

Novel Human-Machine Interface via a Robust Hand Gesture Recognition System Using a Channel-Pruned You Only Look Once Version 5 Model

Abir Sen

SRM University-AP

Rabi Shaw

shawr@nit.j.ac.in

Dr. B. R. Ambedkar National Institute of Technology

Tapas Kumar Mishra

National Institute of Technology

Ratnakar Dash

National Institute of Technology

Research Article

Keywords: Deep Learning, Hand Gesture Recognition, Image Processing, Object Detection, You Only Look Once, Human-Computer Interaction

Posted Date: October 7th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10038394/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Novel Human-Machine Interface via a Robust Hand Gesture Recognition System Using a Channel-Pruned You Only Look Once Version 5 Model

Abir Sen¹, Rabi Shaw², Tapas Kumar Mishra³, Ratnakar Dash³

¹Computer Science and Engineering, SRM University-AP, Andhra Pradesh, India.

^{2*}Information Technology, Dr. B. R. Ambedkar National Institute of Technology, Jalandhar, Punjab, India.

³Dept of CSE, National Institute of Technology, Rourkela, Odisha, India.

Contributing authors: abir.s@srmap.edu.in; shawr@nitj.ac.in; tap1cse@gmail.com; ratnakar@nitrkl.ac.in;

Abstract

Hand gesture recognition (HGR) is a vital component in enhancing the human-computer interaction (HCI) experience, particularly in multimedia applications, such as virtual reality, gaming, and smart home automation systems. Users can control and navigate through these applications seamlessly by accurately detecting and recognizing gestures. However, in real-time scenarios, the performance of gesture recognition systems is sometimes affected due to complex backgrounds, low-light illumination, and occlusion problems. Another issue is building a fast and robust gesture-controlled interface in real-time scenarios.

The overall objectives of this paper are: (1) to develop an efficient hand gesture detection and classification model using a channel-pruned You Only Look Once version 5 small model, (2) utilizing the lightweight pruned model to build a gesture-controlled interface with a quick response time (in milliseconds) and higher detection speed (in frames per second). First, the You Only Look Once version 5 small (YOLOv5s) model is selected for the gesture detection task. Next, the model is simplified using a channel-pruning algorithm, and the pruned model is further fine-tuned to ensure detection efficiency. We compare our proposed scheme with other state-of-the-art methods and observe that our model achieves superior results in terms of mean average precision (mAP), precision, recall, rapid inference time, and detection speed.

Our proposed method enables the deployment of a pruned YOLOv5s model for real-time gesture-command-based interfaces to control applications such as the VLC media player and Spotify player using accurately classified gesture commands. The average detection speed of our proposed system exceeds 60 frames per second, meeting the requirements of real-time application control. Furthermore, we illustrate the efficacy of gesture-controlled VLC and Spotify players in real-time scenarios with regard to detection rate and response time, and improved the robustness of gesture-controlled applications.

Keywords: Deep Learning, Hand Gesture Recognition, Image Processing, Object Detection, You Only Look Once, Human-Computer Interaction

1 Introduction

Hand gesture recognition is now essential to human-computer interaction. Hand gesture-controlled HCI enables flexible, user-friendly communication between humans and machines without physical contact. There are many human-machine interface modalities, including GUI, voice-command, hand-gesture, and touch-screen. Hand gesture recognition is a promising modality because of its naturalness and convenience. Robust hand gesture detection systems are essential for enhancing human-machine communication and optimizing user experience. There are two types of hand gesture recognition systems: wearable sensor-based [1, 2] and vision-based [1, 2]. The sensor-based recognition system requires a glove and connected connection to classify gestures. It sometimes produces false gesture classes with complex backgrounds. In other hand vision-based gesture recognition involves collecting images from a camera, segmenting to eliminate background, feature extraction, and gesture image classification. Hand gesture recognition has made human-machine interaction easier by eliminating the need for a keyboard and mouse. However, building a robust hand gesture recognition system in the presence of a cluttered background and light varying conditions is a challenging task. This study presents a lightweight channel-pruned YOLOv5s model-based robust hand gesture identification and gesture-controlled HMI to operate multimedia applications to overcome these challenges. The significant contributions of our work are summarized as follows:

1. Fine-tuning of the YOLOv5s model using hyperparameter settings for hand gesture detection and classification in real-time scenarios.
2. To make the model more compact and accelerate the inference speed in real-time, the YOLOv5s model is further pruned using a channel-pruning algorithm.
3. After obtaining the pruned model, there will be a slight drop in accuracy due to the model size-accuracy trade-off. To enhance the detection accuracy, the model is subsequently retrained with the hyperparameter settings.
4. Development of an HMI to control VLC and Spotify players with quick response time and higher inference speed in terms of frames per second (fps) by utilizing the fine-tuned pruned YOLOv5s model.
5. The robustness of the gesture-controlled HMI is improved in real time.

