Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

3d Multiobject Detection And Tracking Using The Deep Learning Based Multilayer Stacked Yolo V8

D. Raj , Dr. R. Murugeswari

[1]Research Scholar , [2]Associate Professor
[1,2]Kalasalingam Academic Research
Education, Kalasalingam University.

The widespread availability and growing number of uses for surveillance cameras has prompted an increase of study into the best ways to identify and keep track of moving targets in real time. However, various difficulties have evolved that impede the detection and tracking processes used by monitoring systems. Once the starting location of the target object is specified, we provide a deep learning-based Multilayer stacked YoLoV8 algorithm that predicts the position of the object in each successive frame of a video by considering both position and appearance changes. Here Initially, the data was retrieved and the data were preprocessed using the Mean kalman filter (MKF). Then the data can be augmented further the features are extracted using the Discriminant crossover component analysis(DCCA). Then the specialized features are selected using the Sparse adam optimization algorithm(SAO). Finally the Multilayer stacked YoLoV8 was used to identify the exact object . And for tracking deep sequence angle sort algorithm was used.The recommended technique, when combined with the optimization mechanisms considerably enhances tracking performance over realtime customized video dataset. The results on the customized dataset show that the suggested technique outperforms the current state of the art methods by obtaining good detection and classification accuracy.

Index Terms—3D scenes, Mean kalman filter filter, Discriminant crossover component analysis, Sparse adam optimization algorithm, deep sequence angle sort, Multilayer stacked YoLoV8.

I. INTRODUCTION

Because of the growing number of surveillance cameras deployed globally, particularly in vital cities, they will need regular monitoring by the operator. Adding more people to run the show and more cameras will drive up the price. Massive computers and high-quality servers are required to control these cameras, as well as analyze and store these video footage. As a result, an autonomous, intelligent detection and tracking system had to be developed. Furthermore, there are a number of obstacles unique to real-time video capture, such as the image's low resolution, the presence of various noises, the appearance of moving shadows, the objects' varying sizes and speeds, and the algorithm's ability to distinguish between them. It may result in a high computational cost and demands efficient hardware, tools, and huge computational memory, which might be difficult to deliver. There has been a proliferation of studies and papers that take various routes toward developing a real-time, efficient system.

Many individuals in control centers are finding it challenging to maintain the same level of productivity throughout the day as a result of the expansion of monitoring systems and the growth in CCTV cameras. Therefore, there was a need for efficient real-time detection and tracking systems. It has a wide range of potential uses, including the early identification of alien species and violent crime.The following are the two steps of a newly suggested algorithm for object detection and tracking, and they constitute its major contribution.

• The detecting phase includes two mandatory processes and is the initial stage. First, the video frame is captured, noise is removed, features are extracted using Discriminant crossover component analysis, then the specialized features are selected using the Sparse adam optimization algorithm, and finally, moving objects within the video frame are detected by isolating the object (objects detected) from the background (background scene). To improve motion detection, a YoLoV8 architecture with many layers has been used.

• The second step involves keeping tabs on the objects that have been seen and assigning unique identifiers to each one; this is done by keeping a running tally of each object's appearance, position, and velocity in each frame and connecting that data from one to the next. It also addresses some of the difficulties that arise along the process, as when an object's speed changes from frame to frame or when the number of switches used to alter the object ID being tracked in a given video scene needs to be minimized.

• The dataset is comprised of both synthetic and real-world video recordings made using a camcorder, and the algorithm is applied to both.

The document was formatted and structured into various sections: A basic introduction to detection and tracking systems and the suggested method were presented in the first part. Similar or comparable works will be in the second section. The issue statement for multiple object detection will be in the third part. The fourth part will then go into more depth about the suggested algorithm. The suggested algorithm's testing and experiment results, as well as a comparison of those findings with those of the current algorithm to assess its performance, efficacy, and efficiency, are covered in the fifth part. In the last part, we will show our findings and explain the suggested method.

