A DEEP LEARNING APPROACH FOR THE ANALYSIS AND DETECTION OF OBJECT IN VIDEO FRAMES USING YOLO FASTER RCNN

Krishna Kumar[1], Dr. C.L.P Gupta[2], Dr. Krishan Kumar[3]
[1]Department of CSE, BIET, Lucknow, kk_gkp@rediffmail.com, [2]Department of CSE, BIET, Lucknow, clpgupta@gmail.com, [3]Department of Computer Science, Gurukul Kangri, Deemed to be University), Haridwar, India, krishan.kumar@gkv.ac.in

**Corresponding Author: -*
kk_gkp@rediffmail.com

Abstract: - *Object detection often refers to a collection of generic computer vision tasks which potentially identifies objects from the given video inputs. As object detection combines two main tasks like image classification and object localisation which eventually identifies one or more objects in a specified image frame. The space in which this research is very popular is one where researchers continue developing new aspects in detecting objects, and in various areas including autonomous driving, health-care monitoring, anomaly detection etc. Traditional object detection is done using shallow features and handcrafted architecture which eventually doesn't give effective results. So, to overcome this, the use of advanced technology such as Deep learning comes into play as it has a wide hand in this field. Thereby this paper brings an effective object detection model from video frames in which initially a)Data Collection from ImageNet VID and CIFAR-10 video analysis b) Feature extraction using a convolutional autoencoder c) feature selection using SE-block d) Classification using integration of Yolo-Faster-RCNN. The study shows that the proposed method outperforms with 95% accuracy when compared with other state-of-art models.*

Keywords*: - Deep learning, Classification, RCNN Video Analysis, Yolo.*

1. INTRODUCTION

Many applications are utilizing video, for detecting pedestrians, recognizing unusual behaviour in parking lots and so on. Nowadays, retrieving moving objects and automating the analysis of video is more frequently used. An example of multimedia data is video, which combines a variety of types of data like text, picture, metadata, visual, and audio. The primary aim of video data analysis is to recognize and track moving objects such as people across frames. Security, surveillance, entertainment, medical and legal applications, as well as medical education and sports make use of Video Data Mining. Video data mining is based on finding and analyzing patterns in massive amounts of video data. The video is made up of a series of images. The video material may be divided into two categories: i) low-level feature data, such as colour, texture, and form and ii) high-level feature details, such as audio and video. Syntactic information like video material contains conspicuous objects, their spatial-temporal [2] location and their spatial-temporal connection. Semantic data, explain what is happening in the video, such as spatial features offered by a video frame, position characters moved in the screen etc. The aspects of time characterize a succession of video frames such as the actions of actors and the movement of objects in sequence. The first step in extracting information about the objects in a video is to detect moving objects in video streams. In many computer vision applications, including video surveillance and people annotation, this is the first step. Figure 1a. shows the overall use of emerging technology over object detection. Figure 1b explains most of the research works that happened over the object detection areas.

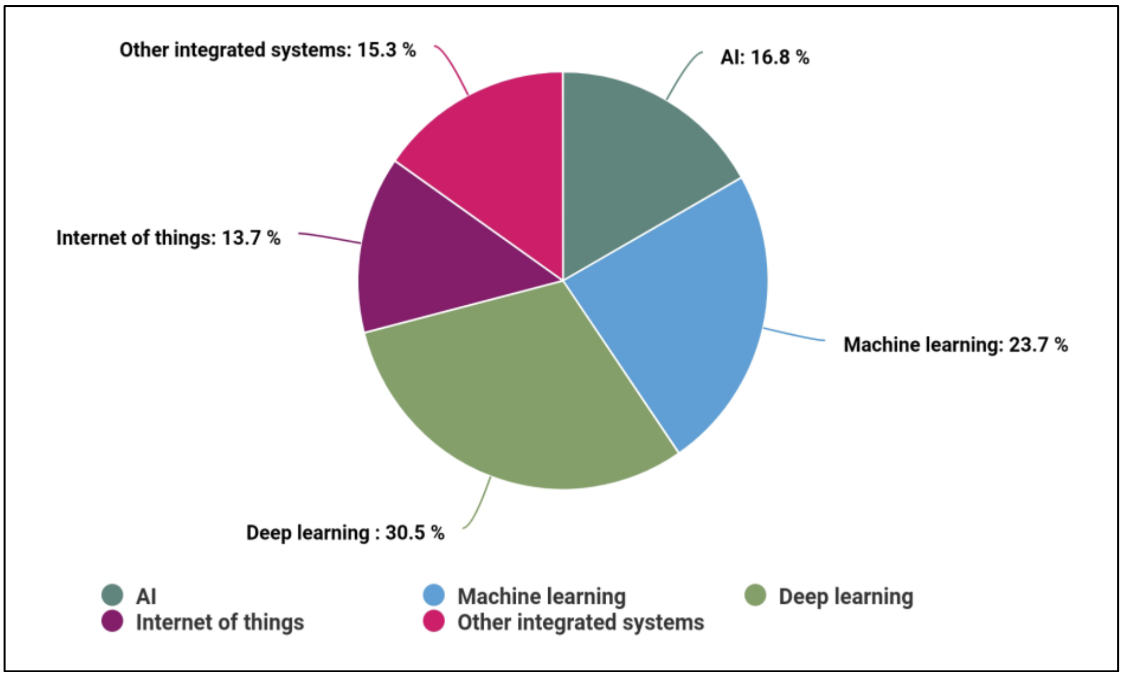


Figure 1a. Most used emerging technology for the Object detection

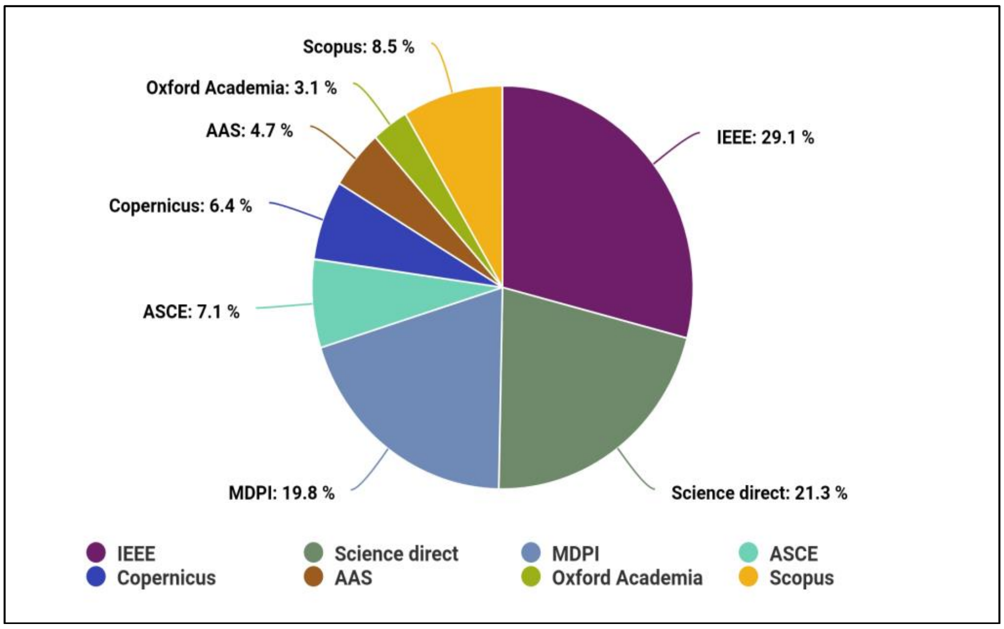


Figure 1b. Papers published regarding object detection

Item detection in computer vision entails finding an object or locating an instance of interest within a collection of suspicious frames. Tracking is the process of locating a specific instance and detecting when it occurs within suspicious frames [3]. Figure 1 shows a basic block diagram for object detection and tracking. The data set is divided into two halves. Trajectory or route tracking is the process of determining a path or trajectory that an object follows in a number of frames. The resulting image is a collection of frames. The dataset consists of 80% training images and 20% testing images. CNN and Yolov3 algorithms are used to locate objects in an image. When the Intersection over Union (IoU) is greater than 0.5, a bounding box is generated over the object. Multi-Object Tracking (MOT) monitors the bounded box in multiple frames at the same time [4]. Bounding boxes are transmitted to neural networks as a reference for tracking. Figure 2 depict the general object detection block.

1.1 Key Highlights

This paper focuses on effective object detection using video frames with following objectives:

- The paper shows an effective object detection model using an integration of Yolo and Faster-RCNN
- Feature extraction and selection is happening through Convolution autoencoder and SE block respectively.
- The proposed model will be evaluated with other state-of-models over certain measures.

Organization of the paper: Based on the idea of object detection over video, rest of the paper is divided in following sections:

- Section 2 covers the related works,
- Section 3 focuses on methodology,
- Section 4 describes the performance evaluation,
- And, finally section 5 concludes the paper.

2. Related Works

Fast convolutional neural networks, such as Faster-RCNN, Faster-Faster-RCNN, and Single Shot Detector (SSD), are being used to detect objects based on Chandan et al. (2018) (Real-time object recognition and tracking using Deep Learning and OpenCV). Deep learning combines SSDs with Mobile Nets to identify and track objects effectively. Fast-Faster-RCNN and SSD have superior accuracy, whereas YOLO is more effective when speed is a priority over accuracy [5]. This technique detects objects quickly and efficiently without sacrificing performance.

Traditional 2D object detection results in bounding boxes that are axis-aligned (x, y) with two dimensions (w, h), according to Shreyas et al. Whereas 3D bounding boxes have six degrees of freedom: 3D physical size (w, h, l), 3D centre (x, y, z). Autonomous driving, for example, relies on the depth information provided by 3D object detection to complete crucial tasks in computer vision. The depth information required for 2D object recognition and tracking methods is lacking. In several areas where 3D object-tracking and identification are employed, more information is needed to get exact findings [6]. The paper describes several methods for tracking and detecting 3D objects for a wide range of computer vision applications including robotics, driving, space exploration, and military use.

In 2019 (A real-time object detection algorithm for video), Lu et al. proposed a real-time object detection method using YOLO networks. To remove the effects of background images, they used image preprocessing. To identify object faces, they used the Fast YOLO model. Researchers in this study improved the Yolo network through a tiny convolution procedure created by combining Google Inception Nets (GoogleNets) and Google Inception networks using GoogleNet architecture, which could reduce the number of parameters and drastically reduce the time required for object detection [7]. Real-time detection of objects in videos can be accomplished using this Fast YOLO method.

