Journal of Information Systems Engineering and Management
2025, 10(10s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Hybrid Model Compression Pipeline for Efficient Object Detection using Parallel Recurrent Convolutional Neural Networks

Shraddha S. More[1*] and Dr. Rajesh Bansode[2]
[1]Research Scholar, Department of Information Technology, Thakur College of Engineering and Technology, Mumbai, India
[2]Professor, Department of Information Technology, Thakur College of Engineering and Technology, Mumbai, India
[*]Corresponding author : moreshraddha30@gmail.com

ARTICLE INFO

Received: 04 Nov 2024

Revised: 22 Dec 2024

Accepted: 05 Jan 2025

ABSTRACT

Several Models of Deep Learning (DL) have demonstrated impressive performance across multiple object detection problems. Large object detection methods based on DL are typically computationally and memory-intensive. Hence, this paper presents model compression strategies for object identification with Parallel Recurrent Convolutional Neural Networks (MCS-OD-PRCNN). Initially, the input photos come from the Common Objects in Context (COCO) 2017 dataset. Next, using Improved Bilateral Texture Filtering (IBTF), the input images are pre-processed. The pre-processed images are then given to the suggested deep-learning model, Parallel Recurrent Convolutional Neural Networks (PRCNNs), which identifies and localizes the objects in the image. After training and validating the PRCNN model on the pre-processed dataset, compression model precision is decreased with the application of strategies like quantization and pruning, eliminating redundant weights and connections, and training a smaller, more efficient student model based on the larger PRCNN model. To ensure optimal performance, hybrid fox and chimp optimization algorithms (Hyb-FCOA) are employed for the compression model's parameter tuning. The suggested methodology is carried out in Python environment, and fundamental evaluation metrics such as Accuracy, Precision, Recall, F-Measure, mean Average Precision (mAP), Matthew's Correlation Coefficient (MCC), Intersection over Union (IoU), and Positive Predictive Value (PPV) are employed to evaluate the strategy's performance. The proposed method attains 20.08%, 23.35%, and 27.79% higher accuracy compared to existing techniques such as using one-to-one instruction and guided hybrid quantization for remote sensing object detection (GHOST-GQSD), Fast Region-Based Convolutional Neural Network (Fast-RCNN), and You Only Look Once version 4 (YOLOv4), respectively.

Keywords: Deep Learning, Hybrid Fox and Chimp optimization algorithm, Improved Bilateral Texture Filtering, Model Compression Strategies, Object Detection, Parallel Recurrent Convolutional Neural Network

1. Introduction

Convolutional neural networks (CNNs) have become more and more popular in recent years, especially for segmentation, object recognition, and picture classification [1]. Object detection networks include YOLOv3\v4, FasterRCNN, and SSD, whereas classification uses AlexNet, ResNet, and MobileNets [2]. Some jobs require neural network models with over 100 layers, up from eight earlier. Large networks, despite the additional resources needed, can better represent features. The YOLOv4 network model, for example, is 256 MB in size, with 64 million parameters and 162 layers of depth [3]. 29G FLOPs (Floating Point Operations Per Second) are needed to process a 416 by 416 image, and additional RAM is required for the intermediate variables. The scale of the model means that embedded systems with low resources cannot fulfill the memory requirements for inference and computational load [4]. Model compression approaches aimed at neural network deployment on embedded or mobile devices rely on methods including lightweight network design, quantification, pruning, and knowledge distillation [5]. Weight-level pruning strategies were proposed as a means to reduce the model's parameter count without sacrificing network accuracy. However, weight-level pruning models demand the use of specialized hardware accelerators [6]. Filter-level pruning techniques have been proposed as a way to reduce deployment costs without requiring extra hardware support. The binary and ternary networks which integrate quantization and pruning were initially introduced in the quantification work [7] and are employed in the classification network. Each layer's compression ratio is determined by the pre-trained parameter information, and the effective compression impact is measured using the shared codebook technique. Low-bit quantification networks have the potential to reduce model sizes, but accuracy may deteriorate significantly in the process. Furthermore, its implementation usually necessitates the use of a specific software acceleration library [8]. Few studies have looked at employing the aforementioned pruning and quantization approaches for detection networks; they are more commonly used in classification networks. Through quantization and pruning, the present network parameters and structure will be compressed [9]. To prevent accuracy loss due to quantization or pruning, optimization or direct design are utilized in knowledge distillation and lightweight network architecture. One tactic for leveraging the increased student network performance through teacher networks is knowledge distillation. The distillation principle was initially applied in the classification network by Hinton [10].

Natural language processing, computer vision, and speech recognition all make extensive use of knowledge distillation nowadays [11]. Knowledge distillation has little to no impact on lowering the model's size and parameters, but it can enhance the student network's performance. Additionally, there is a barrier that prevents the instructor and student networks from conveying the knowledge that has to be condensed [12]. The structural differences between student and instructor networks have a substantial impact on the distillation effect. To mitigate the effects of structural differences on distillation, representation learning is combined with knowledge distillation [13]. For each class in the image, item counts and picture labels are all that are needed for this approach. Regression activation maps and class activation maps are masked to achieve this. When combined, they provide an accurate localization of items in the scene [14]. These efforts might partially resolve the distillation's knowledge representation issue. It offers a poorly supervised method for object detection and explains how to address the partial view alignment problem using representation learning [15]. Lightweight network design, or the direct building of small networks or modules, offers a simple network application for tasks involving object identification. Constant channel modules reduce memory access costs, whereas attention feature modules boost network accuracy [16]. For semantic segmentation tasks, two separate encoder-decoder lightweight designs were proposed. The ensuing work minimizes the encoder's parameter count by using channel split and shuffle modules and also improves accuracy in the decoder by adding an attention module [17]. The former work enhances contextual clues by up-sampling the deep layer's convolution features to the shallow deconvolution layers. Directly building lightweight modules allows for a fair balance between model size and precision. Activities involving autonomous driving can be easily included with these thin networks [18]. These well-planned networks or modules can usually be run on host computers in laboratories. To properly integrate these networks on edge devices, however, much testing and tuning will be required to ensure their efficacy [19]. Before deploying the parameters and network to the edge device, they must be compiled and quantified. Certain creative modules or network layers are usually impossible to develop and pass due to the hardware device's limited instruction set and fundamental operators, which reduces these networks' adaptability and makes it more difficult to deploy them to edge devices [20].

CNNs are one of the most popular deep learning models and have recently shown exceptional performance in a variety of computer vision uses, such as object identification. However, these models present substantial hurdles in real-world applications due to their high computational and memory requirements, particularly on devices with constrained resources like mobile phones and embedded systems. Traditional CNNs, with their complicated structures and a large number of parameters, are frequently overly resource-intensive, leading to long inference times and excessive energy usage. This constraint reduces their practical utility in applications needing real-time processing and low-latency answers. Furthermore, current object detection techniques demand significant storage and processing resources, limiting their scalability and accessibility. The COCO 2017 dataset, which is widely used to train object detection models, presents a considerable computing challenge due to its size and complexity. Thus, there is an urgent need for effective model compression algorithms that can minimize these models' computing weight while retaining their performance and correctness. The goal of this manuscript is to overcome the aforementioned issues by presenting a compression approach designed exclusively for object detection tasks. The suggested strategy employs several novel techniques to improve model efficiency while maintaining accuracy.

The primary contributions of the manuscript are as follows,

- By combining PRCNN with IBTF, the model becomes more adept at locating and identifying items in intricate images.
- Model efficiency can be raised by using model compression techniques such as quantization, pruning, and knowledge distillation, which can significantly lower computing and memory needs.
- Compressed models can be implemented on resource-constrained devices, like mobile phones and embedded systems, increasing the practical utility of high-performance object identification.
- The suggested approach is adaptable to other deep learning models and datasets, providing a versatile solution for computer vision challenges.
- Hyb-FCOA algorithms optimize compressed models, maintaining high accuracy despite reduced size and precision.

The manuscript's remaining sections are arranged as follows: Important works are displayed in Section 2, the proposed part is covered in Section 3, the results and discussion are presented in Section 4, and the manuscript is concluded in Section 5.