II. RELATED WORKS

Several existing methodologies are there for the purpose of object detection and tracking, some of themcan be depicted below, In [1], deep learning is used to solve the problems of vehicle identification and fine-grained categorization. THS-10 is a locally produced dataset with strong intraclass and low interclass variance, making it ideal for performing fine-grained classification and its attendant difficulties. There are a total of 4,250 photos in the collection, including 10 different makes and models of cars: the Honda City, Honda Civic, Suzuki Alto, Suzuki Bolan, Suzuki Cultus, Suzuki Mehran, Suzuki Ravi, Suzuki Swift, Suzuki Wagon R, and Toyota Corolla. utilizing the AU-AIR dataset, the authors of [2] describe innovative methods for monitoring and securing aerial images of traffic utilizing state-of-the-art and widely-used deep learning (DL) object identification models (Faster-RCNN, SSD, YOLOv3, and YOLOv4). Due to the extreme inequity present in this data set, an additional 500 photos

were harvested by web-mining methods. This work's original contribution is twofold. To begin, this essay uses a scientific approach to highlight how useless ground-view photos are for aerial object recognition. Second, the efficacy of these algorithms has been analyzed by doing a regress comparison. Eight different CNN architectures were employed to evaluate how well they performed on the Analytics Vidhya Emergency Vehicle dataset [3]. For the purpose of collecting anomalies from traffic cameras and properly determining when they began and ended, [4] suggests using a Decision-Tree enabled technique driven by Deep Learning. Anomaly detection and analysis were part of their methodology when a detection model was developed. The detection model we built was based on YOLOv5. [5] In their study, they offer a novel method for categorizing vehicles, drawing attention to the difficulties of working with unbalanced data. Both the MIOvision Traffic Camera Dataset and the Beijing Institute of Technology Vehicle Dataset are mined for their raw information. To further improve the quality of the gathered vehicle photos and to recognize cars from the denoised images, we further use adaptive histogram equalization and the Gaussian mixture model. The feature vectors are then extracted from the identified cars using the Steerable Pyramid Transform and the Weber Local Descriptor. A deep learning ensemble method is then fed the derived characteristics to classify vehicles. Generative adversarial networks (GANs) were created by [6] to analyze traffic surveillance footage and identify automobiles. There are three stages to the methodology used for categorizing vehicles. For the first training, GAN used a traffic dataset. This allowed it to provide adversarial samples for the uncommon classes. After training an ensemble-based Convolutional Neural Network (CNN) on the skewed dataset, the second step is to perform sample selection to get rid of the adversarial examples of poorer quality. Last but not least, the chosen adversarial samples were used to fine-tune the ensemble model on the enhanced dataset. Extensive trials shown that the proposed GAN technique performed well in vehicle classification on MIO-TCD as measured by the Cohen kappa score, mean recall, precision, and the mean precision. However, as the deeper networks are set to converge, the created GAN technique would encounter degradation problems. Based on a hierarchical multi-SVM (multi-Support Vector Machine) classifier, [7] created a novel method for classifying vehicles. After developing a hierarchical multi-SVM approach for vehicle classification, we began by isolating foreground objects from surveillance recordings. For the performance assessment, we also employed a voting-based rectification mechanism to keep tabs on the cars' classifications. In this research paper, we apply the hierarchical multi-SVM method to the real-world problem of accurate vehicle categorization in dense traffic. Since diverse perspectives, shadows, and high occlusion make the presented approach useless in real-world busy traffic scenarios. The little YOLO was used with a support vector machine classifier in [8] to identify and categorize vehicles. In the experimental phase, the created model's accuracy and recall were verified using the BIT Vehicle Dataset. The experimental results verify that the created model accurately categorizes the vehicle kinds in live-streaming traffic recordings. A significant drawback of this research was that SVM was a binary classifier, which only allows for binary categorization. Based on the more rapid R-CNN method, [9] created a system to categorize different kinds of automobiles. The suggested method was tested using a real-time dataset consisting of photographs of the intersection taken at the same time. Detecting an obscured vehicle in photos that vary in brightness, perspective, and size requires an innovative approach that may be refined in the future. [10] created a convolutional neural network (CNN) model for vehicle categorization that features precise