The researchers used deep learning to identify small objects in video surveillance, using the deep learning method Object Detection with Binary Classifiers, a two-level deep learning method intended to recognize small objects. Hernandez et al. (2020) proposed applying the deep learning method Object Detection with Binary Classifiers to identify small objects. In the first level, candidate regions are selected from the input frame, In the second level, CNN-classifiers are used with One-Versus-All or One-Versus-One binarizations. In this study, they determined that it is hard to detect weapons in video surveillance if they are handled with a hand. In experiments, the proposed methodology minimizes the frequency of false positives compared with the baseline multiclass detection model[8]. The proposed methodology created six databases: a pistol, a knife, a phone, a bill, a pocketbook, and a card.

Raj et al. (2021), detected objects by dividing the video into individual frames and processing each frame individually. Preprocessing the image and removing the noise was the first step in obtaining the relevant shape and texture of the image. The next step was edge detection and ended with feature extraction. Convolutional layers include many filters, size of filters, pooling layer, and maximum pooling layer. The items can be identified using the activation function Re Lu, which means a higher-level feature-containing shape is fed into the convolution layer. A gradient descent optimization technique is also employed to improve accuracy. ImageNet has been used to test the system [9].

3. Methodology

A suggested architecture for object detection is shown in Figure 3, in which a video is taken as an input and a single frame is selected for object detection. A video frame is first broken into image frames, and then a single frame is used to tect the object of interest [10]. During the first stage of processing, the images are analyzed for their characteristics. he structure and form of the pixels provide enough information to identify the significant elements in an image, so not every pixel is sent to the neural network. The edges of the picture may be recognised using the feature extraction procedure. Then it will be passed for dimensionality reduction where with the help of SE block those features will be selected and finally given to classification for required results.

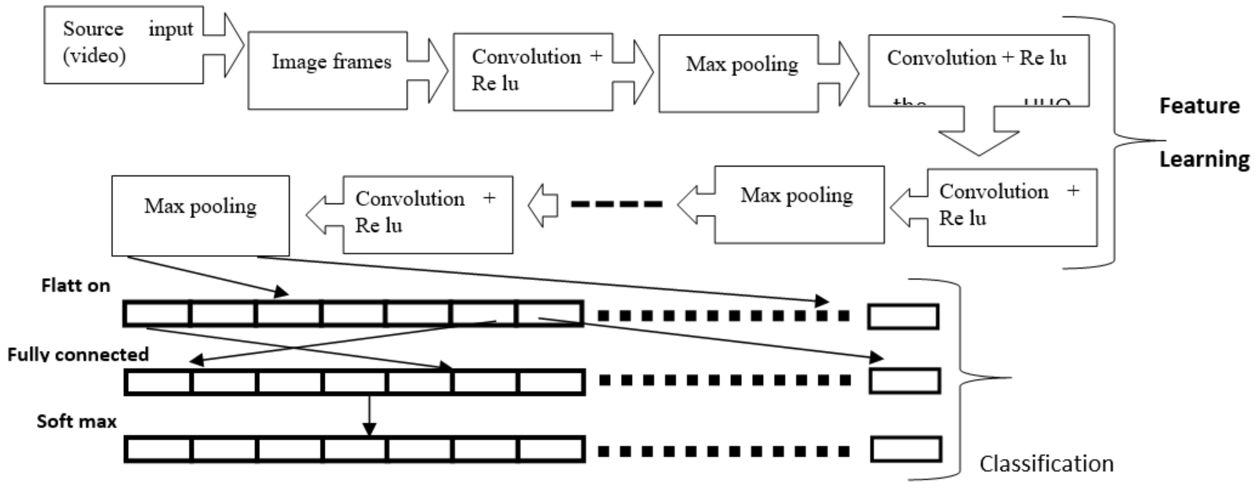


Figure 3: Flow of objective detection

3.1 Data Collection
For effective object detection, deep learning classifiers require sufficient data for training and testing. So, we took 2 datasets (ImageNet VID and CIFAR-10) based on video analysis where these are already been divided into image frames or video frames of various categories.

The most commonly used dataset for video object detection is the ImageNet VID dataset, which consists of two parts: the training set and the validation set and consists of 3862 and 555 video samples respectively[11]. In addition to thirty item categories, this dataset includes a frame rate of 25 or 30 frames per second. The data set also includes a subset of categories found in the ImageNet DET dataset [12]. For the ILSVRC classification problem, we generated a large-scale dataset using the picture collection and annotation technique described in earlier sections. Table 1(a,b) shows the amount of training, validation, and test images across the years of the challenge. Figure 4 shows ImageNet VId instances from different years.

Table 1a. Image Classification

Year	**Train Image**	**Validation image**	**Test image**
ILSVRC2010	1,261,406	50,000	150,000
ILSVRC2011	1,229,413	50,000	100,000
ILSVRC2012-14	1,281,167	50,000	100,000

Table 1b. Multiple object detection localization

Year	**Annotate training images with Bbox**	**Annotated train bboxes**	**Bbox annotations on Val images**	**Val bboxes annotated**	**Bbox annotations are used to test photographs**
ILSVRC2011	315,525	344,233	50,000	55,388	100,000
ILSVRC2012-14	523,966	593,173	50,000	64,058	100,000

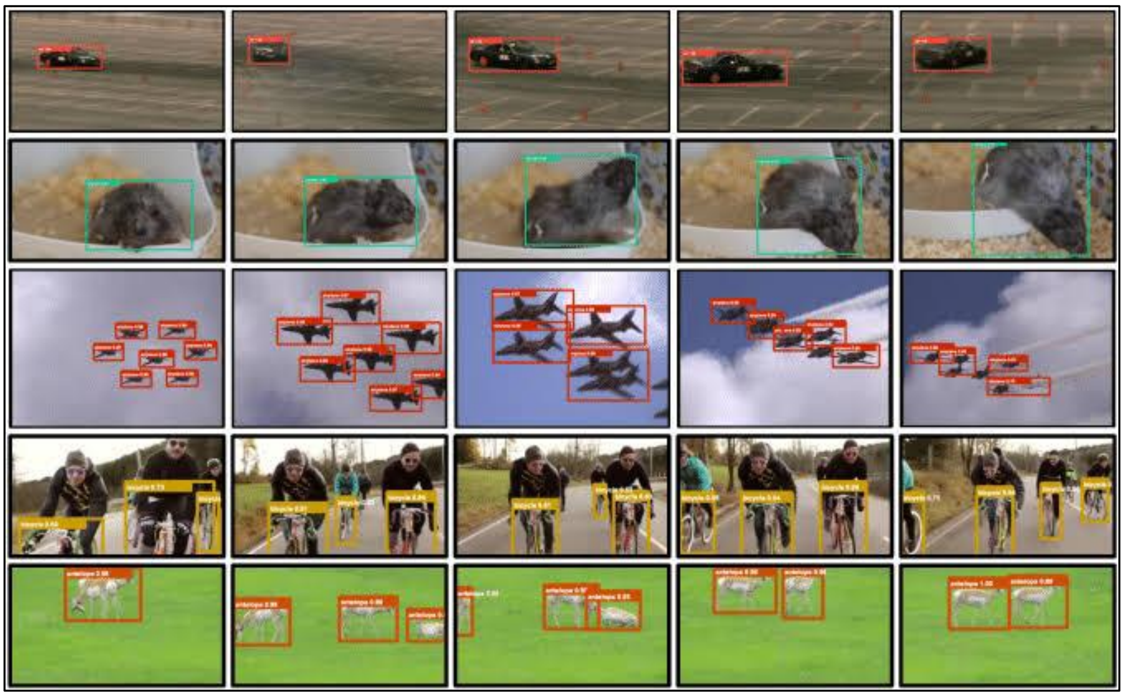


Figure 4. ImageNet VID dataset instances of video frames categories

CIFAR-10 contains 60000 32x32 colour photos divided into ten classes, each with 6000 images. The training portion of the dataset contains 50000 photos while the testing portion contains 10,000 photos. The training portion of the dataset is divided into five training batches, while the testing portion is divided into one. It consists of exactly 1000 photographs randomly chosen from each class in a test batch[13]. The remaining photographs are randomly distributed in training batches, with some batches having more images from each class than others. Figure 5 shows random images from CIFAR-10 cases grouped by class. Each training batch contains exactly 5000 images[14].

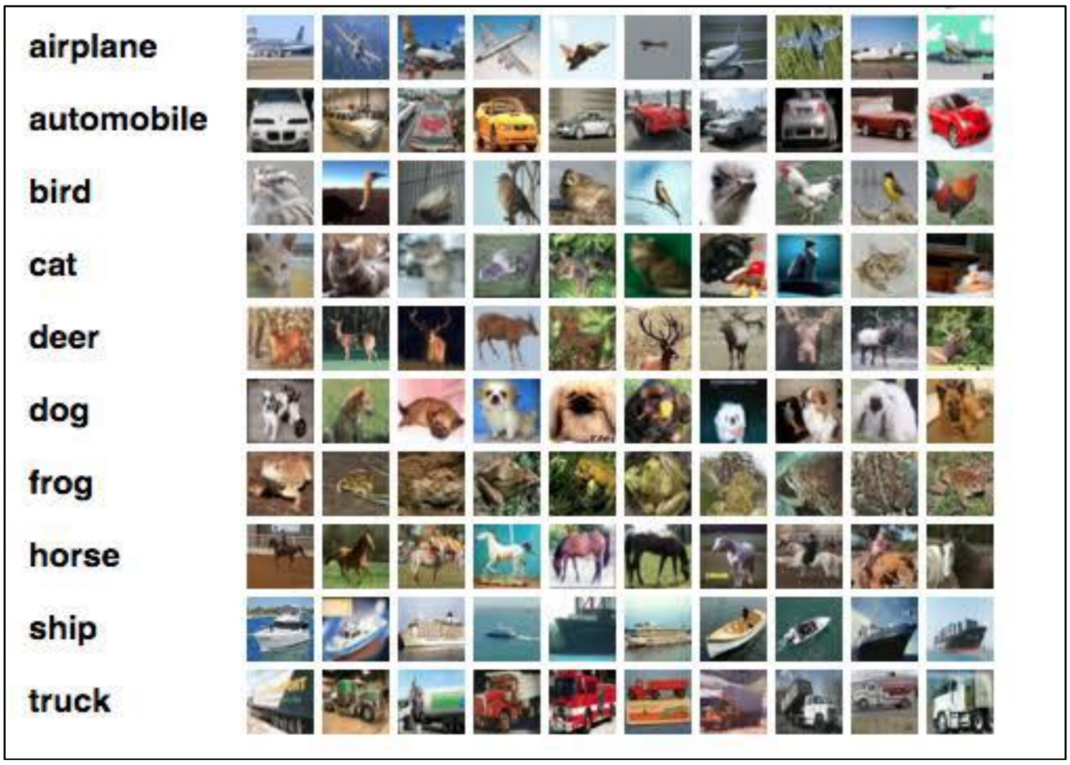


Figure 5. Instances from the CIFAR-10 dataset where random 10 categories are shown