2. Related Work: A Brief Review

This section contains recent attempts among several studies on object detection utilizing model compression and deep learning approaches.

In 2023, Zhang, J., *et al.*, [21] have shown an one-to-one self-teaching in conjunction with guided supervised hybrid quantization for remote sensing image object detection. To accomplish lightweight, first design a framework called GQSD, a novel idea that blends quantization and distillation. The method of training was driven by the full-precision model of the quantization model, which saved time and money by removing the requirement to build a previously trained model in advance. Second, include an HQ module that uses a distribution distance threshold between the center and samples in the weight value search space to automatically choose the appropriate bit-width within a constrained constraint. Third, offer an OST module so that the student network can assess itself and improve knowledge transformation.

In 2023, Zhang, L. and Ma, K., [22] have contributed to the organized knowledge extraction process for precise and effective object detection. First, suggest the major reasons about knowledge distillation fails in object detection were disparities in pixel counts between foreground and background, as well as a lack of information distillation concerning the relationships between distinct pixel counts. Next, offer a structured method for knowledge distillation that addresses the two issues by fusing non-local distillation with attention-guided distillation. The idea behind attention-guided distillation is to have students work harder to understand the characteristics of foreground things by identifying critical pixels that have attention mechanisms. In addition to teaching students about the attributes of a single pixel, it was suggested that non-local distillation be used to teach students about the relationships between pixels collected by non-local modules.

In 2022, Chu, Y., *et al.*, [23] have demonstrated spatial attention distilling and group channel pruning for object identification. First, provide a dynamic sparse training technique with changeable sparse rate. The network was trained to get rid of unnecessary channels and strike a good compromise between precision and sparsity. Secondly, offer a unique pruning technique called group channel pruning to mitigate the impact of pruning on network accuracy. Specifically, categorizes the network into multiple categories based on the closeness of its module structure and feature layer sizes. The channels within each category were then trimmed using various criteria. Utilize an enhanced knowledge distillation method to regain the trimmed network's accuracy.

In 2022, Junos, M.H., *et al.*, [24] have shown how to employ a condensed CNN model for automated object detection on aerial photographs for embedded devices with constrained storage. A movable inverted bottleneck module serves as the cornerstone for a viable, lightweight deep CNN-based object identification model. Concatenating the multi-scale local region features with an upgraded spatial pyramid pooling approach further broadened the network's receptive field. The experimental results demonstrated that, in terms of average precision and memory storage requirements, the proposed model performed better than previous studies. Furthermore, the suggested model provides optimal trade-offs between detection time, model size, and accuracy, making it a top choice for deployment on capacity-constrained embedded devices.

In 2022, Chen, C., *et al.*, [25] have shown how to improve object detection's resilience using 6G vehicle edge computing. A recommended technique for improving the resilience of AI model deployment using 6G-VEC was demonstrated using object recognition as an example. Two phases comprised this technique: model adaption and model stability. In the former, incorporating cutting-edge methodologies improves the model's robustness. In the latter, trade-offs between runtime resources and model performance are required due to the application of two targeted compression strategies: knowledge distillation and model parameter trimming. The suggested method performed better than the other possibilities in onboard edge terminals, according to numerical results, and it was simple to implement there.

In 2022, Zheng, Y.J., *et al.*, [26] have offered model compression that utilizes pruned differentiable network channels. Provide a model compression method based on DNCP. The suggested method uses gradient descent to effectively discover the ideal substructure that satisfies resource restrictions, unlike existing approaches that need sampling and evaluating multiple substructures. More precisely, assign a learnable probability to every potential number of channels in each network layer, relax the selection of a specific number of channels to a softmax over all possible channels, and apply gradient descent to maximize the learnable probability in an end-to-end manner. Once the network's parameters have been adjusted, use the learnable probability to trim the network until you reach the optimal substructure.

In 2022, Pikoulis, E.V., *et al.*, [27] have shown how to accelerate deep convolutional networks using a new clustering-based method. Compared to traditional k-means-based approaches, offer a clustering-based method that can speed up the procedure and use more centroids/representatives. To that purpose, the specifics of the current issue allow for the imposition of a unique framework on the employed representatives. Furthermore, the system's major hyperparameters that influence theoretical acceleration increases were identified and supplied. Comprehensive evaluation trials were conducted using different cutting-edge DNN models trained in image classification are used to show how effective the suggested methodology over existing approaches for MCA tasks.

3. Proposed Methodology

In this section, model compression strategies for object detection using Parallel Recurrent Convolutional Neural Networks (MCS-OD-PRCNN) is discussed. The proposed methodology involves initially preprocessing input images from the COCO 2017 dataset using Improved Bilateral Texture Filtering (IBTF), followed by training and validating the parallel Recurrent Convolution Neural Networks (PRCNN) on these preprocessed images to accurately identify and localize objects, and subsequently applying model compression techniques such as quantization, pruning, and knowledge distillation, complemented by hybrid optimization strategies for parameter tuning, to achieve an efficient and high-performance object detection model suitable for deployment on resource-constrained devices. The suggested methodology's block diagram is depicted in Figure 1, and the proposed framework's thorough description is provided below.

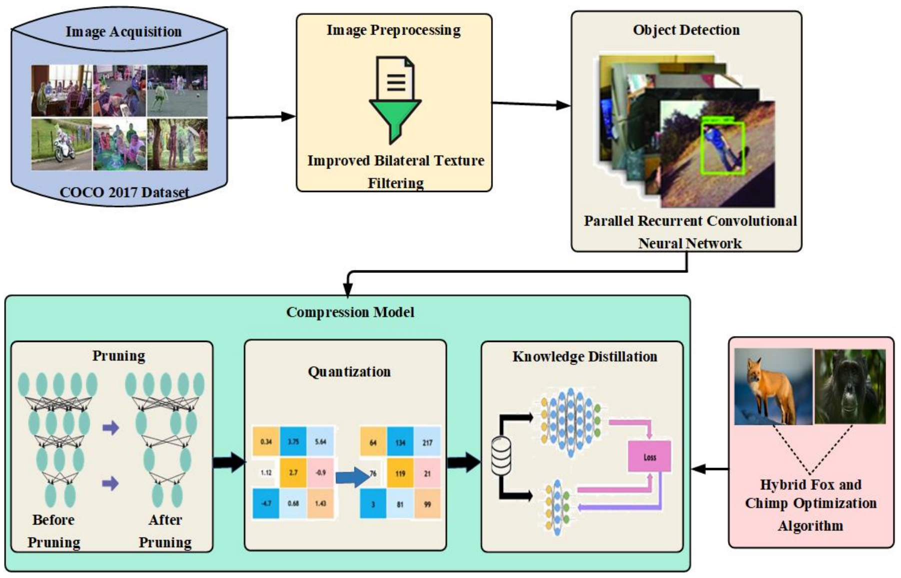


Figure 1: Block Diagram for the Proposed MCS-OD-PRCNN Methodology

3.1. Image Acquisition

Initially, the training procedure uses a publicly accessible COCO-2017 database [28]. This enormous collection contains 328,000 images, totaling 2.5 million categorized instances. The dataset addresses three major research challenges in scene recognition: exact two-dimensional object localization, contextual inference between objects, and recognizing non-famous or insignificant angles of view. Choosing object categories is a difficult task. All classes must be covered in the categories, which should also be relevant to real-world applications and frequently large enough to allow for the collection of significant datasets.