tweaking and pretraining. The ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) dataset was used for the pretraining phase's application of GoogLeNet in order to get the first model with connection weight. In order to acquire a final classification, the derived initial model was fine-tuned on the vehicle dataset. Van, minibus, truck, bus, automobile, and motorbike are only some of the six vehicle types included in the gathered highway surveillance footage used in this research review. During the experimentation phase, the vehicle dataset was analyzed for performance. However, the created CNN model is computationally costly and suffers from the serious issue of "overfitting." [11] created a novel framework for classifying automobiles, which consists of six stages including data pretreatment, vehicle detection, vehicle tracking, structure matching, feature extraction, and vehicle classification. Data preparation was achieved by noise reduction and color conversion after the collection of traffic surveillance recordings. Vehicles were identified with the use of an Otsu thresholding method and a background removal technique. The Kalman filter was then used for vehicle tracking in order to keep tabs on the cars in motion. An artificial neural fuzzy inference system (ANFIS) was utilized to categorize the automobiles based on the feature vectors extracted using a log Gabor filter and a Harrish corner detector. Extensive experimental evidence demonstrated that the established framework significantly improved upon previous attempts at vehicle categorization in terms of mistake rate and accuracy. The dimensionality problem that is responsible for the complexity of the model is made worse by the established framework. A novel semisupervised CNN architecture for auto-classification was introduced in [12]. The designed architecture makes use of a sparse Laplacian filter to extract detailed and unique vehicle information. A multitask-learned softmax classifier was used to classify vehicles in the final layer. The characteristics learnt by the semisupervised CNN architecture in this literature review were sufficiently discriminatory to achieve notable performance in the complicated scenarios. To assess the effectiveness of the designed architecture in terms of classification accuracy, extensive tests were performed on the BIT Vehicle Dataset and a publicly available dataset. Since the semisupervised CNN design has several layers, training takes much longer. For the purpose of vehicle type classification on the BIT Vehicle Dataset and the MIO-TCD, [13] presented a novel densely connected single-split super learner and applied modifications. The created model was straightforward; improved performance in vehicle type categorization was achieved without the use of logic reasoning or specially built characteristics. The created approach brings the vanishing gradient problem to the complex datasets, which is a crucial issue in this research. Using Principal Component Analysis Convolutional Network (PCN), [14] constructed a novel semisupervised model for vehicle type classification. Convolutional filters were used in the created model to extract the discriminative and hierarchical features. The proposed model outperformed the baseline in simulations because it is resistant to noise contaminations, lighting conditions, rotation, and translation, all of which might affect performance in real-world settings. Overfitting occurs because the created PCN model has more parameters for training. For the purpose of vehicle classification, [15] created the Sparse-Filtered Convolutional Neural Network with Layer Skipping (SF-CNNLS) method. Three SF-CNNLS channels were used in this research to extract discriminant and comprehensive vehicle attributes. In addition, information on the cars' color, brightness, and shape was retrieved from the three channels of each picture. The Experimental Results and Discussion section includes benchmark performance validation results for the SF-CNNLS method. Truck, minivan, bus, passenger,