3.2 Preprocessing

Following the entry of a video frame, image cropping is usually the initial step. In this stage, non-interesting areas of the picture will be cropped off so that future processing may concentrate on the regions of interest, lowering the computing cost. The rest of the areas are then subjected to image preprocessing operations. Local operators [15] are the simplest of all preprocessing adjustments since they operate by determining how well the output pixels relate to the input pixels. Let x represent the position in the picture and f(x) represent its value; a local operator in the continuous domain may be expressed by:

$$h(x) = g(f(x)) \qquad (1)$$

As a common local operator, multipliers and adders with constants are used [16], which operate over ranges that may be either scalar or vector (for example, colour images or 2D motion).

$$g(x) = af(x) + b \qquad (2)$$

A linear blend operator is also available, which can cross-dissolve two video frames: a and b are terms that can be used to describe gain and bias parameters, respectively:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x) \qquad (3)$$

It may be thought of as an image morphing method because it ranges from 0 to 1. By taking the average of all intensity values in a video frame, converting it to a middle grey value, and adjusting the range until every pixel in the image is covered, then it can automatically determine the optimal brightness and gain control balance. In addition to seeing how the intensity changes over time using the individual colour channels, we can also find out the minimum, maximum, and average intensity value of a frame using the histogram distribution[17]. As a consequence, histogram equalisation may be used to identify an intensity mapping function that produces a flat histogram.
Neighbourhood filters most commonly use linear filters with a weighted sum of the input pixels as the output pixel. Neighbourhood filters can be used to reduce noise, sharpen fine details, or enhance edges in images.

$$g(i,j) = \sum f(i-k, j-l)h(k,l) \qquad (4)$$

The main purpose of picture sharpening is to enhance or bring out blurred features in a video frame. It is defined as h(k, l) where k represents the weight kernel and l represents the neighbouring data. Using k and l as the difference, f(x) can be computed as a first-order derivative.

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \qquad (5)$$

In a colour picture, each channel consists of one or more colour values. Images containing RGB colours contain three different colour channels (red, blue and green), while those of HSI colour models contain three colours (hue, saturation, and intensity) and those of CMYK colour models contain four colours (cyan, magenta, yellow, and black). In turn, human observers cannot understand the meaning of these colour representations because they were created specifically for devices. Colour transformations are thus required to represent colours from one colour space to an understandable one[18]. Colour transformations are most commonly used for video frame recognition tasks to create a grayscale representation which is used for further video processing. The brightness technique calculates the grayscale value of an RGB colour by multiplying it by

$$\text{Grey} = (\max(R, G, B) + \max(R, G, B))/2 \qquad (6)$$

Averaging RGB values is simply taking the average of the values.

$$\text{Grey} = (R + G + B) / 3 \qquad (7)$$

As a result of all these activities, we received preprocessed photos, which will be used for feature extraction. Some approaches use the weighted average of multiple RGB channels to improve human perception.

3.3 Feature Extraction

A traditional AE is usually composed of fully connected layers and it takes a One - Dimensional (1D) vector as input, which destroys the original spatial structure of the data. This is because the convolution-based operation has high flexibility in processing multi-dimensional data and has a strong ability in feature extraction. A 3D-CAE with convolutional layers instead of fully connected layers is designed in this paper, which makes the input form of the network more variable[19]. HSIs are 3D tensor data containing hundreds of spectral bands, which can provide abundant spectral and spatial information. This design for 3D-CAE consists of three layers of fully 3D convolutional layers and three layers of deconvolutional layers (Figure 6), where Conv-n represents an n^{th} convolutional layer and Deconv-n represents an n^{th} deconvolutional layer, respectively. For each pixel in HSIs, a 3D block centred on the current observed pixel is used as the input of 3D-CAE to learn its invariant characteristics.

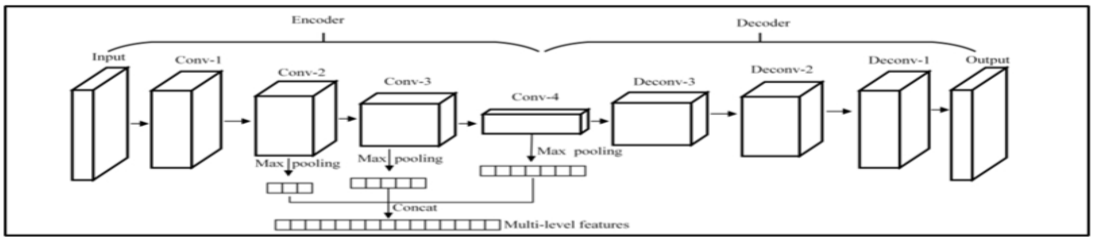


Figure 6. 3D- Convolution Autoencoder for feature extraction

Initially, a 3D-CAE is constructed. The 3D-CAE is designed as a symmetrical structure composed of 3D convolutional layers and deconvolutional layers, as shown in Figure 6. The size of the feature map is gradually reduced, and the number of convolution kernels are gradually increased. The size of the output is same as the size of the input. Secondly, train and optimize the 3D-CAE network. The data is sent as the input into the 3D-CAE and encoded as a low-dimensional representation through the encoder [20]. The decoder is responsible for recovering the original input data from the representation. The 3D-CAE is constantly adjusted by minimizing the error between the output (O x,y,z) and input (I x,y,z), as described in Equation (2). When the network can reconstruct the input data well, it is believed that the network has a strong ability to mine the useful information in the data.

$$Error\ = \frac{1}{I_1 \times I_2 \times I_3} \sum_{x=0}^{I_1-1} \_y = 0\hat{}1/2 - 1I_3 - 1\left(I_{z=0}^{x,y,z} - 0^{x,y,z}\right)^2 \qquad (8)$$

Thirdly, multi-level features from the optimized encoder are obtained. The hierarchical structure of the encoder from the bottom to the top provides us with features of different levels and different scales. The filter size of max-pooling has been set equal to the size of the corresponding feature map to reduce the dimension of the feature and increase its invariance [21]. Through pooling operations, each layer can get a feature vector containing different information. The final features are concatenated by these feature vectors from multiple layers of the encoder to make them contain more information and have high scale robustness. It is worth noting the proposed multi-level features from a single network. When compared with training multiple networks to obtain multi-level features, the proposed method is more effective and saves training time. We expect to make full use of the well-trained network to obtain as much information as possible and then help to improve the subsequent classification accuracy.

3.4 Feature Selection

Feature mappings U ∈ R H×W×C are mapped from an input X ∈ R H0×W0×C0 to a Squeeze-and-Excite block based on the transformation Ftr. Ftr is a convolutional operator in the following notation, where V = [v1, v2,..., vC] denotes the learnt set of filter kernels, with vC referring to the c-th filter's parameters. The outputs may therefore be written as U = [u1, u2,..., uC], where

$$u_c = v_c * X = \sum_{s=1}^{C'} \ ?\, v\_c\hat{}s * x^s. \qquad (9)$$

Here ∗ denotes convolution, v_c = [v1c, v2c, . . . , vC0c], X =[x1, x2, . . . , xC0] and u_c ∈ RH×W . The v^s_c is a two-dimensional spatial kernel that represents a single VC channel and its corresponding X channel. Bias terms are deleted to simplify the notation. As v_c implicitly includes channel dependences, they are entangled with the local spatial correlation collected by the filters, since the output is determined by averaging all channels. In convolutional models, interactions occur mainly implicitly and locally (except for at the top). We believe that explicitly modelling channel interdependencies will improve convolutional feature learning, allowing the network to raise its sensitivity to useful characteristics that may be exploited by future modifications. As a result, we'd want to provide it universal access and tune filter responses in two stages, squeezing and excitement, before feeding them into the next transformation [22]. Figure 7 is a schematic depicting the construction of an SE block.

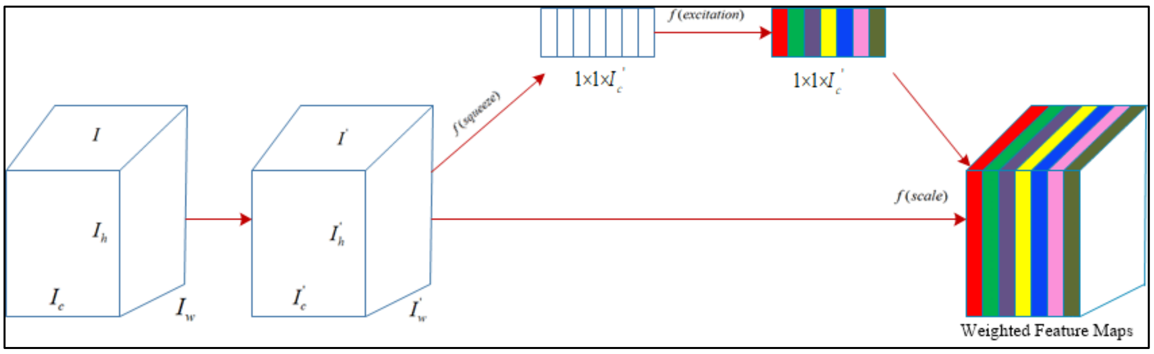


Figure 7. SE-block for feature selection

Embedding information: Analyzing the output features reveals that channel dependence does not exist. Since each of the learned filters has a limited reception field, the transformation output U does not incorporate contextual information from outside its receptive field. To solve this problem, we propose to construct channel descriptions by compressing global spatial information. We accomplish this by creating channel-specific data by pooling global averages. By decreasing U through its spatial dimensions, we calculate the c-th element of z, so the stat z R C becomes:

$$z_c = F_{sq}(u_c) = \frac{1}{H\times W}\sum_{i=1}^{H} \_j = 1^\wedge W\, u_c(i,j). \quad (10)$$

Adaptive Re-calibration: After the squeeze operation, we perform another operation that prioritizes the collection of channel-wise dependencies to take advantage of the information gathered. The system must meet three requirements to accomplish this goal: First, it must be flexible (i.e. able to learn nonlinear relationships between channels); Secondly, it must learn relationships between multiple channels that are not mutually exclusive[23] and Thirdly, it must be fast. We use a simple gating system with sigmoid activation to achieve these requirements.

$$s = F_{ex}(z,W) = \sigma(g(z,W)) = \sigma\big(W_2\delta(W_1 z)\big) \quad (11)$$

where δ represents the function ReLU [63], W1 ∈ RCr ×C and W2 ∈ R C× Cr . Using dimensionality-reduction layers with reduction ratios r, and a ReLU layer, we form a bottleneck around the non-linearities and in the next layer, dimensionality will increase, returning to the channel dimensions of the transformation output U, to reduce model complexity and facilitate generalization. By scaling U with the activations s, the final output of the block can be obtained as:

$$x_c = F_{scale}\,(u_c, s_c) = s_c u_c \quad (12)$$

The excitation operator converts a collection of channel weights into the input-specific descriptor z. This aspect of SE blocks is that they have an input-dependent dynamical response, which is similar to a self-attentional function on channels that aren't limited by the local receptive field to which the convolutional filters respond.