3.2. Preprocessing Using IBTF

The IBTF [29] is used to the input images as a pre-processing step to normalize and improve image quality. With the IBTF adjustment, the bilateral filter becomes a non-linear filter that decreases noise while preserving edges. The bilateral filter considers the degree of similarity between the intensities and locations of pixels during the smoothing process. Taking these methods reduces the noise in the supplied images. The spatial weights are first generated using a Gaussian kernel, and the image resolution affects the standard deviation, as shown in equation (1).

$$l[a,b] = \begin{pmatrix} Y_{a,a} & Y_{a,b} \\ Y_{b,a} & Y_{b,b} \end{pmatrix} \tag{1}$$

The adjacent pixel of l is represented as Y and $[a,b]$ when the pixel for spatial weight in the equation above is given as $l[a,b]$. This yields the spatial weights of the Gaussian kernel's standard deviation. The intensity weights are then constructed using a Gaussian kernel, and equation (2) expresses the standard deviation in terms of the amount of noise contained in the image.

$$I_a = \alpha I_a + (1-\alpha_a) l_a \tag{2}$$

In the computation above l, the pixel's intensity weight is given as I_a, and its intensity is shown as α . l, the pixel next to it, l, is represented as αI_a, and the Gaussian kernel's standard deviation, $1-\alpha_a$, is displayed when calculating the intensity weight. Lastly, determine the weighted average of the pixels that surround each pixel in the image. Multiply the intensity weights by the product of the spatial weights to determine the weights. The formula (3) is used to calculate it.

$$l(\vartheta_a) = \Delta(\vartheta_a) \frac{\sum (\partial l)_a}{m * m} \tag{3}$$

where pixel l's filtered intensity is represented by the equation above as ϑ_a, the pixels surrounding pixel l are defined as $\Delta(\vartheta_a)$, the pixel's spatial weight is indicated by ∂ as l, the pixel's intensity weight l is represented as $(\partial l)_a$, and the pixel's next neighbor's intensity is indicated by m as l.

3.3. Parallel Recurrent Convolutional Neural Network for Object Detection

The recommended Parallel Recurrent Convolutional Neural Network (PRCNN) [30] is a crucial and often-used instrument for object detection in the field of classification and prediction techniques. The two deep learning structures that make up the model successfully include the best features of both Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) for temporal and spatial feature extraction, respectively. The enhanced RNN structure, or "Long Short-Term Memory (LSTM)" model, gathers contextual data for streaming 1D input vectors, while the CNN unit is devoted to mining cross-channel correlations and extracting features from 2D frames. The object identification procedure is finished by combining the recovered attributes using a feature fusion technique after these two units have finished operating. To extract spatial data, the CNN segment consists of three consecutive 2D convolutional layers sharing a 4×4 kernel size. Despite 3×3 kernels being frequently utilized in computer vision, a 4×4 filter is required because object classification frames are sparse. The 4x4 filter may look for correlations between extra channels when this option is used. Zero padding is used in every convolutional layer to avoid information loss at the frame edges. From the original 32 feature maps, the number of feature maps in each successive layer of the convolutional layer is doubled.

Consequently, the second and third convolutional layers have feature maps of 64 and 128 respectively. This model does not have a pooling layer after the convolutional layer, in contrast to traditional CNN designs that usually have. Although the pooling layer is occasionally implemented for data dimension reduction at the risk of some information loss, it is not necessary for this assault recognition because the data frame is significantly smaller than in computer vision applications. In order to preserve all the data, the model doesn't use pooling techniques. To speed up the model's training process, a batch normalization (BN) step is applied following each convolution process. Following the three convolutional layers, a depth concatenate operation is used to integrate the spatial feature vectors with the temporal feature vectors that the RNN produced to create a cohesive cube from the spatial feature maps. When processing incoming data frames, the CNN module prioritizes spatial information. On the other hand, the RNN module, which is responsible for extracting temporal features, works with 1D data vectors without translating them into 2D frame sequences. This design decision highlights a purposeful approach to preserve the data's sequential structure, enabling the RNN to recognize temporal relationships essential for successful object detection. Equation (4) is used to create a Spatial Feature Vector following the 2D-CNN processing of each segment.

$$UHX_i = Conv2FD(U_i), UHX_i \quad (4)$$

There is no translation of the 1D object recognition data vectors to 2D frames because the RNN component extracts the temporal features. The j^{th} input windowed segment is processed in its original 1D format by the RNN.

$$T_i = \left[w_{u,} w_{u+1,\ldots,w_{u+T-1}} \right] \quad (5)$$

The hidden state is represented in equation (6), at the final time step of a given section.

$$i_{u+U-1} = LSTM(T_i), i_{u+U-1} \quad (6)$$

In order to enhance the ability to communicate temporal information, a completely linked layer where d represents the hidden state size of the LSTM unit is positioned both before and after the LSTM layers. Therefore, using equation (7) for segment T_i, the Temporal Feature Vector can be expressed as follows.,

$$VHX_i = GD\left(i_{u+U-1}\right), VHX_i \quad (7)$$

Before and after the LSTM layers, a fully linked layer is employed to enhance the attack categorization data representation. The suggested method successfully identifies the object as a result.

3.4. Compression Pipeline

This section further details the compression algorithm in this part, covering quantization, knowledge distillation, and pruning to compress the proposed PRCNN deep learning model.

3.4.1. Pruning

The absolute value of the scale factor γ indicates each channel's significance, which is a function of the Batch Normalization layers in neural networks. Here, employ some real statistics to represent the effect of γ on feature map channels intuitively. For comparison and statistics, choose the PRCNN network's Batch Normalization layers. To quantify and examine every channel in the feature map, employ the $L1$ norm. It is evident that the feature map's $L1$ norm, It is occasionally dispersed close to 0 and was acquired after the convolution layer. Nevertheless, the following Batch Normalization layer, which comprises many scale factors distributed about 0, results in a significant number of channels with minimal $L1$ norm in the feature map. The network's scale factors are all expressed as H in absolute terms:

$$H = \{|\gamma_1|, |\gamma_2|, \ldots, |\gamma_n|\} \quad (8)$$

where n indicates how many channels there are in the network overall. By defining a pruning rate in H , can choose a relatively modest threshold, which is specified as θ. For a scale factor with an absolute value less than θ, pruning is feasible. This framework employs an iterative pruning technique to get a better fitting ability. Following several rounds of pruning and fine-tuning can obtain a slim network that not only fulfills the requirements but also results in a comparatively reduced accuracy loss. The trial has demonstrated the effectiveness of this strategy.

As the pruning process proceeds, certain convolutional layers might be removed due to the irregular distribution of scale factors. Accuracy would suffer greatly as a result. The initial trimming would have significantly altered the network's structure. The goal is to merely eliminate a portion of each layer's filters, not to remove an entire convolutional layer. The chain derivative principles dictate that a substantial change in the structure would cause a considerable movement in the gradient. Eventually, the model would become too inaccurate, and fine-tuning would take longer. It also needs to consider the scale factors in the current layer and their overall significance for the Batch Normalization layers. To preserve the network structure's integrity, it has placed another scale factor threshold list ϑ here:

$$\vartheta = \{\theta_1, \theta_2, ..., \theta_N\} \tag{9}$$

Were,

$$\theta_i = \alpha \times \max\left(\{\gamma_{i1}, \gamma_{i2}, ..., \gamma_{i3}\}\right) \tag{10}$$

The number α is both more than 0 and less than 1, and there are N convolutional layers. The i^{th} Batch Normalization layer is set to have an extra θ_i point cutoff. For instance, the channel may be rejected if the associated scale factor absolute value is below both the θ_i and θ limits. This accomplishes two goals. One benefit is that it can prevent the removal of some convolutional layers by retaining at least a small portion of the channels during pruning.

3.4.2. Knowledge Distillation

Following pruning, the thin model's accuracy is somewhat less than the original network's. Reducing knowledge is a good strategy to compensate for reduced accuracy. The first step in knowledge distillation involves moving knowledge from a large, pre-trained network to a smaller, smaller network. The following equation can be used to represent the small model's ultimate loss function during the backpropagation phase of the small network.

$$L_t = L_o + \lambda * L_{kd} \tag{11}$$

The numbers L_t , L_o , Display the first loss function, the final loss function, and the output difference between small and big network and L_{kd} in the equation, respectively. It must change the weight coefficient of λ . The cooperative training of two networks is called mutual learning. In other words, teacher network training can be guided by student networks as well. The large and tiny networks can be represented by $Net-1$and $Net-2$, respectively, and their overall mutual learning loss may be calculated using the formula below:

$$L_{t1} = L_{o1} + L_{kd1} \tag{12}$$

$$L_{t2} = L_{o2} + L_{kd2} \tag{13}$$

Through mutual learning, $Net-1$and $Net-2$take part in the process of backpropagation. In mutual learning, λ remains constant as opposed to the initial one-way knowledge distillation, which can save a significant amount of tuning time. Figure 2 shows the compression model for object detection.