2 Literature survey

The advancement of computer vision in recent years has greatly facilitated human-machine interaction

through hand gesture detection. However, operating multimedia applications with gesture commands presents obstacles such as gesture segmentation, cluttered backgrounds, and low-light conditions. So, recognizing gestures is a primitive task for developing a gesture-controlled HMI. Second, we require a lightweight deep learning model that can recognize hand gestures, map them, and use them as gesture commands to control multimedia applications with quick response time. Various state-of-the-art works for hand gesture recognition have been reported in the last decades. Some vision-based gesture classification are based on Hidden Markov model (HMM), K-nearest neighbor algorithms, and support vector machine (SVM). For instance, a static hand gesture recognition system was proposed by Huang *et al.* [3]. They have used the Gabor filter for feature extraction tasks and SVM classifier classification. Their proposed approach has achieved a recognition rate of 95.2% and a processing time of 0.2 sec per frame. Another contribution of Rahman and Afrin [4] presents a static hand gesture recognition system using SVM classifier. In this work, the Canny edge detector is used to detect the edges of a gesture portion. Next, the SVM classifier is employed to classify the gestures. Recently, with the advent of deep learning, the CNN algorithm has gained popularity for extracting high-level features and classification tasks. Many state-of-the-art works have been reported regarding hand gesture classification using CNN. Sharma *et al.* [5] have suggested a deep learning approach after modifying VGG11 and VGG16 architecture to classify static hand gestures of sign language. In their experiment, they have considered ASL and ISL datasets to evaluate the model performances. They have achieved the classification accuracy of 94.84% and 99.96%. Sen *et al.* [6] have devised a novel system for recognizing hand gestures. The system consisted of three stages: (1) detecting gestures using binary thresholding, (2) segmenting the gesture portion, and (3) training three custom CNN classifiers simultaneously. The output scores of these models were then calculated to construct an ensemble model for the final prediction.

The difficulty lies in developing a robust framework for gesture recognition that yields precise outcomes. However, it faces obstacles in real-world scenarios, including lighting conditions, complex backgrounds, varying distances, and occlusion issues. The researchers find it difficult to detect and recognize the correct gesture classes under these conditions with quick response time and higher detection speed. Under these scenarios, object detection algorithms such as, YOLO [7], Faster-RCNN (region-based convolutional neural networks) [8] play a vital role in localizing and recognizing the gesture classes in real-time. There are some research articles regarding the robust hand gesture recognition system using object detection algorithms. For instance, Bose *et al.* [9] have Introduced a hand gesture recognition system in the presence of intricate backgrounds. They utilized the Faster-RCNN framework with the Inception-V2 model as feature extractors. Their proposed method has achieved a detection time of 140ms per gesture. In [10], the

authors have proposed a gesture recognition system utilizing the Faster RCNN model. They train the model using the Standard hand gesture database under complicated backgrounds. Their classification accuracy exceeds 90.6%. Sen *et al* [11] has suggested a robust hand gesture recognition system based on frozen YOLOv5s model under complicated backgrounds and low-light conditions. In their work, they have optimized the YOLOv5s architecture by freezing some convolutional layers in the backbone portion, resulting in a reduction in model complexity. Their suggested framework has achieved a mean average precision above 92%, along with a detection speed of 55 frames per second (fps). Nowadays, gesture-controlled human-machine interfaces (HMIs) have attracted a great deal of interest, particularly in virtual reality, gaming, and automotive interfaces. However, constructing the interfaces is challenging because of the uninterrupted succession of frame sequences in the real-time video stream. Only a handful of low-cost gesture-controlled HMI are available. In [12], the authors have shown a gesture recognition system based on CNN models and a vision transformer. Moreover, they have depicted a gesture-controlled HMI to operate VLC, Spotify, and 2D-Mario-Bros game in real-time by using the correctly classified gesture classes. A few issues with their suggested approach include poor background, lower recognition rate (%), a longer inference time, a lack of resilience, etc. For instance, slightly hand movement can lead to miss-classification of gestures, while controlling the applications in real-time. So, to address these issues, in this work, a robust hand gesture recognition system and a gesture-controlled HMI have been depicted. Here the gesture-controlled interface can map the correctly classified gestures to put commands/specific actions to control multimedia applications with lower latency and higher detection speed in real-time.

3 Suggested channel-pruned YOLOv5s for gesture classification

This section details our proposed scheme for developing a novel HMI via a robust hand gesture recognition system using a channel-pruned YOLOv5s model to operate multimedia applications in real-time scenarios. Our proposed methodology consists of five vital stages: (1) data acquisition, (2) pre-processing phase including data annotation, (3) utilization of the YOLOv5 model for both detection and classification tasks, (4) pruning of the YOLOv5s model to reduce the number of parameters and model size (in MB), (5) utilize the lightweight pruned model to design a gesture-controlled HMI in real-time scenarios. The block diagram of this framework is depicted in Figure 1, and the steps followed to build the channel pruned YOLOv5s model are illustrated in Algorithm 1. All the stages of building the model framework are described as follows:

Algorithm 1 Suggested channel-pruned YOLOv5s model

```

1: Input: Load the trained YOLOv5s model
2: Output: Lightweight channel-pruned YOLOv5s model
3: Let the channel scaling factors be  $\gamma_1, \gamma_2, \dots, \gamma_n$ 
4: Initialize epoch  $\leftarrow 1$ , epochs  $\leftarrow 60$ , pruning rate  $\leftarrow 0.20$ 
5: Start sparse training with sparsity regularization
6: while epoch  $\leq$  epochs do
7:   Step 1: Sort BN scaling factors  $\gamma$  in ascending order
8:    $sl \leftarrow$  sorted list of  $\gamma$  values
9:   Step 2: Compute the maximum allowable pruning threshold to avoid pruning all channels in any BN layer
10:  Step 3: Compute threshold index
11:  threshold index  $\leftarrow \lfloor \text{len}(sl) \times \text{pruning rate} \rfloor$ 
12:  Step 4: Compute pruning threshold
13:  pruning threshold  $\leftarrow sl[\text{threshold index}]$ 
14:  while  $sl \neq \emptyset$  do
15:     $f \leftarrow sl[0]$ 
16:    if  $f <$  pruning threshold then
17:      Eliminate the channel whose scaling factor is  $f$ 
18:    end if
19:     $sl \leftarrow sl \setminus \{f\}$ 
20:  end while
21:  epoch  $\leftarrow$  epoch + 1
22: end while
23: return channel-pruned YOLOv5s model

```

3.1 Gesture detection based on YOLOv5-small as baseline model

YOLO [7] is a one-stage network for object recognition that transforms object classification into a regression problem. Although recent YOLO versions such as YOLOv7, YOLOv8, YOLOv9, YOLOv10 have depicted remarkable detection performance, but they introduce higher computational complexity and excessive training time. So in our experiment, YOLOv5 version has been considered for its compact size and fast inference speed. Among the four variants of YOLOv5 models (small, mega, large, and extra large), YOLOv5-small (YOLOv5s) is considered the fastest and performs best on two popular object detection datasets, Pascal VOC [13] and Microsoft COCO [14] datasets. The three components of the YOLOv5s model are: (1) backbone, (2) neck, and (3) detection part. A brief description of each component is provided below.

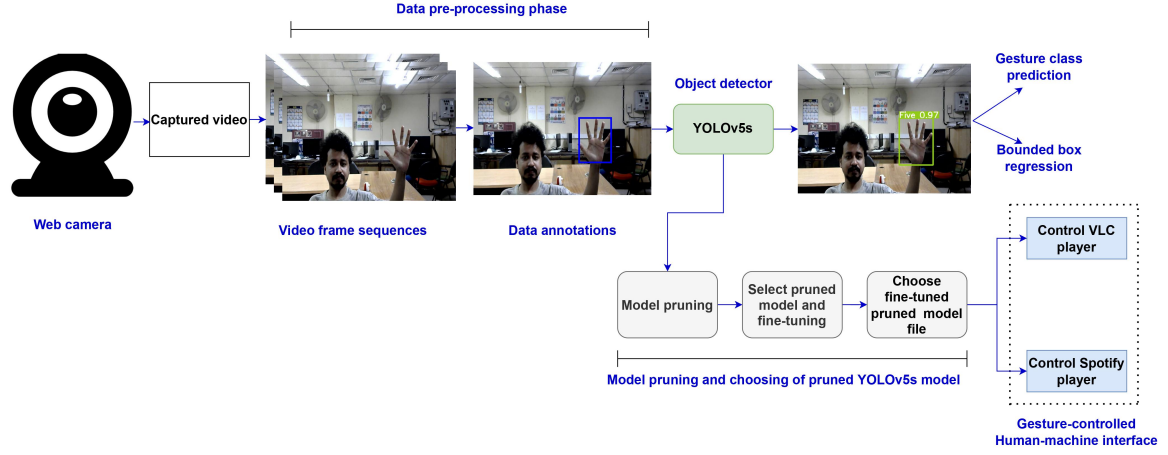


Fig. 1: Block diagram of our suggested scheme.

Backbone layer

Three components make up the backbone layer [7][11]: (1) the focus layer, (2) CSPDarkNet, and (3) SPP. The backbone network of YOLOv5 is built upon CSPDarkNet53, which is a modified version of Darknet53 featuring a ‘cross-stage partial network’ (CSP) architecture aimed at enhancing the data transmission capabilities of the network.

Neck layer

The FPN and PAN structures are used to combine features from different levels of the backbone via bottom-up and top-down pathways in order to improve the distribution of semantic features and locate data.

Detection layer

The detection layer in YOLOv5s estimates the likelihood of each object belonging to a specific class. The YOLOv5s model comprises three detection layers, each corresponding to one of the three processing scales employed for the input image. The final output is generated by combining the results of these three detection layers through concatenation.

In our experiment, the YOLOv5s model was initialized with the model parameters, which were pre-trained on the COCO dataset [14]. The gesture images are scaled to 480×480 to fine-tune the YOLOv5s-based gesture detection model. The learning rate, batch size, and epochs are set to 0.01, 32, and 60. SGD optimizer is used to optimize the network parameters iteratively.

3.2 Channel-Pruned YOLOv5s

In the channel pruning algorithm, at first, the batch normalization (BN) layer is introduced with the scaling factor (denoted as γ). The channels whose corresponding scaling factors (γ) are below the pruning threshold values will be eliminated. The main stages of channel

pruning are (i) sparsity training, (ii) channel pruning, and (iii) fine-tuning of the pruned model.