By inserting the SE block after the non-linearity following each convolution, it may be incorporated into conventional architectures like Faster-RCNN and Yolo. Furthermore, because of the SE block's versatility, it may be used for transformations other than normal convolutions directly. To demonstrate this idea, we create SENets by combining SE blocks into a variety of more complicated structures which will be discussed next. The building of SE blocks for Inception networks is first considered [24]. We may obtain a SE-Faster-RCNN-Yolo network by simply changing the transformation Ftr to be a whole Inception module and repeating this for each such module in the design. The non-identity branch of the relevant module is assumed here to represent the SE block transformation Ftr. Before summation with the identity branch, Squeeze and Excitation both act. Table 2 shows a comparison of the feature selection approach with other strategies over two datasets.

Table 2. An overall analysis of feature selection methods

Methods	**Accuracy**	**Computation time**	**MSE**	**Dataset**
ASSO	88	6.7	67	ImageNet VID
PSO	89	5.4	79	
SSO	84	7.9	82	
LASSO Regression	90	5.1	66	
PCA	94	4.3	77	
SE-Block (Ours)	94	4.1	87	
ASSO	90	4.1	83	CIFAR-10
PSO	88	6.2	78	
SSO	84	6.9	73	
LASSO Regression	85	6.1	77	
PCA	93	4.3	89	
SE-BLOCK (ours)	95	4.2	92	

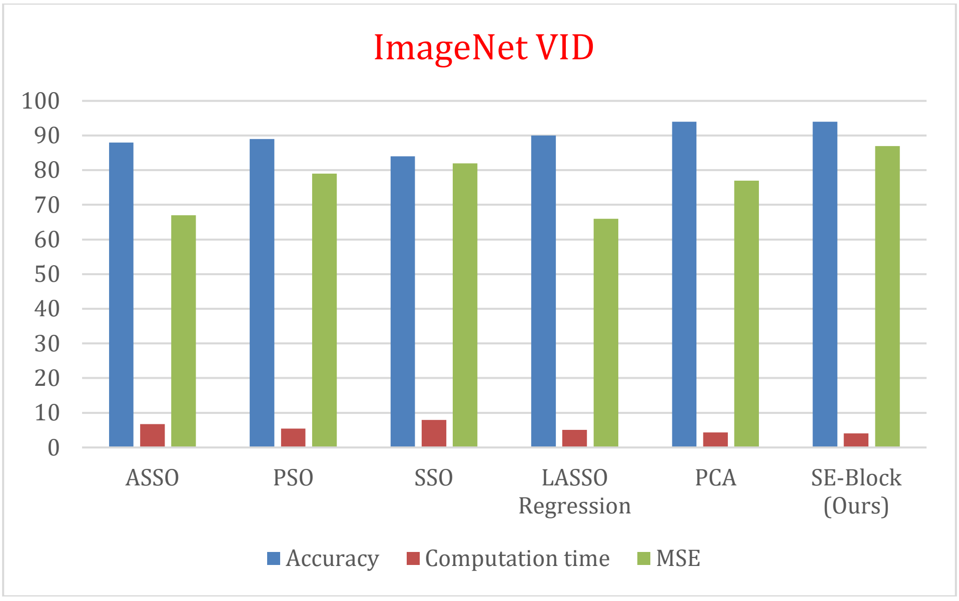


Figure 8(a). Method vs Accuracy, Computation time, MSE over ImageNet VID

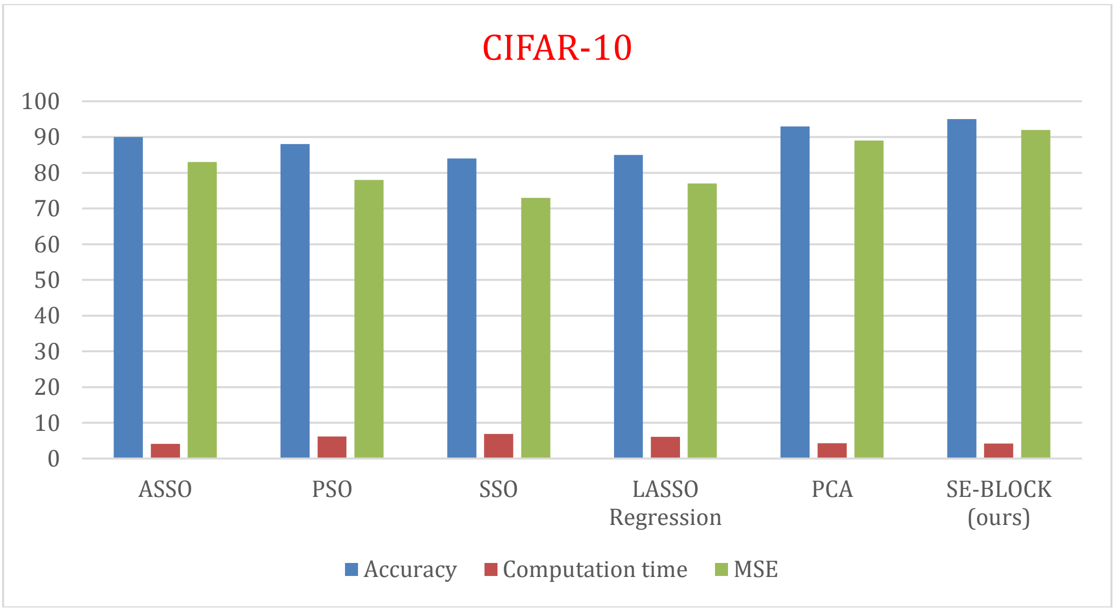


Figure 8(b). Method vs Accuracy, Computation time, MSE over CIFAR

3.5 Object Detection

Once all of the features have been chosen, the data will be sent to a classification/detection stage utilizing two networks: Faster-RCNN and Yolo. The framework for our model is the Faster-RCNN network, which makes use of two modules: a deep convolutional network in the first module, which identifies regions, and a Fast R-CNN detector in the second module, which uses the suggested regions. A unified network is used for object detection (Fig. 9). Using RPN processes, the Fast R-CNN module tells the RPN module where to look.

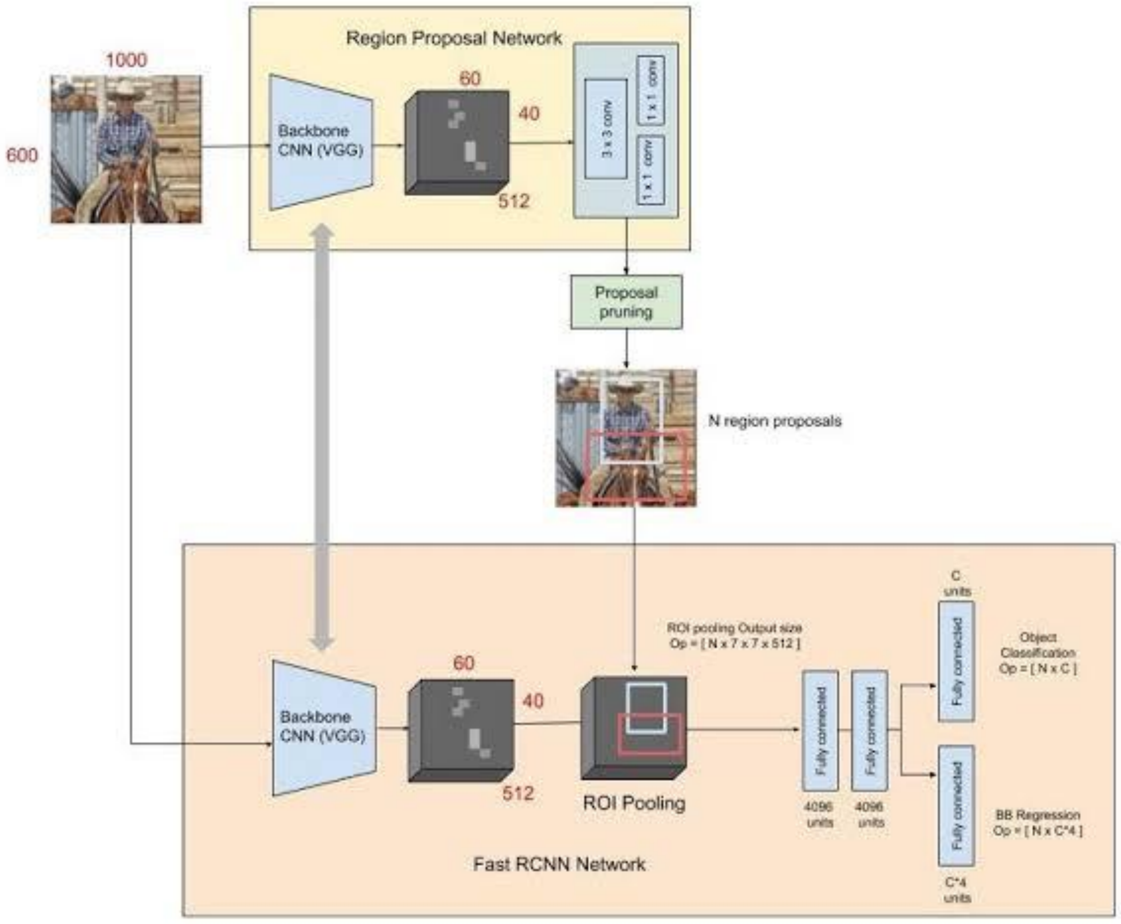


Figure 9. Faster-RCNN and RPN for object detection

Using a fully convolutional net [7], we simulate this process by generating a sequence of three rectangle-shaped ideas based on an image (of any size). This section describes our fully convolutional net. We assume that both Fast R-CNN, as well as Fast R-CNN, have the same set of convolutional layers because our ultimate goal in sharing computation with them is to recognize objects [2].

This tiny network slides over the convolutional feature map that is generated by the final shared convolutional layer. The input convolutional feature map is fed into an n spatial window of this network. Assigned to each sliding window are lower-dimensional features (256-dimensional ZF, 512-dimensional VGG, followed by ReLU [25]). The two sisters' layers that feed into the feature are the box-regression layer and the box-classification layer. This study was conducted with n = 3 because the input image has a large effective receptive field (171 and 228 pixels). The result of this analysis is shown in Figure 10 at a single location. Due to the sliding-window structure of the mini-network, the fully linked layers are accessible at multiple locations. During this design (for reg and cls, respectively), two sibling convolutional layers are followed by an NN layer.

