potentially limit the generalization and application of FP methods in networks such as ResNets

In future research, we will explore some interesting topics, including the implementation of the transformation of receptive fields for convolutional kernels of different sizes, as well as the development of improved algorithms for identifying the contributions of convolutional kernels, providing new perspectives and insights for the study of model lightweighting.

References

[1] X. Zhang, C. Bai, and K. Kpalma, "OMCBIR: Offline mob ile content-based image retrieval with lightweight CNN opt imization," *Displays,* vol. 76, p. 102355, 2023/01/01/ 2023, doi: https://doi.org/10.1016/j.displa.2022.102355.

[2] S. Mehraj *et al.*, "RBWCI: Robust and Blind Watermarkin g Framework for Cultural Images," *IEEE Transactions on Consumer Electronics,* vol. 69, no. 2, pp. 128-139, 2023, d oi: 10.1109/TCE.2022.3217974.

[3] H. Wang, Z. Li, Y. Li, B. B. Gupta, and C. Choi, "Visual s aliency guided complex image retrieval," *Pattern Recogniti on Letters,* vol. 130, pp. 64-72, 2020/02/01/ 2020, doi: http s://doi.org/10.1016/j.patrec.2018.08.010.

[4] P. Liu, L. Jiang, H. Lin, J. Hu, S. Garg, and M. Alrashoud, "Federated Multimodal Learning for Privacy-Preserving Dr iver Break Recommendations in Consumer Electronics," *IE EE Transactions on Consumer Electronics,* vol. 70, no. 1, p p. 4564-4573, 2024, doi: 10.1109/TCE.2023.3339630.

[5] D. Li, L. Deng, B. Bhooshan Gupta, H. Wang, and C. Choi , "A novel CNN based security guaranteed image watermar king generation scenario for smart city applications," *Infor*

mation Sciences, vol. 479, pp. 432-447, 2019/04/01/ 2019, doi: https://doi.org/10.1016/j.ins.2018.02.060.

[6] S. Rahman, S. Pal, J. Yearwood, and C. Karmakar, "Analysing Performances of DL-Based ECG Noise Classification Models Deployed in Memory-Constraint IoT-Enabled Devices," *IEEE Transactions on Consumer Electronics,* vol. 70, no. 1, pp. 704-714, 2024, doi: 10.1109/TCE.2024.3370709.

[7] J. Wu, J. Zhang, M. Bilal, F. Han, N. Victor, and X. Xu, "A Federated Deep Learning Framework for Privacy-Preserving Consumer Electronics Recommendations," *IEEE Transactions on Consumer Electronics,* vol. 70, no. 1, pp. 2628-2638, 2024, doi: 10.1109/TCE.2023.3325138.

[8] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Smartphone-based real-time object recognition architecture for portable and constrained systems," *Journal of Real-Time Image Processing,* vol. 19, no. 1, pp. 103-115, 2022/02/01 2022, doi: 10.1007/s11554-021-01164-1.

[9] R. W. Liu, Y. Guo, Y. Lu, K. T. Chui, and B. B. Gupta, "Deep Network-Enabled Haze Visibility Enhancement for Visual IoT-Driven Intelligent Transportation Systems," *IEEE Transactions on Industrial Informatics,* vol. 19, no. 2, pp. 1581-1591, 2023, doi: 10.1109/TII.2022.3170594.

[10] A. M. Srivastava, P. A. Rotte, A. Jain, and S. Prakash, "Handling Data Scarcity Through Data Augmentation in Training of Deep Neural Networks for 3D Data Processing," *International Journal on Semantic Web and Information Systems (IJSWIS),* vol. 18, no. 1, pp. 1-16, 2022, doi: 10.4018/IJSWIS.297038.

[11] S. Din, A. Paul, A. Ahmad, B. B. Gupta, and S. Rho, "Service Orchestration of Optimizing Continuous Features in Industrial Surveillance Using Big Data Based Fog-Enabled Internet of Things," *IEEE Access,* vol. 6, pp. 21582-21591, 2018, doi: 10.1109/ACCESS.2018.2800758.

[12] Y. Wu and J. Chen, "A Lightweight Real-Time System for Object Detection in Enterprise Information Systems for Frequency-Based Feature Separation," *International Journal on Semantic Web and Information Systems (IJSWIS),* vol. 19, no. 1, pp. 1-18, 2023, doi: 10.4018/IJSWIS.330015.

[13] C. Liqun and H. Lei, "Clipping-based Neural Network Post Training Quantization for Object Detection," in *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, 28-30 April 2023 2023, pp. 1192-1196, doi: 10.1109/ICCECT57938.2023.10141287.

[14] N. Singh, J. Rupchandani, and M. Adhikari, "Personalized Federated Learning for Heterogeneous Edge Device: Self-Knowledge Distillation Approach," *IEEE Transactions on Consumer Electronics,* vol. 70, no. 1, pp. 4625-4632, 2024, doi: 10.1109/TCE.2023.3327757.

[15] Y. Yu, Z. Zhao, S. J. Lin, and W. Li, "Accelerating Huffman Encoding Using 512-Bit SIMD Instructions," *IEEE Transactions on Consumer Electronics,* vol. 70, no. 1, pp. 554-563, 2024, doi: 10.1109/TCE.2023.3347229.

[16] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, "Sparse low rank factorization for deep neural network compression," *Neurocomputing,* vol. 398, pp. 185-196, 2020/07/20/ 2020, doi: https://doi.org/10.1016/j.neucom.2020.02.035.