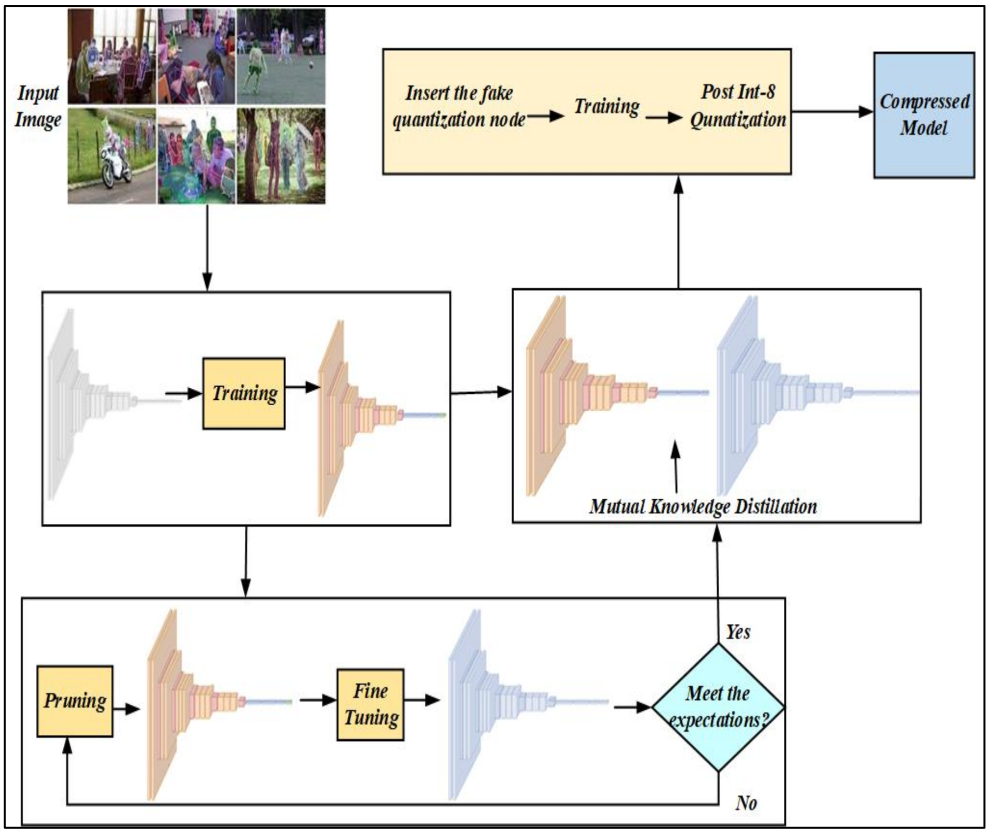


Figure 2: Compression Model for Object Detection

In the task of classifying images, feed the network $Batchsize$ training data and receive an output in the shape of $Batchsize \times Classes$. Use cross-entropy as the loss function in optimization. The $Kullback - Leibler\,(KL)$ divergence [14] is a popular tool for describing the difference between two probability distributions. You can calculate the output's KL divergence by feeding the same input into both networks. The final step in the knowledge distillation process is to make the output of two networks tend to be consistent by incorporating KL divergence into the loss function and backpropagation. The following formula is used to calculate KL divergence,

$$KL(Q\|R) = \sum Q(x)\log\frac{Q(x)}{R(x)} \tag{14}$$

where Q and R stand for two distinct probability distributions with $Batchsize \times Classes$ as their shape. To obtain total loss, we add the original cross-entropy and the KL divergence of the two distributions.

Batch samples are also used as the input for PRCNN, from which both classification and regression results. PRCNN object detection is predicated on previous boxes. First place a sequence of previous boxes on every pixel of feature maps with varying scales. Next, allow the regression and classification processes to be coordinated by each prior box. Each pixel in the feature map corresponds to four separate preceding boxes, such as if the size of the j feature map is $i_j \times x_j$ and there are J Feature maps for regression and classification results. Next, each preceding box on the feature map is matched using convolution to o_j coordinate values and the classification outcome. As a result, $\sum_j^J i_j \times x_j \times o_j$ can be used to symbolize the total of previous boxes. In contrast to the image classification task, there are $NumPriorBoxes$ outputs for every image data in the PRCNN. The output has two different classification shapes: $NumPriorBoxes \times Classes$ and $NumPriorBoxes \times 4$. Only positive and negative samples are used to calculate total loss, even when one image data set includes $NumPriorBoxes$ prior boxes. The early boxes with a

Ground Truth IOU of more than 0.5 are regarded as the positive sample; the earlier boxes with a larger classification loss and a lower IOU are regarded as the negative sample. Because the coordinates of prior boxes are fixed, the selection of positive samples for a given piece of picture data is always the same, regardless of the training state. Negative sample selection, on the other hand, depends on network components and is dynamic.

The investigation has shown that the best results can only be obtained by condensing knowledge on the categorization branch. Given that the regression branch's output is non-discrete, it is not possible to describe the difference in their distribution using just KL divergence. It was found that the method's performance on PRCNN was inferior to that of knowledge distillation alone when it came to the classification branch. In PRCNN, the term "entire loss" is defined as follows:

$$L = \frac{1}{O_{pos}} \left(L_{conf}\left(p, plabel\right) + \alpha L_{loc}\left(l, l_{label}\right) \right) \tag{15}$$

Where,

$$L_{conf}\left(p, plabel\right) = \sum_{j=1}^{O_{pos}+O_{neg}} CE\left(p_j, plabel\right) \tag{16}$$

Positive and negative sample numbers are indicated by O_{pos} and O_{neg}. The results of the classification and label of each previous item are shown in boxes p and $plabel$. The regression label and output for each previous box are shown in boxes l and l_{label}. L_{loc} is the regression part's smooth $L1$ loss function. The $Cross-Entropy$ loss function is represented by CE. The PRCNN uses the boxes that came before it to define its loss function rather than identifying each image as an item. Here select positive samples for knowledge distillation since they are unrelated to the model. This is unable to guarantee the constancy of the negative samples because they vary depending on the network. The object detection network's knowledge distillation technique revolves around this.

$$\begin{aligned} L_{conf(p1, plabel)} &= \sum_{j=1}^{O_{pos}+O_{eg}} CE\left(p1, plabel\right) + KL(p_2 \| p_1) \\ &= \sum_{j=1}^{O_{pos}+O_{eg}} CE\left(p1, plabel\right) + \sum_{i=1}^{O_{pos}} p_{2_i} \log\left(\frac{p_{2_i}}{p_{1_i}} \right) \end{aligned} \tag{17}$$

$$\begin{aligned} L_{conf}\left(p_2, plabel\right) &= \sum_{j=1}^{O_{pos}+O_{neg}} CE\left(p_{2_j}, plabel\right) + KL\left(p_1 \| p_2\right) \\ &= \sum_{j=1}^{O_{pos}+O_{neg}} CE\left(p_{2_j}, plabel\right) + \sum_{i=1}^{O_{pos}} p_{1_i} \log\left(\frac{p_{1_i}}{p_{2_i}} \right) \end{aligned} \tag{18}$$

There is an additional branch of object confidence in the distillation's loss function for PRCNN. Therefore, this branch needs to be included while creating the knowledge distillation loss function.