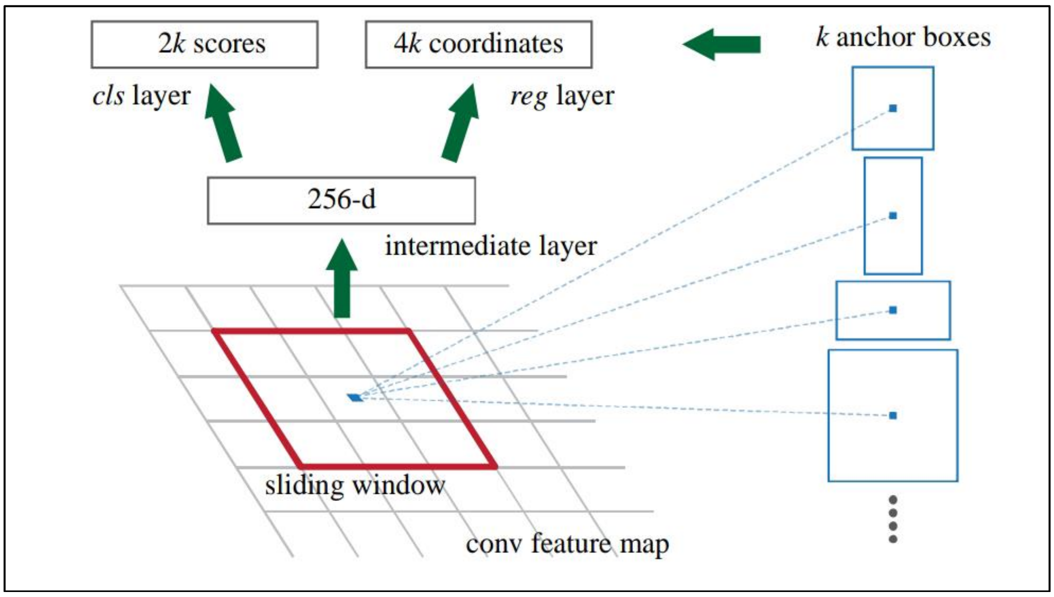


Figure 10. Regions proposal Network

We anticipate receiving numerous area options at each sliding-window location at the same time, with the request specifying the maximum number of possible proposals per location. As a result, a reg layer generates 4k outputs encoding the positions of each box, whereas a cls layer generates 2k scores estimating the possibility of each proposal being an object. Based on k anchors, which are k reference boxes, a cl layer generates a score for each proposal. The anchor displayed at each sliding point of the window, as well as the size and aspect ratio, are shown in figure 10. By default, we use three scales and three aspect ratios, resulting in nine anchors. There are W Hk anchors in total for a convolutional feature map of size W × H (usually ~2,400).

Translating invariance is a critical component of our strategy both for anchors and for methods that calculate suggestions relative to the anchors. Whenever an object is translated in an image, the proposal should also be translated, and forecasts in both locations should be done by the same function. Our technique [4,5] ensures this translation-invariant characteristic. This will operate as a guardian to prevent overfitting on the dataset.

Once the appropriate results obtained utilizing the anchor of faster-RCNN, the data was sent to Yolo V3 for even better detection and classification. As a result, we employed a layered YOLOv3 architecture with layer stacking to achieve this. The model, which was designed using the Darknet-53 architecture and trained using the ImageNet dataset, is capable of detecting even the tiniest details in a photograph. In a single picture, the suggested model can distinguish 80 distinct things. A total of 53 more layers are piled on top of it for detection, giving us a total of 106 completely convolutional layers. [19, 20] Figure 11 depicts the planned architecture.

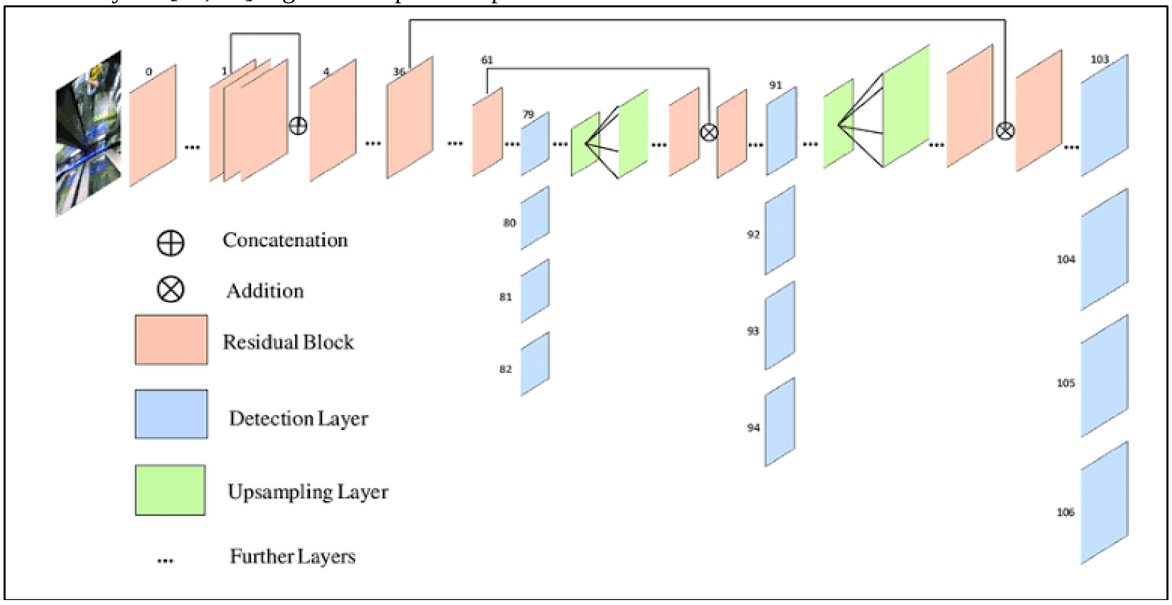


Figure 11. Yolov3 block diagram for object detection

The most notable characteristic is that it detects at three distinct scales. This network uses three separate S x S kernels to perform the identification. Its formula is S x S x (B x 5 + C). In each feature-map cell, there are four bounding boxes.

The fourth is full of attributes, and the fifth represents object confidence. The C is used to represent how many classes are present in the model.

As a result, the kernel has a size of 1 x 1 x 255. The 82nd layer develops the initial detecting procedure. For 81 layers, the picture is down-sampled by 32, yielding a feature map of 13 x 13 for a 416 x 416 picture, and an image with 354 x 354 pixels can be detected with a 13 x 13 x 225 feature map generated by the 1x1 detection kernels[26]. This second detection is constructed based on the 94th layer, creating a 26 x 26 x 225 detection, resulting from the 2x2 upsampling of layer 79. This second detection is then blended with the depth map created from layer 61. A feature map with dimensions of 52 x 52 x 225 pixels is created as a final step in the 106th layer, with an activation function of 1/(1 + e-x). At different levels, Soft-max, Re-Lu, Tan hyperbolic, and other activation functions are utilized to generate scores.

$$\text{Sigmoid: } 1/(1+e^{-x}) \qquad (13)$$
$$\text{Softmax: } e^{x}/\left(sum\ (e^{x})\right)$$
$$\text{Re-Lu: } y = max(0, \infty)$$
$$\text{Tanh:}[2f(1+e^{-2x})]-1$$

It transforms an image into a S x S grid structure. Each grid identifies an item in the image. At the moment, the grid cell predicts the bounding box size for each item. Each bounding box has five items (x, y, w, h, confidence). The confidence score represents the likelihood that a bounding box will be associated with an item, as well as the accuracy of the bounding box[27]. The object's, 'x' and 'y' coordinates are represented by the "x" and "y" axes in the input picture, and the object's width and height are indicated by the "w" and "h" axes, respectively.
Confidence =Probability (object) * IoU

When YOLOv3 is implemented, a cross-entropy loss function replaces the mean squared error (used in YOLOv2). This loss function is:

$$\sum_{c=1}^{\cdots} \int_{x \in c} log\ (P(x \in c)) \qquad (14)$$

After YOLOv3-enhanced images are detected, bounding boxes are calculated via log(p(x€C), which describes the probability that the identified object is a member of class C. Due to the large number of bounding boxes that have emerged for one item, post-processing steps are required. NMS (Non-Maximum Suppression) can be used to solve this problem. The proper bounding box is returned after NMS eliminates all overlapping bounding boxes[28]. Bounding boxes are predicted similarly to Yolov2. As a result, for each bounding box, the network predicts four coordinates: tw, th, tx, ty. Figure 12 shows an illustration of the bounding box.

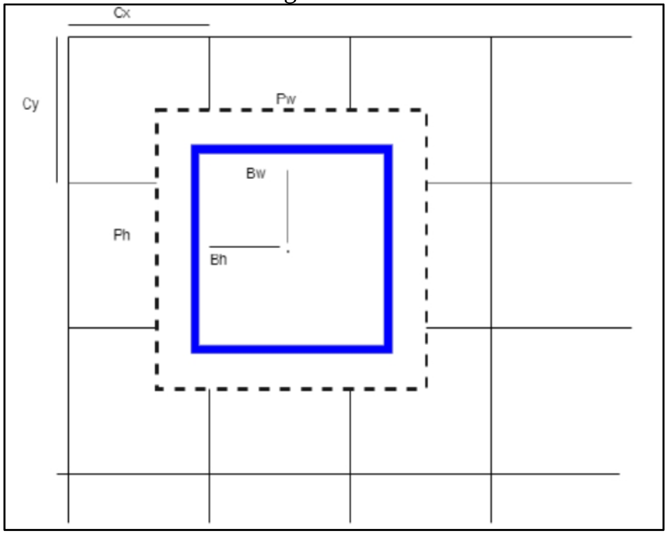


Figure 12. Yolov3 bounding box: prior (black dots), predicted (blue), predicted bounding box

Following formula for calculating bounding box coordinates [9]:

$$b_x = \sigma(t_x) + c_x\ b_y = \sigma(t_y) + c_y\ b_w = p_w e^{t_w}\ b_h = p_h e^{t_h} \qquad (15)$$

The Mean Average Precision (MAP) [29] is calculated using the IoU calculation. The IoU approach is useful for figuring out whether a projected box represents a true positive, a false positive, or a false negative since it implies that anything could be included in the bounding box. It is widely accepted that the threshold for IoU is 0.5. If the IoU value is more than 0.5, we may conclude that the result is genuine positive. False positives are those that have IoUs that is less than 0.5. False negatives are those that have IoUs greater than 0.5 and items are misclassified. YOLO utilizes non-maximal suppression to reject duplicate items. There are no exceptions to non-maximal suppression if there is a threshold for IoU between any prediction in the image.

4. Performance Analysis

The proposed system (Faster-RCNN+yolov3) is developed using software specifications like PyTorch as a programming language which is an open-source library inside python for deep learning purposes. Hardware specifications used for building up this model is GTX 1050 Ti 4GB Graphics (Core i7-8750H 8th Gen/8GB RAM/1TB SSHD + 128GB SSD and Windows 10 OS. The proposed model is evaluated using measures like accuracy, sensitivity, specificity, recall, precision, F1-score, detection rate, TPR, FPR, IoU, mAP, computation time and more Utilization over models like RCNN, VGG16, Alexnet, Googlenet, CNN, Regular Faster-RCNN, Regular Yolov3.