[17] Y. Li *et al.*, "Exploiting Kernel Sparsity and Entropy for Interpretable CNN Compression," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15-20 June 2019 2019, pp. 2795-2804, doi: 10.1109/CVPR.2019.00291.

[18] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing,* vol. 461, pp. 370-403, 2021/10/21/ 2021, doi: https://doi.org/10.1016/j.neucom.2021.07.045.

[19] H. Song, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *ICLR*, 2016.

[20] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-z. Xu, "Dynamic channel pruning: Feature boosting and suppression," *arXiv preprint arXiv:1810.05331,* 2018.

[21] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," *Advances in Neural Information Processing Systems (NeurIPS) 30,* vol. 30, 2178-2188, 2017.

[22] C. Zhang, T. Hu, Y. Guan, and Z. Ye, "Accelerating Convolutional Neural Networks with Dynamic Channel Pruning," in *DCC*, 2019, p. 563.

[23] M. K. Yi, W. K. Lee, and S. O. Hwang, "A Human Activity Recognition Method Based on Lightweight Feature Extraction Combined With Pruned and Quantized CNN for Wearable Device," *IEEE Transactions on Consumer Electronics,* vol. 69, no. 3, pp. 657-670, 2023, doi: 10.1109/TCE.2023.3266506.

[24] U. Kulkarni, A. S. Hosamani, A. S. Masur, S. Hegde, G. R. Vernekar, and K. S. Chandana, "A Survey on Quantization Methods for Optimization of Deep Neural Networks," in *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*, 13-15 Dec. 2022 2022, pp. 827-834, doi: 10.1109/ICACRS55517.2022.10028742.

[25] Q. Sun *et al.*, "One model for all quantization: A quantized network supporting hot-swap bit-width adjustment," *arXiv preprint arXiv:2105.01353,* 2021.

[26] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295,* 2021.

[27] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing,* vol. 520, pp. 152-170, 2023/02/01/ 2023, doi: https://doi.org/10.1016/j.neucom.2022.11.072.

[28] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in Neural Information Processing Systems,* vol. 29, 2016.

[29] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704-2713.

[30] A. Alzahrani and M. Z. Asghar, "Maintaining User Security in Consumer Electronics-Based Online Recommender Systems Using Federated Learning," *IEEE Transactions on Consumer Electronics,* vol. 70, no. 1, pp. 2657-2665, 2024, doi: 10.1109/TCE.2023.3325224.

[31] K.-L. Du, M. N. S. Swamy, Z.-Q. Wang, and W. H. Mow, "Matrix Factorization Techniques in Machine Learning, Signal Processing, and Statistics," *Mathematics*, vol. 11, no. 12, doi: 10.3390/math11122674.

[32] J.-H. Luo and J. Wu, "AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference," *Pattern Recognition,* vol. 107, p. 107461, 2020/11/01/ 2020, doi: https://doi.org/10.1016/j.patcog.2020.107461.

[33] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance Estimation for Neural Network Pruning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15-20 June 2019 2019, pp. 11256-11264, doi: 10.1109/CVPR.2019.01152.

[34] Z. Zhuang *et al.*, "Discrimination-aware channel pruning for deep neural networks," *Advances in Neural Information Processing Systems,* vol. 31, 2018.

[35] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *International Journal of Computer Vision,* vol. 129, no. 6, pp. 1789-1819, 2021/06/01 2021, doi: 10.1007/s11263-021-01453-z.

[36] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[37] M. Poyser and T. P. Breckon, "Neural architecture search: A contemporary literature review for computer vision applications," *Pattern Recognition,* vol. 147, p. 110052, 2024/03/01/ 2024, doi: https://doi.org/10.1016/j.patcog.2023.110052.

[38] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning Efficient Convolutional Networks through Network Slimming," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 22-29 Oct. 2017 2017, pp. 2755-2763, doi: 10.1109/ICCV.2017.298.

[39] Z. Lyu *et al.*, "A survey of model compression strategies for object detection," *Multimedia Tools and Applications,* vol. 83, no. 16, pp. 48165-48236, 2024/05/01 2024, doi: 10.1007/s11042-023-17192-x.

[40] I. Sergey and S. Christian, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015/06/01. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html.

[41] T. Huang *et al.*, "DyRep: Bootstrapping Training with Dynamic Re-parameterization," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 18-24 June 2022 2022, pp. 578-587, doi: 10.1109/CVPR52688.2022.00067.

[42] F. Wang et al., "Residual Attention Network for Image Classification," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 6450-6458, doi: 10.1109/CVPR.2017.683.

[43] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681,* 2019.

[44] N. Massouh, L. Brigato, and L. Iocchi, "RoboCup@Home-Objects: Benchmarking Object Recognition for Home Robots," in *RoboCup 2019: Robot World Cup XXIII*, Cham, S. Chalup, T. Niemueller, J. Suthakorn, and M.-A. Williams, Eds., 2019// 2019: Springer International Publishing, pp. 397-407.

[45] Z. Li, Q. Wang, T. Zhang, C. Ju, S. Suzuki, and A. Namiki, "UAV High-Voltage Power Transmission Line Autonomous Correction Inspection System Based on Object Detection," *IEEE Sensors Journal,* vol. 23, no. 9, pp. 10215-10230, 2023, doi: 10.1109/JSEN.2023.3260360.