3.2.1 Sparse training

During sparse training, the scaling factor of the BN layers of the YOLOv5s model was subjected to the L1 norm to make the model more sparse. During sparse training, both scaling factors and weight parameters are simultaneously trained. After sparse training, the values of γ are approaching 0, indicating that the associated channels with smaller scaling factors will be eliminated.

3.2.2 Channel pruning

After sparse training, the γ values from the BN layers are sorted to find the threshold value. The next phase is to identify the position in the sorted list of BN weights at which the channels will be pruned using the pruning threshold index. The channels whose scaling factors (γ) are less than the pruning threshold, will be eliminated.

In the experiment, the pruning rate has been set from 10% to 20% as higher pruning rates potentially degrade model performance due to the reduction of the number of trainable parameters. We have compared the performance of YOLOv5s model on the ‘NITR-HGR’ dataset at pruning rates varying from 10% to 20% and the comparison results (after fine-tuning of channel pruned YOLOv5s models) have been shown in Table 1.

3.2.3 Fine-tuning of the pruned model

The pruning rate reduces the detection accuracy after channel pruning due to reducing the number of channels. So, the channel-pruned model is further retrained with the hyper-parameter settings to enhance the detection accuracy. Like the training phase, in the fine-tuning phase, the batch size and number of epochs are both configured to 32 and 60, respectively.

3.3 Mathematical formulation

In the case of the channel pruning method [15, 16], the scaling factor γ is associated with each convolution layer after the introduction of batch normalization. The optimization function for pruning, specifically targeting BN layers, is focused on pruning the network by removing less important channels (filters) based on the scaling factors from BN layers. Let sl be the sorted list of γ values in ascending order and α be the pruning rate i.e. is the percentage of channels to be pruned. Let τ_i be the threshold index i.e. is the index that determines the pruning threshold in the sorted list of γ values. We find the threshold index, τ_i , followed by calculating the pruning threshold, τ . Both the formulas to calculate the threshold index and pruning threshold are shown in Equations 2 and 3.

$$sl = \text{sort}(\gamma_1, \gamma_2, \dots, \gamma_n) \quad (1)$$

$$\tau_i = \lfloor \text{len}(sl) \times \alpha \rfloor \quad (2)$$

$$\tau = sl[\tau_i] \quad (3)$$

This loss function [17] during sparse training involves two key components: the primary loss function and a regularization term that encourages sparsity. The $\mathcal{L}(X, Y; \theta)$ is the loss function which is the standard loss used for training the model.

Let γ_l represents the scaling factors for l layers in the BatchNorm layers. The second term, the Regularization term promotes sparsity in the model. The regularization term $\lambda \cdot \sum_{l=1}^L \|\gamma_l\|_1$ encourages the scaling factors γ_l to become sparse, effectively promoting the removal of less important channels. If γ_l is small or zero, it suggests that the corresponding channel can be pruned according to the pruning threshold (τ), which is a value below which channels are considered unimportant and can be pruned. Here, the optimization function becomes,

$$\min_{\theta} (\mathcal{L}(X, Y; \theta) + \lambda \cdot \sum_{l=1}^L \|\gamma_l\|_1) \quad ; \gamma_l > \tau, \forall l \quad (4)$$

However the sparsity is achieved through pruning by applying a mask on the BatchNorm (BN) layers based on the γ values (scaling factors in BatchNorm). The mask is applied to prune specific channels. Let $\mathcal{L}(\theta \circ M)$ ($\circ$ represents the composition operator) represents the loss function after applying pruning mask M to the model parameters θ where M is the mask that controls which channels are to be kept (denoted as 1) and which are to be pruned (denoted as 0). The M values are determined based on the absolute values of the BatchNorm scaling factors γ . The revised optimisation function becomes,

$$\min_{\theta} (\mathcal{L}(\theta \circ M) + \lambda \cdot \sum_{l=1}^L \|\gamma_l\|_1) \quad ; \gamma_l > \tau, \forall l \quad (5)$$

where,

$$M_l = \begin{cases} 1, & \text{if } |\gamma_l| \geq \tau \\ 0, & \text{otherwise} \end{cases}$$

4 Experimental Evaluation

To evaluate the effectiveness of the suggested framework, simulations have been carried out on two annotated hand gesture datasets: one private dataset, ‘NITR-hand gesture’ (NITR-HGR) [11] (mentioned in our previously published article [11]), and ‘American sign language’ (ASL) (public dataset) datasets. The data collection and annotation details have been explored in subsection 4.1.

4.1 Dataset collection and details

The NITR-hand gesture dataset [11] was created with the assistance of five individuals from ‘NIT Rourkela’ (four men and one woman). Initially, the recordings were captured using the web camera at a frame rate ranging from 25 to 30 fps. The frames were subsequently extracted from the videos. Consequently, the bounding box annotates the gesture region within the frames. There are thirteen different classes (example, Three, Two, Thumb, Palm etc.) of gesture images in this dataset. There are 250 images in each gesture class, each having a resolution of (640 × 480) pixels. The images were captured from different angles and distances. Figure 2 shows some examples of NITR-hand gesture datasets.