Table 3 depicts the overall analysis of the proposed framework with other models over measures like Accuracy, specificity and sensitivity. Figure 13(a,b) depict the graphical representation of various models over the proposed method in which the proposed method integrated network outperforms the other models due to double-layer network structure and tendency to accurately detect those.

Table 3. An overall analysis of models over accuracy, sensitivity and specificity

Models	Dataset	Accuracy	Sensitivity	Specificity
RCNN	ImageNet VID	81	88	85
VGG16		83	87	86
Alexnet		80	85	84
Googlenet		86	91	90
CNN		87	92	91
Regular Faster-RCNN		91	97	93
Regular Yolov3		94	96	95
Faster-RCNN + Yolov3 (ours)		96	98	97
RCNN	CIFAR-10	83	87	85
VGG16		86	90	89
Alexnet		81	86	87
Googlenet		88	92	93
CNN		90	95	92
Regular Faster-CNN		94	96	97
Regular Yolov3		96	98	97
Faster-RCNN + Yolov3 (Ours)		97	99	98

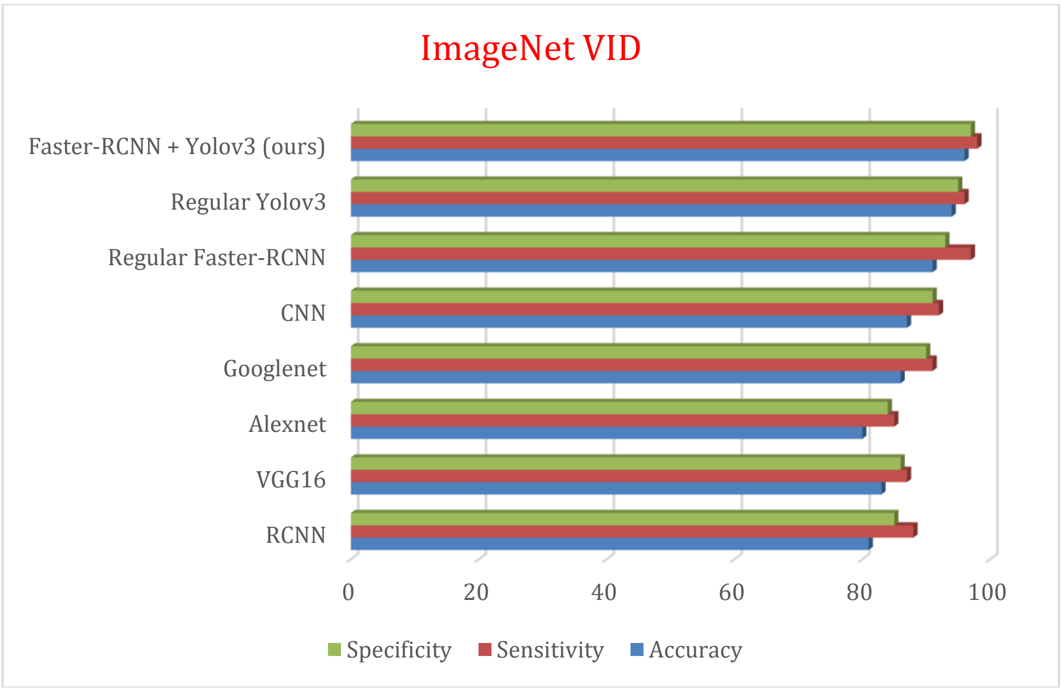


Figure 13. (a) Models vs Specificity, Sensitivity and accuracy over ImageNet VID

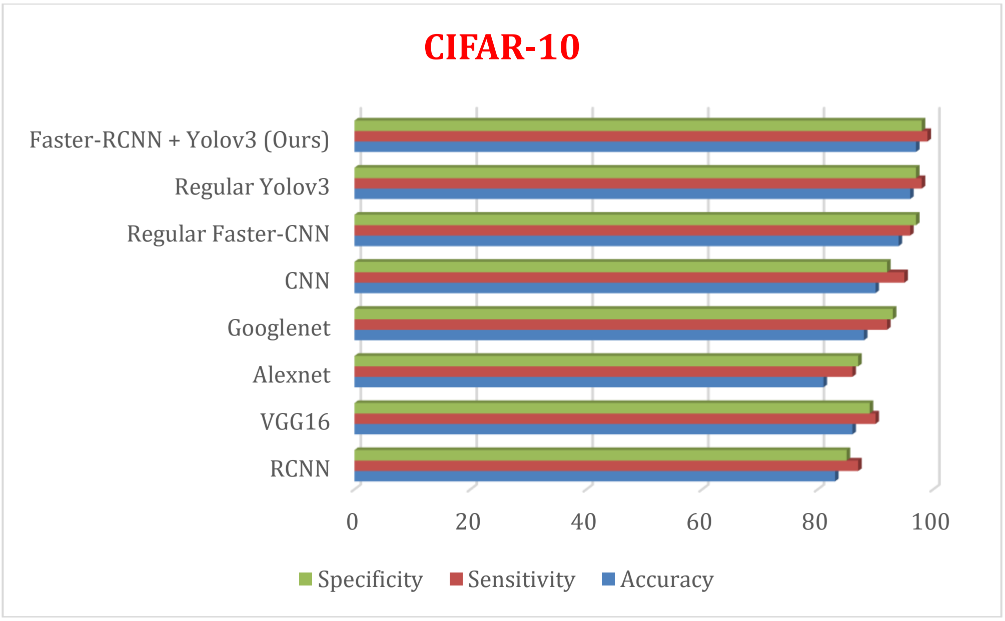
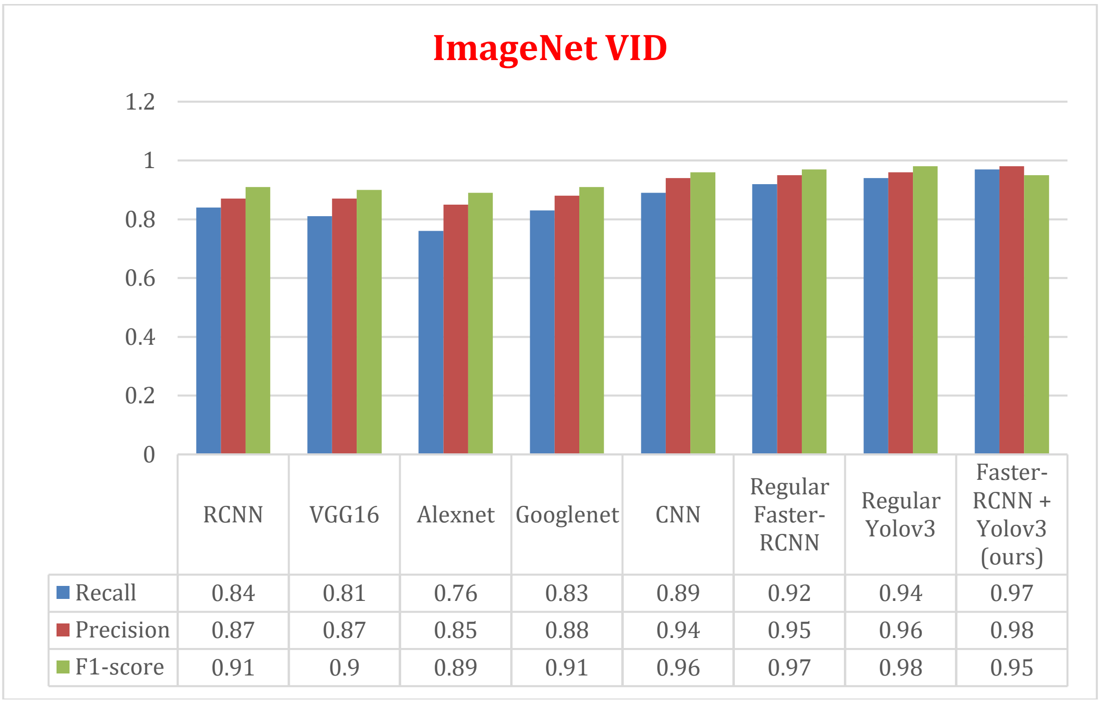


Figure 13. (b) Models vs Specificity, Sensitivity and accuracy over CIFAR-10

Table 4 depict the overall analysis of various models over the proposed method under measures like precision, F1-score and recall. Figure 14(a,b) depict a graphical representation of various models over the proposed method in which the proposed integrated method really outperforms better with the high-level result due to the effect of SE-block as the booster feature selection and thereby increases the efficiency.

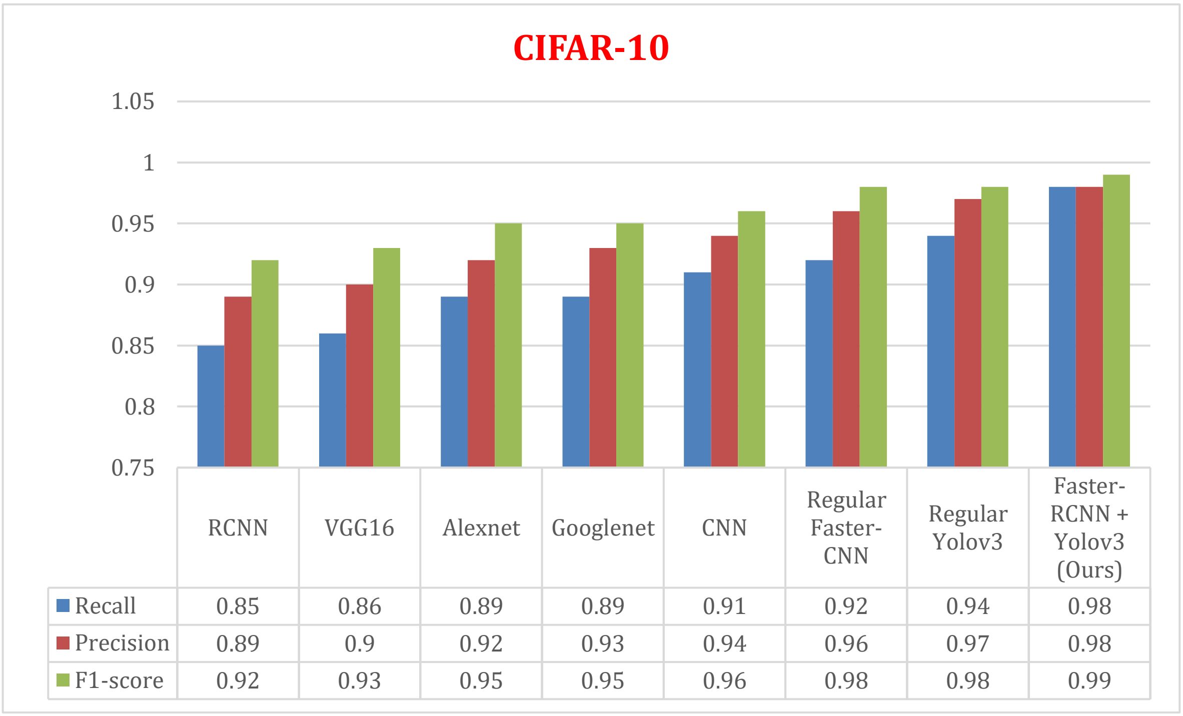
Table 4. An overall analysis of various models over recall, precision and F1-score