3.4.3. Quantization

After knowledge distillation is complete, quantize the over-parameterized neural network model using FP-32 to Int-8 with the least amount of accuracy drop. This is unable to simply convert the thin model into the Int-8 integer model due to its very low resilience. In lightweight models, quantization-aware training could yield good results. Hence incorporate fictitious quantization nodes into recognizable procedures. These nodes are used to count the highest and lowest values of data that pass through them as they are being trained. This does not do quantization; instead, this training procedure only models the quantization procedure. FP-32 is still used in both the forward computation and backpropagation processes. To improve performance after training, transform it to an Int-8 model. The quantization formula from the floating point to the fixed point is as follows:

$$R = round\left(\frac{S}{T}\right) + U \tag{19}$$

The inverse quantization formula from fixed point to floating point is as follows:

$$S = \left(R - U\right) \times T \tag{20}$$

A floating-point value is represented by S ; a quantized fixed-point value is represented by U ; a floating-point value is represented by T ; and the lowest scale that may be conveyed following fixed-point quantization is represented by R . A floating-point type is converted to an integer type by the process of rounding. The following formula can be used to determine numbers T and U .

$$T = \frac{S_{\max} - S_{\min}}{R_{\max} - R_{\min}} \tag{21}$$

And

$$U = R_{\max} - round\left(\frac{S_{\max}}{T}\right) \tag{22}$$

The maximum floating-point number in the suggested approach is $S_{\max}$, the smallest is $S_{\min}$, the highest fixed-point value is $R_{\max}$, and the lowest fixed-point value is $R_{\min}$, or $\left(0\right)$. Insert the quantization node and use the optimizer to lower the quantization loss in order to incorporate the previously mentioned method into the training process. In addition, each Batch Normalization layer and the convolution layer need to be fused. This is the exact formula for the algorithm. The new convolution layer finishes the work left by batch normalization and the first convolution by adjusting the weight and offset value. The convolution layer can be computed using the formula below:

$$z_1 = x * y + c \tag{23}$$

where x , y , and c stand for the convolutional layer's weight, input, and bias, respectively. The convolutional layer's output is represented by z_1 . After merging, may obtain the following output directly, taking into account the function of the Batch Normalization layer:

$$z = \gamma \times \frac{z_1 - \mu}{\sqrt{\delta^2 + \in}} + \beta \tag{24}$$

$$z = \gamma \times \left(\frac{\left(x * y + c\right) - \mu}{\sqrt{\delta^2 + \in}}\right) + \beta \tag{25}$$

$$z = \frac{\gamma \times x}{\sqrt{\delta^2 + \in}} * y + \frac{\gamma \times \left(c - \mu\right)}{\sqrt{\delta^2 + \in}} + \beta \tag{26}$$

The numbers γ and β correspond to the BN layer's scale factor and bias factor, respectively. The z_1 mean and variance are represented by μ and δ . The output of the BN layer is represented by Z . The denominator's value is fixed at $1e-5$ since $\in$ prevents it from reaching 0. The final combined convolutional layer's weight and bias can then be obtained.

$$x_{merged} = x \times \frac{\gamma}{\sqrt{\delta^2 + \in}} \tag{27}$$

$$c_{merged} = (c - \mu) \times \frac{\gamma}{\sqrt{\delta^2 + \in}} + \beta \tag{28}$$

To improve the performance of the compression model, weight parameters N, L, R from pruning, knowledge distillation, and quantization are optimized more efficiently using the hybrid Fox and Chimp optimization algorithm (Hyb-FCOA). The stepwise procedure of the Hybrid optimization algorithm is as follows,

3.5. Stepwise Procedure of the Hybrid Fox and Chimp Optimization Algorithm

In this section, the weight parameters N, L, R from pruning, knowledge distillation, and quantization are provided to the hybrid Fox and Chimp optimization algorithm (Hyb-FCOA) for the process of optimization. The proposed Fox Optimization Algorithm (FOA) [31] is a metaheuristic optimization algorithm. Fox distinguishes itself by incorporating the hunting and searching behaviors of red foxes in snowy environments. It mimics a red fox's elegant dive and spring into the snow to get its meal. In order to determine the best jumping method for a successful hunt, Fox measures the distance from the prey throughout the exploitation phase. Additionally, an artificial fox is employed to execute jumps in both northeast and opposite directions, determining the new position by considering factors such as prey distance, jump value, and direction range. On the other hand, the Fox Optimization Algorithm is improved by utilizing the Chimp Optimization Algorithm (COA) [32]. The chimpanzee hunting process is categorized into two primary phases: the "Exploration" phase involves activities such as driving, blocking, and chasing the prey, while the "Exploitation" phase encompasses the actual attack on the prey. The suggested Hyb-FCOA algorithm's step-by-step process is given below:

Step 1: Initialization

In this step, initialize the weight parameters N, L, R from pruning, knowledge distillation, and quantization for the process of optimization.

Step 2: Random Generation

Chooses at random the most appropriate answer from the initialized input parameter upon initialization.

Step 3: Fitness Function

Here, the objective function for the weight parameter optimization using Hyb-FCOA optimization is represented in equation (29),

$$Fitness\,Function = Optimization\left[N, L, R\right] \tag{29}$$

Step 4: Exploitation Phase in Fox

The variable has a probability distribution in the interval [0, 1] during the exploitation phase. Should the randomly generated number q exceeds 0.18, It implies that the red fox needs to relocate. This new position is determined by a number of factors, encompassing the distance that sound travels $Dist\_S\_T_{it}$, the separation between the prey and the red fox $Dist\_Fox\_\Pr ey_{it}$ and the jumping value $Jump_{it}$ must be taken into account. Subsequently, the sound travel duration is represented by a randomly generated number, ranging from 0 to 1. By increasing its speed through the atmosphere, $Time\_S\_T_{it}$ can determine the red fox's sound travel distance. Sp_S in conjunction with the sound travel time $Time\_S\_T_{it}$. Using equation (30), the following computation is made:

$$Dist\_S\_T_{it} = Sp\_S * Time\_S\_T_{it} \tag{30}$$

The medium's sound speed, represented as Sp_S, stays steady at 343 above the ground. Additionally, $Time\_S\_T_{it}$ is a number that is created at random from [0, 1]. Iterations range from one to five hundred.

Step 5: Updating COA for FOA

At this point, equation (31) is used to calculate the update of the Chimp optimization method for the Fox optimization algorithm.

$$Dist\_S\_T_{it} = \left[Sp\_S * Time\_S\_T_{it}\right] \times d \tag{31}$$

Where, d is considered as the updation function of the Chimp optimization algorithm and is assessed using equations (32) and (33) as follows,

$$d = \left|k.X_P(t) - aX_{chimp}(t)\right| \tag{32}$$

$$X_{chimp}(t+1) = X_P(t) - r.d \tag{33}$$

Where, the total number of iterations is denoted as (t), $k, a,$ and r are represent coefficient vectors, The prey's location is shown as, $X_P(t)$ and the chimp's position is indicated as $X_{chimp}(t)$.

Step 6: Exploration Phase in Fox

At this point, the red fox controls the haphazard stroll by deliberately searching for the nicest site it has located so far. At this point, instead of employing a jumping strategy, the fox chooses to randomly explore the search area in order to carefully examine possible prey. The shortest duration variable $MinT$ and variable a are two crucial elements that work together to ensure that the fox meanders aimlessly in the direction of the best location. Equations (34) and (35) show the results of the computations for variables $MinT$ and a. Finding variable tt minimal value is necessary to determine variable $MinT$.

$$tt = \frac{sum\left(Time_{S_{T_{it}}}(i,:)\right)}{Dimesnion}, M\text{in}T = Min\left(tt\right) \tag{34}$$

The average time tt is obtained by dividing the $sum\left(Time_{S_{T_{it}}}(i,:)\right)$ by the dimension of the problem.

$$a = 2 * \left(it - \left(\frac{1}{Max_{it}}\right)\right) \tag{35}$$

Where, Max_{it} indicates the maximum iterations.

Step 7: Returning the Solution's Optimal Place

Step 8: Termination

Check the requirements for stopping. Stop the process if, after the allotted number of iterations, the halting criteria are satisfied; otherwise, go to step 3. At last, the proposed Hyb-FCOA algorithm tunes the weight parameters more efficiently.