The ASL dataset consists of 29 gesture classes, including 26 for the letters A through Z, and 3 additional classes. We have collected 2844 annotated gesture samples from the Roboflow [18].



Fig. 2: Gesture samples from the NITR-hand gesture dataset.

4.1.1 Results on NITR-HGR dataset

To assess the efficacy of the channel pruned-YOLOv5s-based gesture detection model, it is compared with the YOLOv5s baseline model on the NITR-HGR dataset.

The performance analysis is shown in Table 1. The performance of the suggested channel-pruned YOLOv5s-based HGR system under various lighting conditions is shown in Figure 3.

Performance analysis of channel pruned YOLOv5s model

Table 1 demonstrates the performance comparison of the baseline YOLOv5s model and the pruned YOLOv5s models after fine-tuning on the validation set of NITR-HGR dataset at different pruning rates (%). Specifically, the model parameters, FLOPs, and the model size (MB) are used to compare the efficiency of various models. FLOPs are used to quantify the computational complexity of models. In Table 1, it is noticed that the GFLOPs and the number of parameters are reduced after using different pruning rates (%). It is observed in Table 1 that the pruning rate of 10% to 20% resulted in a reduction of the number of GFLOPs in pruned YOLOv5s models by 13%, 18%, and 25% respectively, as compared to the baseline YOLOv5s model. In case of inference time, it has been reduced by 0.9 to 1.1 ms by using a different pruning rate. Similarly, the model size is reduced by increasing the pruning rate (%). When compared to the baseline YOLOv5s model, the fine-tuned pruned model performs well in terms of inference time (in ms) and detection speed (in fps), despite a reduction in detection accuracy (in mAP). It assists in faster detection and classification of the gesture class in real-time scenarios.

Performance comparison of object detection models on NITR-HGR

To further analyze the performance of the suggested model, 15 object detection models were evaluated on the NITR-HGR dataset. These models included Faster RCNN (with Inception-V2 as the backbone), SSD, YOLOv3, YOLOv5s (with ghost module), YOLOv5s (with transformer module), two larger versions of the YOLOv5 model such as YOLOv5-mega, YOLOv5-large, YOLOv7, YOLOv8s, YOLOv9s, YOLOv10s, YOLOv11s and three frozen YOLOv5s models [11] (after freezing different layers). Table 2 depicts the comparison analysis of the suggested model with 15 different models in terms of evaluation metrics. It is observed in Table 2, that the precision, recall, F1-score, and mAP values of our suggested framework are higher than the other state-of-the-art works presented by [19], [20], [21] and [11]. It is also noticed in Table 2 that both YOLOv7 and our suggested fine-tuned pruned YOLOv5s (with pruning rate 10%) have the highest precision compared to other models, but its size is larger than our suggested framework. It is observed that the size of the YOLOv5s model with ghost module is lowest compared to other models, but the precision, recall, F1-score, and mAP of this model are less than YOLOv5s (with transformer module), YOLOv7 and suggested fine-tuned pruned YOLOv5s model. The proposed pruned YOLOv5s models after fine-tuning

provide the best balance between detection performance and model efficiency. The pruned version 10% achieved the highest F-score of 99.65% and mAP (50-95%) of 92.50% with a model size of 12 MB. It outperformed both the frozen-layer variants YOLOv5s and other models from the YOLO series. Even with 20% pruning, the proposed approach maintained an F-score of 98.00% while reducing the model size to 10.7 MB, indicating strong robustness to parameter reduction. It is observed that an increase in the pruning rate leads to a slight reduction in detection accuracy, reflecting the inherent trade-off between model size and accuracy. However, the comparison analysis demonstrates that our suggested 10% pruned approach outperformed the other state-of-the-art works in terms of precision, recall and mAP scores.

4.1.2 Results on ASL dataset

This subsection provides an overview of the experiments performed on the ASL dataset in order to address the 29-gesture class problem. The experiments were performed using our proposed model on the ASL dataset, with the default input size of (224×224) . Five object detection models were trained on the labeled ASL dataset. These models are Faster RCNN (with Inception-V2 as the backbone), three larger versions of the YOLOv5 model such as YOLOv5-mega, YOLOv5-large, YOLOv5-extralarge, and YOLOv7. It is noticed in Table 3, that the precision of YOLOv5l is larger than other used models but less than our suggested 10% pruned YOLOv5s model. Table 3 shows that the performance of the suggested model has outperformed the other scheme presented by [19] and the four used object detection models.

4.2 Building of novel HCI by utilizing pruned YOLOv5s model

This contribution shows a gesture-controlled HMI to control VLC and Spotify players in real-time. After obtaining the fine-tuned, pruned YOLOv5s model (trained on the ‘NITR-HGR’ dataset), the model’s weight file (in pytorch format) is further utilized to operate two applications (VLC media player and Spotify player) in real-time scenarios with the correctly classified gesture classes as input commands. The details of our gesture-controlled VLC and Spotify player have been demonstrated in the following subsection.