Models	**Dataset**	**Recall**	**Precision**	**F1-score**
RCNN	ImageNet VID	0.84	0.87	0.91
VGG16		0.81	0.87	0.90
Alexnet		0.76	0.85	0.89
Googlenet		0.83	0.88	0.91
CNN		0.89	0.94	0.96
Regular Faster-RCNN		0.92	0.95	0.97
Regular Yolov3		0.94	0.96	0.98
Faster-RCNN + Yolov3 (ours)		0.97	0.98	0.95
RCNN	CIFAR-10	0.85	0.89	0.92
VGG16		0.86	0.90	0.93
Alexnet		0.89	0.92	0.95
Googlenet		0.89	0.93	0.95
CNN		0.91	0.94	0.96
Regular Faster-CNN		0.92	0.96	0.98
Regular Yolov3		0.94	0.97	0.98
Faster-RCNN + Yolov3 (Ours)		0.98	0.98	0.99



	RCNN	VGG16	Alexnet	Googlenet	CNN	Regular Faster-RCNN	Regular Yolov3	Faster-RCNN + Yolov3 (ours)
🟦 Recall	0.84	0.81	0.76	0.83	0.89	0.92	0.94	0.97
🟥 Precision	0.87	0.87	0.85	0.88	0.94	0.95	0.96	0.98
🟩 F1-score	0.91	0.9	0.89	0.91	0.96	0.97	0.98	0.95

Figure 14. (a) Models vs Recall, Precision and F1-score over ImageNet VID



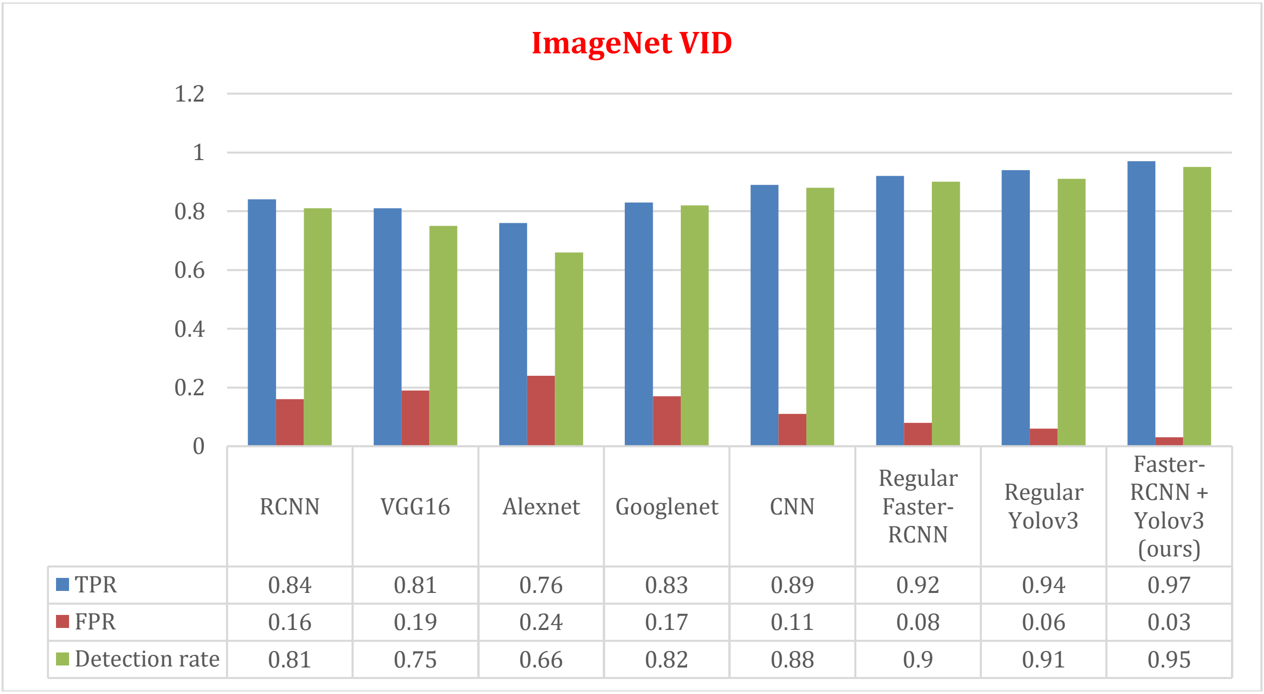
	RCNN	VGG16	Alexnet	Googlenet	CNN	Regular Faster-CNN	Regular Yolov3	Faster-RCNN + Yolov3 (Ours)
🟦 Recall	0.85	0.86	0.89	0.89	0.91	0.92	0.94	0.98
🟥 Precision	0.89	0.9	0.92	0.93	0.94	0.96	0.97	0.98
🟩 F1-score	0.92	0.93	0.95	0.95	0.96	0.98	0.98	0.99

Figure 14. (b) Models vs Recall, Precision and F1-score over CIFAR-10

Table 5 shows the overall analysis of various models over the proposed method under measures like TPR, FPR and detection rate. Figure 15(a,b) depict the graphical representation of various models over the proposed method.

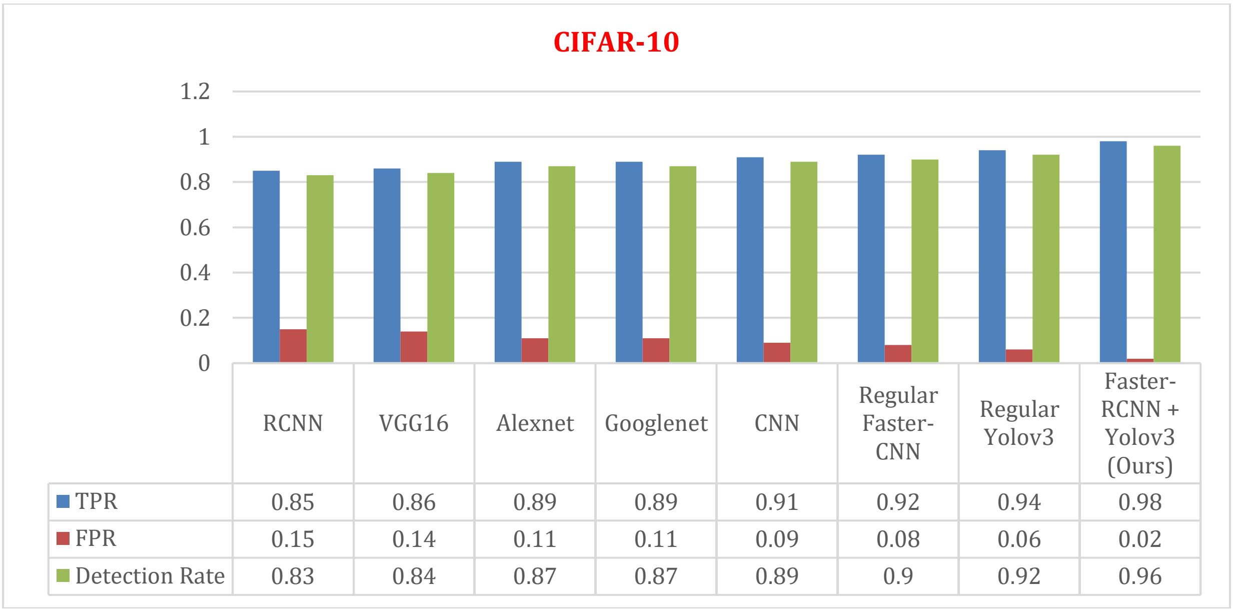
Table 5. An overall analysis of various models over TPR, FPR and detection rate

Models	**Dataset**	**TPR**	**FPR**	**Detection rate**
RCNN	ImageNet VID	0.84	0.16	0.81
VGG16		0.81	0.19	0.75
Alexnet		0.76	0.24	0.66
Googlenet		0.83	0.17	0.82
CNN		0.89	0.11	0.88
Regular Faster-RCNN		0.92	0.08	0.90
Regular Yolov3		0.94	0.06	0.91
Faster-RCNN + Yolov3 (ours)		0.97	0.03	0.95
RCNN	CIFAR-10	0.85	0.15	0.83
VGG16		0.86	0.14	0.84
Alexnet		0.89	0.11	0.87
Googlenet		0.89	0.11	0.87
CNN		0.91	0.09	0.89
Regular Faster-CNN		0.92	0.08	0.90
Regular Yolov3		0.94	0.06	0.92
Faster-RCNN + Yolov3 (Ours)		0.98	0.02	0.96



	RCNN	VGG16	Alexnet	Googlenet	CNN	Regular Faster-RCNN	Regular Yolov3	Faster-RCNN + Yolov3 (ours)
🟦 TPR	0.84	0.81	0.76	0.83	0.89	0.92	0.94	0.97
🟥 FPR	0.16	0.19	0.24	0.17	0.11	0.08	0.06	0.03
🟩 Detection rate	0.81	0.75	0.66	0.82	0.88	0.9	0.91	0.95

Figure 15. (a) Models vs TPR, FPR and Detection Rate over ImageNet VID



	RCNN	VGG16	Alexnet	Googlenet	CNN	Regular Faster-CNN	Regular Yolov3	Faster-RCNN + Yolov3 (Ours)
🟦 TPR	0.85	0.86	0.89	0.89	0.91	0.92	0.94	0.98
🟥 FPR	0.15	0.14	0.11	0.11	0.09	0.08	0.06	0.02
🟩 Detection Rate	0.83	0.84	0.87	0.87	0.89	0.9	0.92	0.96

Figure 15. (b) Models vs TPR, FPR and Detection Rate over CIFAR-10

Table 6 shows the overall analysis of various models over the proposed method under measures like IoU, mAP. Figure 16(a,b) depict a graphical representation of various models over the proposed method.