4. Result and Discussion

This section presents the experimentation done on model compression strategies for object detection using Parallel Recurrent Convolutional Neural Networks (MCS-OD-PRCNN). The following performance metrics are measured while implementing the suggested method on a python environment: Accuracy, Recall, F-Measure, Precision, mean Average Precision (mAP), Matthew's correlation coefficient, Intersection over Union (IoU), Positive Predictive Value (PPV). Here, the recommended MCS-OD-PRCNN methodology is evaluated against established techniques like

Using one-to-one instruction and guided hybrid quantization to recognize objects in remote sensing pictures (GHOST-GQSD) [21], Organized knowledge extraction for precise and effective item identification (Fast-RCNN) [22] and Pruning group channels and condensing spatial attention to discover objects (YOLOv4) [23] respectively.

4.1. Performance Measures

In this situation, the recommended technique's usefulness is assessed using performance metrics such as Accuracy, Recall, F-Measure, Precision, mean Average Precision (mAP), Matthew's correlation coefficient, Intersection over Union (IoU), and Positive Predictive Value (PPV). Use the metrics of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to better comprehend the proposed approach.

4.1.1. Accuracy

Accuracy is defined as the ratio of accurately anticipated instances to total instances. It is calculated as using equation (36),

$$Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN} \right) \tag{36}$$

4.1.2. Precision

Precision is defined as the ratio of accurately predicted positive observations to expected positives. It is computed using equation (37),

$$\Pr ecision = \left(\frac{TP}{TP + FP} \right) \tag{37}$$

4.1.3. Recall

Recall is the ratio of correctly anticipated positive observations to all observations made in class. The calculation is performed using equation (38),

$$\operatorname{Re} call = \left(\frac{TP}{TP + FN} \right) \tag{38}$$

4.1.4. F1-Score

The F-Measure balances precision and recall by taking the harmonic mean of both. The calculation is done with equation (39).

$$F - Measure = 2 \times \left(\frac{\Pr ecision \times \operatorname{Re} call}{\Pr ecision + \operatorname{Re} call} \right) \tag{39}$$

4.1.5. Mean Average Precision (mAP)

The mean average precision (mAP) is calculated by averaging the precision (AP) for all classes. The area under the Precision-Recall curve (AP) is calculated. It is computed using equation (40),

$$mAP = \frac{1}{n} \sum_{i=1}^{n} AP_i \tag{40}$$

4.1.6. Matthews Correlation Coefficient (MCC)

Even when the classes have drastically different sizes, the MCC metric which accounts for true and false positives as well as negatives is widely recognized as a balanced measurement. It is computed using equation (41),

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{(TP + FP)(TP + FN)(TN + FP)(TN + FN)} \tag{41}$$

4.1.7. Intersection over Union (IoU)

IoU calculates the overlap between the ground truth and predicted bounding boxes. It is computed using equation (42),

$$IoU = \frac{Area\,of\,Overlap}{Area\,of\,Union} \quad (42)$$

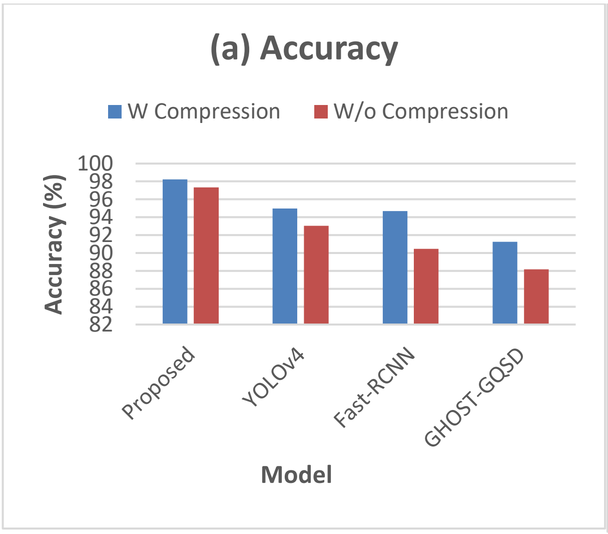
4.1.8. Positive Predictive Value (PPV)

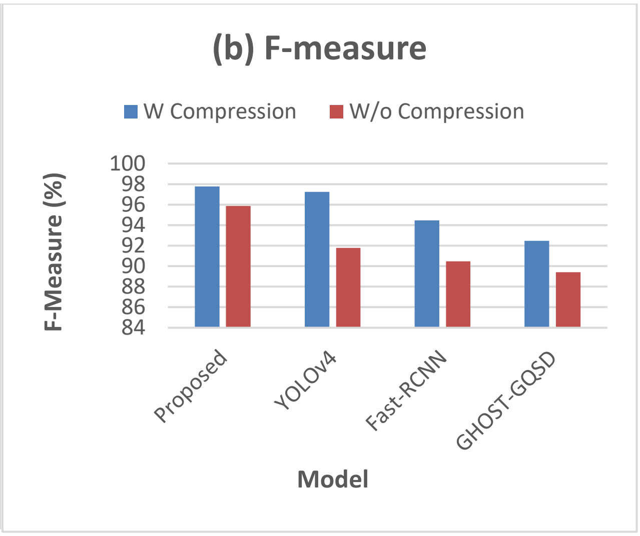
The Positive Predictive Value (PPV), also known as precision, calculates the proportion of true positive detections among all positive detections made by the model. It is computed using equation (43),

$$PPV = \frac{TP}{TP + FP} \quad (43)$$

4.2. Simulation Results Comparing the Suggested Method with Current Methods

Figure 3(a)-(h) displays the outcomes of the recommended approach's simulation. Here, the effectiveness of the proposed strategy is contrasted with that of alternative techniques like GHOST-GQSD, Fast-RCNN, and YOLOv4 respectively with compression (W Compression) and without compression model (W/O Compression).