VLC player and Spotify player control

In this work, the ‘playerctl’ [22] command-line interface has been utilized to control VLC and Spotify player’s functions, such as play, pause, proceed to next video, volume up/down, etc. To operate the VLC player using gesture commands, the keyboard event of the keys is triggered after obtaining the predicted gesture classes. For example, the gesture ‘Ok’ has been set to play a video. Similarly, gestures ‘Two’ and ‘Three’ denote to move the next or previous media in a playlist. We have used predicted gesture classes to trigger each function



Fig. 3: Performance of our suggested scheme in various light conditions, (A) Normal-light, (B) Low-light, (C) dark-light.

Table 1: Performance comparison of YOLOv5s model for gesture detection after fine-tuning of channel pruned models with different pruning rates (%) on NITR-HGR dataset.

Models	Pruning rate (%)	Number of parameters (in millions)	GFlops	Model size (MB)	F-score	mAP 50-95	Inference time (in ms)	Average detection speed (in fps)
Baseline YOLOv5s	0	7.05	15.1	14.5	99.65	92.50	7.1	54
Suggested Pruned YOLOv5s model (after fine-tuning)								
10%-pruned YOLOv5s	10	6.2	13.1	12	99.65	92.50	6.2	59
15%-pruned YOLOv5s	15	5.8	12.1	11.5	99.60	91.10	6.1	60
20%-pruned YOLOv5s	20	5.3	11.2	10.7	99.00	90.80	6.0	62

of the VLC and Spotify player five times to achieve promising outcomes.

Table 4 displays the performance of the gesture-controlled VLC and Spotify player, including the average response time (in milliseconds), hit and miss counts. Two demos of this gesture-controlled VLC and Spotify player have been shown in Figures 4.

Improvement of the robustness of the gesture-controlled VLC and Spotify player

It is noticed that while controlling the VLC/Spotify player with the correctly classified gesture commands, it shows poor results for various functions, such as volume up/down, going to the next, or previous media because of the higher processing of frames sequence in real-time with respect to time. The core functions of the VLC and Spotify players are playing, pausing, volume up/down, and switching from the next/previous media/song, etc. Among these functions, playing/pausing are considered as ‘discrete’, and the other remaining are considered as ‘continuous’ functions. A higher frame rate may result in the skipping of desired media from the playlist when the player is being operated using the corresponding gesture commands, which becomes an issue. For instance, if a user indicates the corresponding gesture ‘Two’ in front of the camera and wishes to proceed to the next song on the gesture-controlled Spotify player, it will bypass the specified song and proceed to multiple others. The user experiences the same result when adjusting the volume using the corresponding gesture commands. To combat these issues, we have delayed the program’s execution by a fraction of the time (in seconds). In our experiment, we call the ‘time sleep ()’ function. It reduces the higher frame rate in real-time testing tasks, which benefits the user from triggering continuous functions (such as play/pause, volume up/down,

etc) in the Spotify/VLC player with the corresponding gesture commands. By incorporating the sleep() function after each label (gesture class) mapping, the synchronization is maintained between mapping the right class and activating a continuous function of VLC and Spotify player in real-time. Hence, it brings robustness to the suggested gesture-controlled VLC/Spotify player.

5 Discussions

In this article, a robust hand gesture recognition system is depicted by using the channel-pruned YOLOv5s model. The suggested model is evaluated on two annotated hand gesture datasets such as the ‘NITR-HGR’ (private) and ‘ASL’ (public). In this contribution, the YOLOv5s model is chosen for the gesture detection task. Next, the YOLOv5s-based gesture detection model is optimized by using a channel-pruning algorithm to remove the unnecessary channels to obtain a lightweight model for faster inference tasks. It is observed that after pruning, the detection accuracy is slightly reduced, so the pruned model has been further trained using hyper-parameter settings to ensure better detection accuracy. We have demonstrated the performance comparison of the YOLOv5s model for gesture detection after fine-tuning channel-pruned models with different pruning rates varying from (10-20)% on ‘NITR-HGR’ dataset. It is noticed that mAP, F1-score (%) have been reduced after increasing the pruning rate, but the detection speed has been increased due to the accuracy-speed trade-off. Our suggested channel-pruned model (with different pruning rates (%)) has been compared with 12 object detection models and it is observed in Table 2, that our suggested scheme outperformed the other object detection models in terms of precision (%), recall (%) and mAP50-95 (%).

Table 2: Experimental results of different object detection-based models on NITR-HGR dataset.