Table 6. An overall analysis of models over IOU, mAP

Models	**Dataset**	**IOU**	**mAP**
RCNN	ImageNet VID	0.4	0.85
VGG16		0.3	0.76
Alexnet		0.4	0.87
Googlenet		0.5	0.92
CNN		0.5	0.90
Regular Faster-RCNN		0.5	0.93
Regular Yolov3		0.5	0.96
Faster-RCNN + Yolov3 (ours)		0.5	0.98
RCNN	CIFAR-10	0.5	0.89
VGG16		0.3	0.79
Alexnet		0.3	0.76
Googlenet		0.5	0.91
CNN		0.4	0.86
Regular Faster-CNN		0.5	0.93
Regular Yolov3		0.5	0.97
Faster-RCNN + Yolov3 (Ours)		0.5	0.98

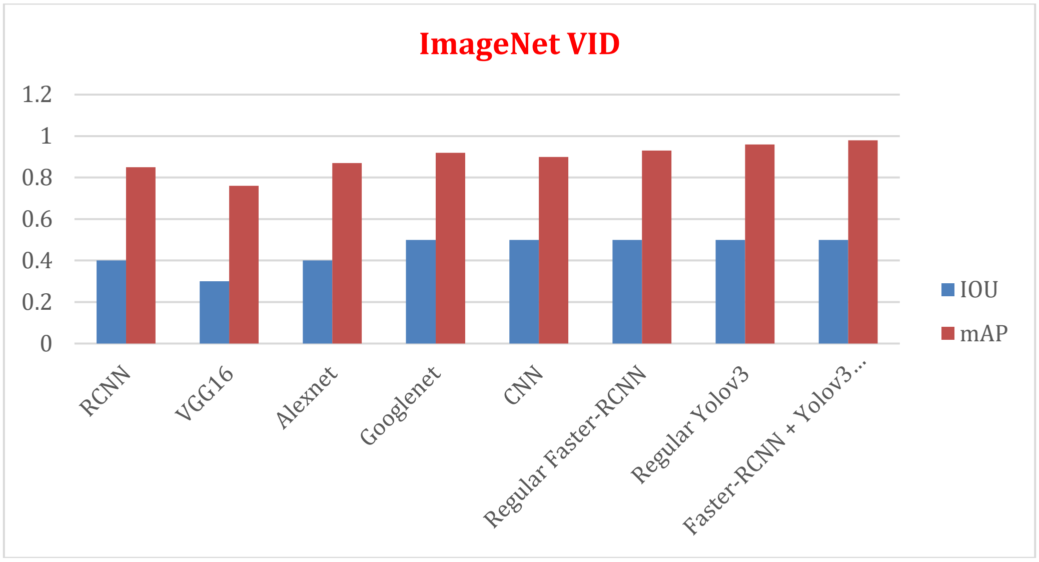


Figure 16. (a) Models vs IOU and mAP over ImageNet VID

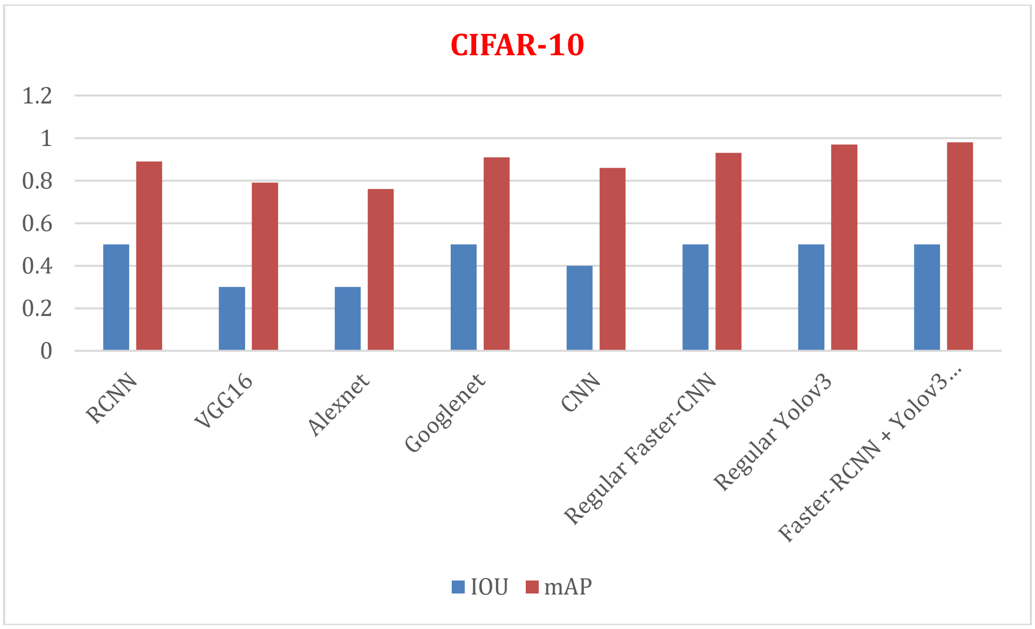


Figure 16. (b) Models vs IOU and mAP over CIFAR-10

Figure 17 (a,b) depict a graphical representation of various models over the proposed method under measures like computation time and memory utilisation. The proposed method outperforms with less memory utilisation and comparatively less computation time.

5. Conclusion

Distinguishing objects in a web based video is intricate because of the profundity of data accessible in each edge. Usually a video is cut into image outlines and then the image of interest is recognized for process. Later many steps are followed like pictures are separated into training and testing set for test etc. This study show that the framework used in this paper effectively handled pictures and hence brings an effective object detection model from video frames integration of Yolo-Faster-RCNN. The study shows that the proposed method outperforms with 95% accuracy when compared with other state-of-art models. The proposed Yolo-FRCNN calculation is utilized and the outcomes are contrasted and different condition state-of-art models under different measures. The outcomes are extremely encouraging and much effective. Additionally, it would be valuable for other exploration expert to burrow profound and get to know all perspectives and ready to carry out even significant level models and accordingly brings much more viable outcomes also.

REFERENCES

[1].Dr YasirZafar Khan, “A Dream to be true”, International Journal of Linguistics and Computational Applications (IJLCA), ISSN 2394-6385, 2015.

[2].Maggipinto, M., Masiero, C., Beghi, A., & Susto, G. A. (2018). A convolutional autoencoder approach for feature extraction in virtual metrology. Procedia Manufacturing, 17, 126-133.

[3].Naikal, Nikhil, Allen Y. Yang, and S. Shankar Sastry. "Informative feature selection for object recognition via sparse PCA." In 2011 International Conference on Computer Vision, pp. 818-825. IEEE, 2011.

[4].Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7132-7141. 2018.

[5].Sun, Qiaoqiao, Xuefeng Liu, and Salah Bourennane. "Unsupervised Multi-Level Feature Extraction for Improvement of Hyperspectral Classification." Remote Sensing 13.8 (2021): 1602.

[6].Ren, Shaoqing, et al. "Faster R-CNN: towards real-time object detection with region proposal networks." IEEE transactions on pattern analysis and machine intelligence 39.6 (2016): 1137-1149.

[7].Oliveira, Inês, Nuno Correia, and Nuno Guimarães. "Image processing techniques for video content extraction." In Proceedings of 4th Dellos Workshop. 1997.

[8].Shahriar, Md Tanzil, and Huyue Li. "A Study of Image Pre-processing for Faster Object Recognition." arXiv preprint arXiv:2011.06928 (2020).

[9].Raj, Jeberson Retna, and Senduru Srinivasulu. "Object Detection in Live Streaming Video Using Deep Learning Approach." In IOP Conference Series: Materials Science and Engineering, vol. 1020, no. 1, p. 012028. IOP Publishing, 2021.

[10]. Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115, no. 3 (2015): 211-252.

[11]. Kalaivani, R., and C. Manicha. "Object Detection in Video Frames Using Various Approaches." International Journal of Advanced Research in Computer and Communication Engineering 2, no. 9 (2013): 157-160.

[12]. Patro, Shrikant Jagannath, and V. M. Nisha. "Real-Time Video Analytics for Object Detection and Face Identification using Deep Learning."

[13]. Ravish Aradhya, H. V. "Object detection and tracking using deep learning and artificial intelligence for video surveillance applications." International Journal of Advanced Computer Science and Applications 10, no. 12 (2019): 517-530.

[14]. Ma, Yongjun, and Songhua Zhang. "Feature Selection Module for CNN Based Object Detector." IEEE Access 9 (2021): 69456-69466.

[15]. Ploeger, Spencer, and Lucas Dasovic. "Issues in Object Detection in Videos using Common Single-Image CNNs." arXiv preprint arXiv:2105.12822 (2021).

[16]. Pal, Sankar K., Anima Pramanik, Jhareswar Maiti, and Pabitra Mitra. "Deep learning in multi-object detection and tracking: state of the art." Applied Intelligence 51, no. 9 (2021): 6400-6429.

[17]. Zhao, Zhong-Qiu, Peng Zheng, Shou-tao Xu, and Xindong Wu. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30, no. 11 (2019): 3212-3232.

[18]. T. Lu et al., Video Text Detection, Advances in Computer Vision and Pattern Recognition, Springer-Verlag London 2014.

[19]. Chandan, G., Ayush Jain, and Harsh Jain. "Real-time object detection and tracking using Deep Learning and OpenCV." *2018* international conference on inventive research in computing applications (ICIRCA). IEEE, 2018

[20]. Lu, Shengyu, et al. "A real-time object detection algorithm for video." Computers & Electrical Engineering 77 (2019): 398-408.

[21]. Göring, Christoph, Erik Rodner, Alexander Freytag, and Joachim Denzler. 2014. “Nonparametric Part Transfer for Fine-Grained

[22]. Girshick, Ross, Forrest Iandola, Trevor Darrell, Tian, Yonglong, Ping Luo, Xiaogang Wang, and Xiaoou Tang. (2015). “Deep Learning Strong Parts for Pedestrian Detection.” In Proceedings of the IEEE International Conference on Computer Vision, 1904–12

[23]. Zhang, Ning, Jeff Donahue, Ross Girshick, and Trevor Darrell. (2014). “Part-Based R-CNNs for Fine-Grained Category Detection.” In European Conference on Computer Vision, 834–49

[24]. Redmon, J, and A Farhadi. (2017). “YOLO9000: Better, Faster, Stronger.” In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6517–25.

[25]. Shih, Ya-Fang et al. (2017). “Deep Co-Occurrence Feature Learning for Visual Object Recognition.” In Proc. Conf. Computer Vision and Pattern Recognition

[26]. Faisal I.Basshir et.al “Real-time motion trajectory-based indexing and retrieval of video sequence” IEEE, pp: 1-8, January 7, 2005

[27]. Sanjirani shantaiya et.al, “ A survey on approaches of object detection”, International journal of computer applications, Vol-65, No-18, March 2013

[28]. Lakshmi Rupa G, Gitanjali, “A video mining application for image retrieval”, International journal of computer applications, Vol-20, No-3, pp46-52, April 2011.

[29]. Oh, S. Kang, H,” Object detection and classification by decision-level fusion for intelligent vehicle systems”. *Sensors*,2017 A. Sanin, C. Sanderson, B. C. Lovell, "Shadow detection: A survey and comparative evaluation of recent methods", *Pattern Recognit.*, vol. 45, no. 4, pp. 1684-1695, 2012.