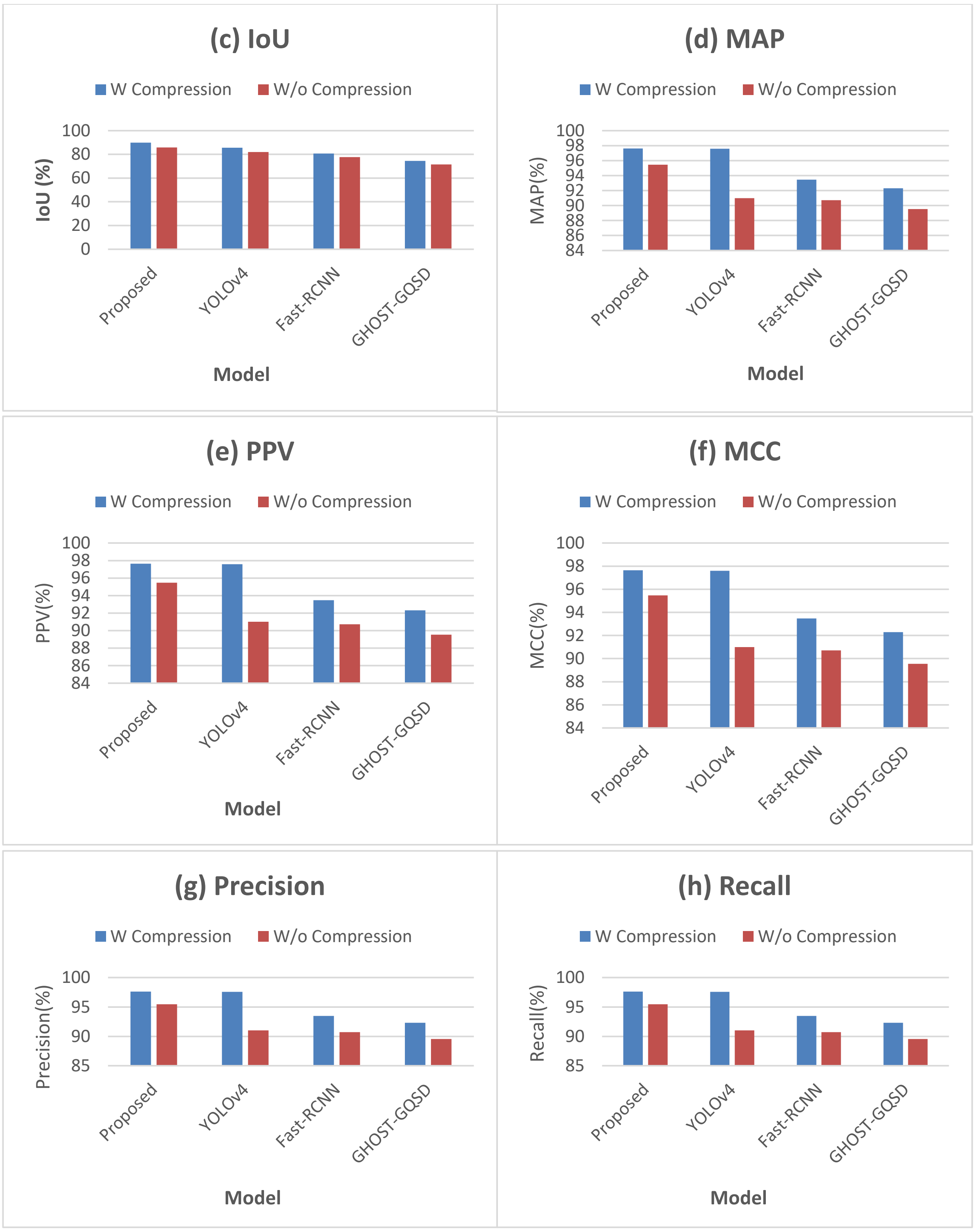


Figure 3: Performance Analysis of (a)Accuracy (b)F-Measure (c)MCC (d)Recall (e)Precision (f)IoU (g)MAP (h)PPV

Figure 3a) displays the Accuracy performance assessment. This assessment of the suggested approach offers 20.65%, 22.45%, and 27.09% higher Accuracy with compression; 23.14%, 24.16%, and 28.15% higher Accuracy without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN and YOLOv4 respectively. Figure 3b) displays the F-Measure performance evaluation. This assessment of the suggested approach offers 20.14%, 22.14%, and 31.61% higher F-Measure with compression; 21.77%, 24.71%, and 35.63% higher F-Measure without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN and YOLOv4 respectively. Figure 3c) displays the MCC performance evaluation. This assessment of the suggested approach offers 19.04%, 20.94%, and 21.24% higher MCC with compression; 24.07%, 27.34%, and 30.71% higher MCC without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN and YOLOv4 respectively. Figure 3d) displays the Recall performance evaluation. This assessment of the suggested approach offers 23.64%, 24.29%, and 29.34% higher Recall with compression; 24.12%, 26.78%, and 31.34% higher Recall without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN and YOLOv4 respectively. Figure 3e) displays the Precision performance assessment. This assessment of the suggested approach offers 25.00%, 27.09%, and 31.74% higher Precision with compression; 20.12%, 22.08%, and 26.47% higher Precision without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN, and YOLOv4 respectively. Figure 3f) displays the evaluation of IoU performance. This assessment of the suggested approach offers 20.64%, 22.45%, and 28.64% higher IoU with compression; 22.05%, 26.25%, and 30.54% higher IoU without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN and YOLOv4 respectively. Figure 3g) displays the evaluation of MAP performance. This assessment of the suggested approach offers 26.04%, 28.65%, and 32.74% higher MAP with compression; 20.54%, 21.31%, and 24.04% higher MAP without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN, and YOLOv4 respectively. Figure 3h) displays the evaluation of PPV performance. This assessment of the suggested approach offers 22.75%, 25.05%, and 30.35% higher PPV with compression; 24.14%, 27.95%, and 29.65% higher PPV without compression compared to existing techniques like GHOST-GQSD, Fast-RCNN and YOLOv4 respectively.

Following figure 4 provide a comparison of the different compression rates for various compression techniques applied to the MCS-OD-PRCNN model. They include pruning, knowledge distillation, quantization, and a proposed compression approach known as the hybrid approach. Results show that the proposed hybrid approach provided the highest compression rate at 5.9. This is followed by Quantization with a rate of 5.3, then 4.8 for Knowledge Distillation, and lastly, 4.3 for Pruning. That is to say, the hybrid approach is most effective in reducing model size with minimal degradation of model performance, hence establishing its superiority over techniques of individual compression.

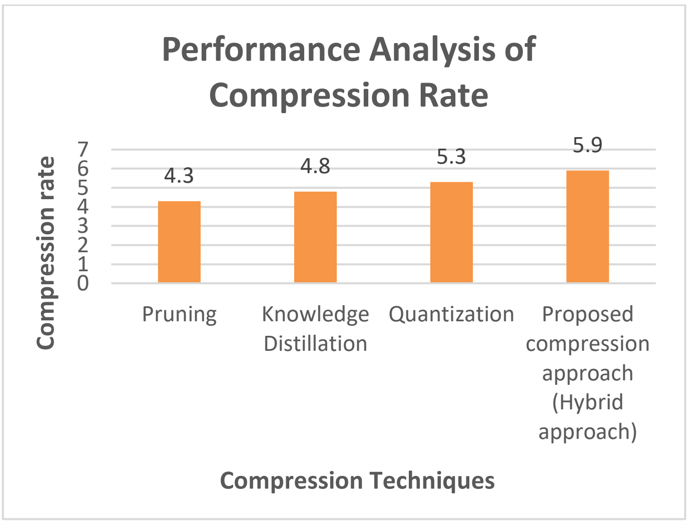


Figure 4: Performance Analysis of Compression Rate

Figure 5 shows a comparison of metrics parameters, FLOPS, and model size for the existing GHOST-GQSD, Fast-RCNN, YOLOv4, and the proposed MCS-OD-PRCNN model. In all respects, the values are maximum for the GHOST-GQSD model: about 1.8 x 10^5 parameters, 7.5 x 10^5 FLOPS, with a model size of 2000 kB. The Fast-RCNN & YOLOv4 model has moderate values, with about 1.0 x 10^5, 0.9 x 10^5 parameters, 6 x 10^5, 4.5 x 10^5 FLOPS, and a size of 1500, 1300 kB. The lowest values for the proposed MCS-OD-PRCNN model with ~0.6 x 10^5 parameters, ~2.5 x 10^5 FLOPS, and a size of 800 kB. These numbers thus reflect that concerning existing models like GHOST-GQSD, Fast-RCNN, and YOLOv4, the proposed model MCS-OD-PRCNN boasts enormous reductions in parameters, computational complexity, and model size using state-of-the-art compression techniques like Pruning, Knowledge Distillation, Quantization, and a Hybrid approach, making it highly efficient and very deployable in resource-constrained environments.

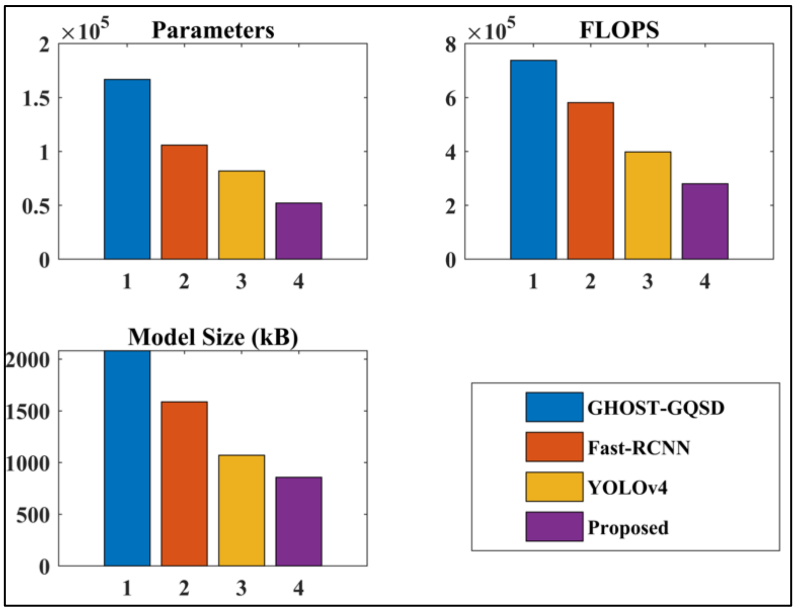


Figure 5: Performance Analysis of Computational Cost and Model Size

5. Conclusion

In this paper, model compression strategies for object detection using Parallel Recurrent Convolutional Neural Networks (MCS-OD-PRCNN) demonstrate significant potential in enhancing the efficiency and practicality of deep learning models. By incorporating Improved Bilateral Texture Filtering (IBTF) for preprocessing and employing sophisticated techniques like as quantization, pruning, and knowledge distillation, the methodology reduces model size and compute needs significantly while maintaining excellent accuracy. Hybrid optimization strategies further ensure optimal performance of the compressed model. This complete method not only addresses the essential issues of deploying object detection models on resource-constrained devices, but it also lays the groundwork for scalable and efficient solutions in a variety of real-world applications. Python environment is used to implement the recommended strategy. Performance metrics like Accuracy, Recall, F-measure, Precision, mean Average Precision (mAP), Matthew's Correlation Coefficient (MCC), Intersection over Union (IoU), and Positive Predictive Value (PPV) are examined here. The proposed method achieves 19.01%, 21.48%, and 22.36% higher precision compared with existing techniques like GHOST-GQSD, Fast-RCNN, and YOLOv4 respectively. To enhance the deployment of object detection models on edge devices like smartphones, drones, and Internet of Things devices, future research will concentrate on broadening the suggested model compression approaches.