Models	Precision (%)	Recall (%)	F-score (%)	mAP50 (%)	mAP 50-95 (%)	Parameters (in millions)	Model size (in MB)
Faster RCNN (Inception V2) [19]	88.32	87.60	87.95	90.45	84.60	50.62	135
SSD (320 × 320)	90.52	89.65	90.08	92.50	84.65	15.7	44
YOLOv3 [20] (DarkNet53 as backbone)	94.82	95.12	94.97	90.80	84.10	62.5	235
YOLOv5s (Ghost module) [21]	98.6	98.20	98.40	92.80	84.20	3.7	9.40
YOLOv5s after freezing 3 layers [11]	99.50	99.35	99.42	99.26	92.60	7.01	13.80
YOLOv5s after freezing 5 layers [11]	99.45	99.35	99.40	99.23	92.50	6.80	13.30
YOLOv5s after freezing 7 layers [11]	98.90	98.50	98.70	98.95	91.16	5.90	12.50
YOLOv5s (Transformer module)	99.60	99.50	99.55	99.40	91.06	7.05	15
YOLOv5m	96.40	95.70	96.05	98.4	85.32	21	40.7
YOLOv5l	95.24	94.12	94.68	97.67	84.46	46	92.7
YOLOv7	99.60	98.40	99.00	99.40	92.45	37.2	74.9
YOLOv8s	98.20	98.80	98.50	98.40	90.64	11.21	22.10
YOLOv9s	98.40	98.10	98.25	98.90	90.90	7.3	15.68
YOLOv10s	98.50	98.40	98.45	99.00	90.50	8.07	15.85
YOLOv11s	96.50	96.20	96.35	99.10	91.10	21.6	18.42
Our suggested Pruned YOLOv5s (after fine-tuning)							
10% Pruned YOLOv5s	99.70	99.60	99.65	99.50	92.50	6.2	12
15% Pruned YOLOv5s	99.50	99.70	99.60	99.40	91.10	5.8	11.5
20% Pruned YOLOv5s	98.50	97.50	98.00	99.30	90.80	5.3	10.7

Table 3: Comparison among various object detection models on ASL dataset.

Models	mAP50(%)	mAP50-95(%)	Precision(%)	Recall(%)	F-score(%)
Faster-RCNN (Inception V2) [19]	94.20	83.45	89.60	89.20	89.40
YOLOv5m	99.10	88.70	98.40	98.80	98.60
YOLOv5l	99.10	88.20	98.70	98.50	98.60
YOLOv5x	95.6	85.3	95.4	94.9	95.2
YOLOv7	98.5	88.5	97.6	96.8	97.2
Proposed channel pruned YOLOv5s (after fine-tuning)					
Pruned YOLOv5s (10% pruned)	99.40	89.20	98.80	98.60	98.70
Pruned YOLOv5s (15% pruned)	99.30	88.70	98.60	98.70	98.65
Pruned YOLOv5s (20% pruned)	99.30	87.20	98.40	98.90	98.65

6 Conclusion

This paper presents a robust hand gesture recognition system based on the suggested lightweight channel-pruned YOLOv5s model under complex backgrounds and light-varying conditions. This study also depicts a gesture-controlled HMI to control two applications, including VLC and Spotify player, using predicted gesture class labels in real-time. It is noticed that after introducing the channel-pruned YOLOv5s model, the detection accuracy is slightly decreased compared to the baseline YOLOv5s model. However, the model complexity and the number of trainable parameters have been decreased compared to the original YOLOv5s model with different pruning rates. So to enhance the detection accuracy, the pruned model is

fine-tuned with the hyper-parameter settings. In our future work, we will build dynamic hand gesture recognition and utilize the correctly predicted gesture classes to control the multimedia applications in real time with quick response time and higher inference speed. Furthermore, the scope of this work can be expanded by integrating additional modalities, such as eye-gaze tracking and facial expression, to provide a multimodal fusion-based interface.

7 Declarations

- **Funding:** No funding was received.
- **Ethics approval:** Not applicable.

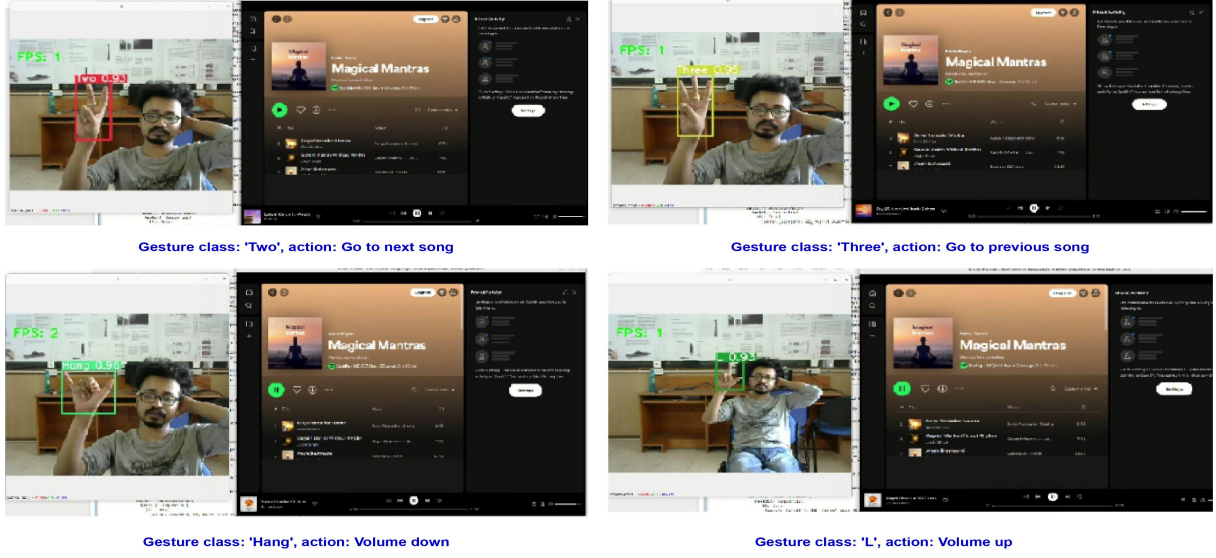


Fig. 4: Depiction of the gesture-controlled Spotify player in real-time.