Reference

[1] Diwan, T., Anirudh, G. and Tembhurne, J.V., 2023. Object detection using YOLO: Challenges, architectural successors, datasets and applications. *multimedia Tools and Applications*, *82*(6), pp.9243-9275.

[2] Bonnaerens, M., Freiberger, M. and Dambre, J., 2022. Anchor pruning for object detection. *Computer Vision and Image Understanding*, *221*, p.103445.

[3] Ni, J., Shen, K., Chen, Y. and Yang, S.X., 2023. An improved ssd-like deep network-based object detection method for indoor scenes. *IEEE Transactions on Instrumentation and Measurement*, *72*, pp.1-15.

[4] Zhao, H., Sun, X., Dong, J., Manic, M., Zhou, H. and Yu, H., 2022. Dual discriminator adversarial distillation for data-free model compression. *International Journal of Machine Learning and Cybernetics*, pp.1-18.

[5] Liu, H., Li, D., Jiang, B., Zhou, J., Wei, T. and Yao, X., 2022. MGBM-YOLO: a faster light-weight object detection model for robotic grasping of bolster spring based on image-based visual servoing. *Journal of Intelligent & Robotic Systems*, *104*(4), p.77.

[6] Balamurugan, D., Aravinth, S.S., Reddy, P.C.S., Rupani, A. and Manikandan, A., 2022. Multiview objects recognition using deep learning-based wrap-CNN with voting scheme. *Neural Processing Letters*, *54*(3), pp.1495-1521.

[7] Kwon, M.J., Nam, S.H., Yu, I.J., Lee, H.K. and Kim, C., 2022. Learning jpeg compression artifacts for image manipulation detection and localization. *International Journal of Computer Vision*, *130*(8), pp.1875-1895.

[8] Yang, Z., Wang, X., Wu, J., Zhao, Y., Ma, Q., Miao, X., Zhang, L. and Zhou, Z., 2022. Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision. *IEEE/ACM Transactions on Networking*.

[9] Liu, K., 2022. Stbi-yolo: A real-time object detection method for lung nodule recognition. *IEEE Access*, *10*, pp.75385-75394.

[10] Huang, J.S., Chen, B.Q., Zeng, N.Y., Cao, X.C. and Li, Y., 2023. Accurate classification of ECG arrhythmia using MOWPT enhanced fast compression deep learning networks. *Journal of Ambient Intelligence and Humanized Computing*, pp.1-18.

[11] Liu, D., Kong, H., Luo, X., Liu, W. and Subramaniam, R., 2022. Bringing AI to edge: From deep learning's perspective. *Neurocomputing*, *485*, pp.297-320.

[12] Rodriguez-Conde, I., Campos, C. and Fdez-Riverola, F., 2022. Optimized convolutional neural network architectures for efficient on-device vision-based object detection. *Neural Computing and Applications*, *34*(13), pp.10469-10501.

[13] Li, B., Ye, L., Liang, J., Wang, Y. and Han, J., 2022. Region-of-interest and channel attention-based joint optimization of image compression and computer vision. *Neurocomputing*, *500*, pp.13-25.

[14] Mani, V.R.S., Saravanaselvan, A. and Arumugam, N.J.M.J., 2022. Performance comparison of CNN, QNN and BNN deep neural networks for real-time object detection using ZYNQ FPGA node. *Microelectronics Journal*, *119*, p.105319.

[15] Zhou, S., Deng, X., Li, C., Liu, Y. and Jiang, H., 2022. Recognition-oriented image compressive sensing with deep learning. *IEEE Transactions on Multimedia*.

[16] Nagarajan, A. and Gopinath, M.P., 2023. Hybrid optimization-enabled deep learning for indoor object detection and distance estimation to assist visually impaired persons. *Advances in Engineering Software*, *176*, p.103362.

[17] Wu, Z., Li, S., Chen, C., Hao, A. and Qin, H., 2022. Recursive multi-model complementary deep fusion for robust salient object detection via parallel sub-networks. *Pattern Recognition*, *121*, p.108212.

[18] Chen, Y., Xu, H., Zhang, X., Gao, P., Xu, Z. and Huang, X., 2023. An object detection method for bayberry trees based on an improved YOLO algorithm. *International journal of digital earth*, *16*(1), pp.781-805.

[19] Zhang, Y., Yu, J., Chen, Y., Yang, W., Zhang, W. and He, Y., 2022. Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application. *Computers and Electronics in Agriculture*, *192*, p.106586.

[20] Ding, P., Qian, H. and Chu, S., 2022. Slimyolov4: lightweight object detector based on yolov4. *Journal of Real-Time Image Processing*, *19*(3), pp.487-498.

[21] Zhang, J., Lei, J., Xie, W., Li, Y., Yang, G. and Jia, X., 2023. Guided hybrid quantization for object detection in remote sensing imagery via one-to-one self-teaching. *IEEE Transactions on Geoscience and Remote Sensing*.

[22] Zhang, L. and Ma, K., 2023. Structured knowledge distillation for accurate and efficient object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[23] Chu, Y., Li, P., Bai, Y., Hu, Z., Chen, Y. and Lu, J., 2022. Group channel pruning and spatial attention distilling for object detection. *Applied Intelligence*, *52*(14), pp.16246-16264.

[24] Junos, M.H., Khairuddin, A.S.M. and Dahari, M., 2022. Automated object detection on aerial images for limited capacity embedded device using a lightweight CNN model. *Alexandria Engineering Journal*, *61*(8), pp.6023-6041.

[25] Chen, C., Yao, G., Wang, C., Goudos, S. and Wan, S., 2022. Enhancing the robustness of object detection via 6G vehicular edge computing. *Digital Communications and Networks*, *8*(6), pp.923-931.

[26] Zheng, Y.J., Chen, S.B., Ding, C.H. and Luo, B., 2022. Model compression based on differentiable network channel pruning. *IEEE Transactions on Neural Networks and Learning Systems*.

[27] Pikoulis, E.V., Mavrokefalidis, C., Nousias, S. and Lalos, A.S., 2022. A new clustering-based technique for the acceleration of deep convolutional networks. *Deep Learning Applications, Volume 3*, pp.123-150.

[28] https://www.kaggle.com/datasets/awsaf49/coco-2017-dataset

[29] Xu, P. and Wang, W., 2018. Improved bilateral texture filtering with edge-aware measurement. *IEEE Transactions on Image Processing*, *27*(7), pp.3621-3630.

[30] Yang, R. et al. (2020) 'Parallel recurrent convolutional neural networks-based music genre classification method for mobile devices', IEEE Access, 8, pp. 19629–19637.

[31] Mohammed, H. and Rashid, T., 2023. FOX: a FOX-inspired optimization algorithm. *Applied Intelligence*, *53*(1), pp.1030-1050.

[32] Khishe, M. and Mosavi, M.R., 2020. Chimp optimization algorithm. *Expert systems with applications*, *149*, p.113338.