Table 4: Analysis of the real-time performance of gesture-controlled VLC and Spotify player

Performance analysis of gesture-controlled VLC player					
Action	Gesture classes	Hit counts	Miss counts	Detection rate (%)	Average Response time (ms)
Play	Ok	5	0	100	5.0
Volume Up	L	4	1	80	8.3
Volume down	Hang	4	1	80	8.2
Go to previous video	Three	3	2	60	10.3
Go to next video	Two	3	2	60	12.5
Pause	Fist	5	0	100	7.3
Performance analysis of gesture-controlled Spotify player					
Play	Ok	5	0	100	6.0
Pause	Fist	5	0	100	7.3
Volume Up	L	4	1	80	10.9
Volume Down	Hang	4	1	80	11.5
Proceed to next track	Two	3	2	60	12.3
Return to previous song	Three	3	2	60	12.9

- **Code availability:** <https://shorturl.at/uUh2l>
- **Deployed Model:** <https://shorturl.at/MsFHL>
- **Competing Interests:** The authors have no competing interests.

References

- [1] Singha, J., Roy, A., Laskar, R.H.: Dynamic hand gesture recognition using vision-based approach for human-computer interaction. *Neural Computing and Applications* **29**(4), 1129–1141 (2018)
- [2] Al Farid, F., Hashim, N., Abdullah, J., Bhuiyan, M.R., Shahida Mohd Isa, W.N., Uddin, J., Haque, M.A., Husen, M.N.: A structured and methodological review on vision-based hand gesture recognition system. *Journal of Imaging* **8**(6), 153 (2022)
- [3] Huang, D.-Y., Hu, W.-C., Chang, S.-H.: Vision-based hand gesture recognition using pca+ gabor filters and svm. In: 2009 Fifth

- International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 1–4 (2009). IEEE
- [4] Rahman, M.H., Afrin, J., *et al.*: Hand gesture recognition using multiclass support vector machine. *International Journal of Computer Applications* **74**(1), 39–43 (2013)
 - [5] Sharma, S., Singh, S.: Vision-based hand gesture recognition using deep learning for the interpretation of sign language. *Expert Systems with Applications* **182**, 115657 (2021)
 - [6] Sen, A., Mishra, T.K., Dash, R.: A novel hand gesture detection and recognition system based on ensemble-based convolutional neural network. *Multimedia Tools and Applications*, 1–24 (2022)
 - [7] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788 (2016)
 - [8] Girshick, R.: Fast r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448 (2015)
 - [9] Bose, S.R., Kumar, V.S.: Efficient inception v2 based deep convolutional neural network for real-time hand action recognition. *IET Image Processing* **14**(4), 688–696 (2020)
 - [10] Yu, X., Yuan, Y.: Hand gesture recognition based on faster-rcnn deep learning. *Journal of Computers* **14**(2), 101–110 (2019)
 - [11] Sen, A., Dombe, S., Mishra, T.K., Dash, R.: Hgr-fyolo: a robust hand gesture recognition system for the normal and physically impaired person using frozen yolov5. *Multimedia Tools and Applications*, 1–19 (2024)
 - [12] Sen, A., Mishra, T.K., Dash, R.: Deep learning-based hand gesture recognition system and design of a human–machine interface. *Neural Processing Letters*, 1–28 (2023)
 - [13] Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* **88**, 303–308 (2009)
 - [14] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: *Computer Vision–ECCV: 13th European Conference, Zurich, Switzerland, Proceedings, Part V 13*, pp. 740–755 (2014)
 - [15] Chen, Y., Wang, Z.: An effective information theoretic framework for channel pruning. *arXiv preprint arXiv:2408.16772* (2024)
 - [16] Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems* **31** (2018)
 - [17] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744 (2017)
 - [18] Recognition, A.: ASL recognition Dataset. Roboflow. visited on 2023-04-14 (2022)
 - [19] Rubin Bose, S., Sathiesh Kumar, V.: Hand gesture recognition using faster r-cnn inception v2 model. In: *AIR*, pp. 19–1 (2019)
 - [20] Mujahid, A., Awan, M.J., Yasin, A., Mohammed, M.A., Damaševičius, R., Maskeliūnas, R., Abdulkareem, K.H.: Real-time hand gesture recognition based on deep learning yolov3 model. *Applied Sciences* **11**(9), 4164 (2021)
 - [21] Hu, D., Zhu, J., Liu, J., Wang, J., Zhang, X.: Gesture recognition based on modified yolov5s. *IET Image Processing* **16**(8), 2124–2132 (2022)
 - [22] Altdesktop, p.: Altdesktop/playerctl: Media Player Command-Line Controller for VLC, MPV, Rhythmbox, web browsers, cmus, MPD, Spotify and others.