taxi, automobile, and SUV classes were ultimately categorized using a softmax regression classifier. Higher-level layers were included in the constructed softmax regression classifier, although incorporating lower-resolution vehicle photos might lead to a loss of vehicle type information. Lightweight vehicle categorization was made possible by [16]'s attention-based method and deep CNN technology. In this paper, the authors use specificity, precision, and -score to verify the generated model's performance on a real-world dataset. Baseline, camera jitter classes, and adverse weather conditions were all areas where the generated model fell short. Each literature study includes a detailed explanation of the methodologies used, the datasets used, the benefits and drawbacks of applying the created approaches to the task of vehicle type classification, and so on. This study proposes a novel ensemble deep learning method to enhance vehicle type categorization, which aims to solve the aforementioned problems. The purpose of [17] is to develop a monitoring and detection system that can adapt to deep structural changes. Among the issues faced by their approach are the recognition of several moving entities with limited resources. In order to create a real-time, smart identification and tracking system, this technique integrates principal component analysis with deep learning networks. By switching between the two methods dynamically, performance may be improved over standard detection and tracking methods. An improved active obstacle-separation model was presented in [18], using push and drag-push operations to physically remove obstacles from the target over the course of three stages. With the exact coordinates of the obstruction, the push and drag vectors may be determined and streamlined. More importantly, the unique strategy used a hybrid vision-based control system, as opposed to the system that just "looked" once during the whole picking operation. Using an increment Gaussian mixture model (IGMM) to model a track's previous appearance distance, [19] proposed a hybrid track association (HTA) technique that combines the obtained statistical data into the assessment of the detection to-track association cost. An architecture based on multiple object tracking (MOT) was presented in [20]. The KCF tracker is used to follow the paths taken by the objects recognized by Fast RCNN. Bounding box matching using the Hungarian method is used to acquire historical trajectory data for better recognition accuracy. A method for following an object detected by a NN system was suggested in [21]. Data collected in Danish coastal waters are used to determine the efficacy of the suggested method. To ensure a good feature that is typical for the presented object and to save time over calculating a new feature, the method employs a feature that is assessed in the detection phase. The issue of automated pedestrian counting and tracking was addressed in [22], which detailed a solution to the problem. First, a backdrop modeling strategy is used to proactively acquire multi-pedestrian candidates; next, categorization is used to authorize the candidates. Next, real-time TLD (Tracking-Learning-Detection) may be used to manage all the pedestrian patches and arrive at a new forecast position based on similarity metrics. in [23] new MODT techniques were introduced. The given method employs the best Kalman filtering technique to follow the shifting object in the video frame. The region growth technique was used to convert the movie into a morphological operation based on the frame count. After performing object differentiation, the probability-based grasshopper method was used to optimize Kalman filtering parameters. The best possible setting allowed a similarity measure to follow the target object through all frames. Background subtraction and the K-means clustering approach were used to create a multi-object identification and tracking model by the authors of [23]. The given method can compensate for camera shake, object occlusion, and shadows. Objects in

motion are isolated from the rest of the data using K-means clustering after the background is removed. Moving objects may be combined or separated using the system's spatial awareness. introduced novel MODT procedures [24]. The given method employs the best Kalman filtering technique to follow the shifting object in the video frame. The region growth technique was used to convert the movie into a morphological operation based on the frame count. After performing object differentiation, the probability-based grasshopper method was used to optimize Kalman filtering parameters. The best possible setting allowed a similarity measure to follow the target objectsthrough all frames. Background subtraction and the K-means clustering approach were used to create a multi-object identification and tracking model by the authors of [25]. The given method can compensate for camera shake, object occlusion, and shadows. Objects in motion are isolated from the rest of the data using K-means clustering after the background is removed. Using spatial context, it can also manage the fusion and separation of in-motion objects.

III. PROBLEM STATEMENT

One of the most challenging aspects of object recognition is the fact that the same thing might seem to be an entirely different thing depending on the viewing angle. There are a great number of interesting things that are not rigid bodies and are capable of being deformed in extreme ways. As a result, it is simpler for an AI system to identify and keep track of objects. The completely linked layer, in which the label class is discrete but the label distance is continuous, is the source of the problems that plague the present CNN. At the very end of the network, there needs to be a layer of regression in addition to a layer of classification so that this problem can be solved. Additionally, using an unsupervised approach, it would be an innovative idea to estimate the video using object identification and tracking algorithms.

IV. PROPOSED WORK

The article's major objective is to recognize and monitor many objects online or in realtime applications, and the suggested system's structure or framework consists of a number of phases or procedures to get there. The emergence or growth of identification code substitution poses a threat to most multi-object tracking approaches. Only with a massive status rating scale can the data relevance parameter be considered legitimate or reliable. Because of this, tracking tracks along these corridors are often disrupted if obstacles emerge in the video. As a consequence, a more accurate adaptive scale was required to replace the data correlation scale by including data from both motion and appearance. To do this, convolutional neural networks were trained on a data set using deep learning and then applied to the Multilayer stacked YoLoV8 algorithm for object recognition and tracking. The suggested algorithm's resilience was improved by training on a large, user-specified data set, with the condition that the system's ease of use, efficiency, and viability in real-time settings must be maintained. Since traffic cameras have become so commonplace, researchers in the area of computer vision have been more interested in image-based methods. The data flow diagram connected with the deep learning approach that was applied is shown in Figure 1.

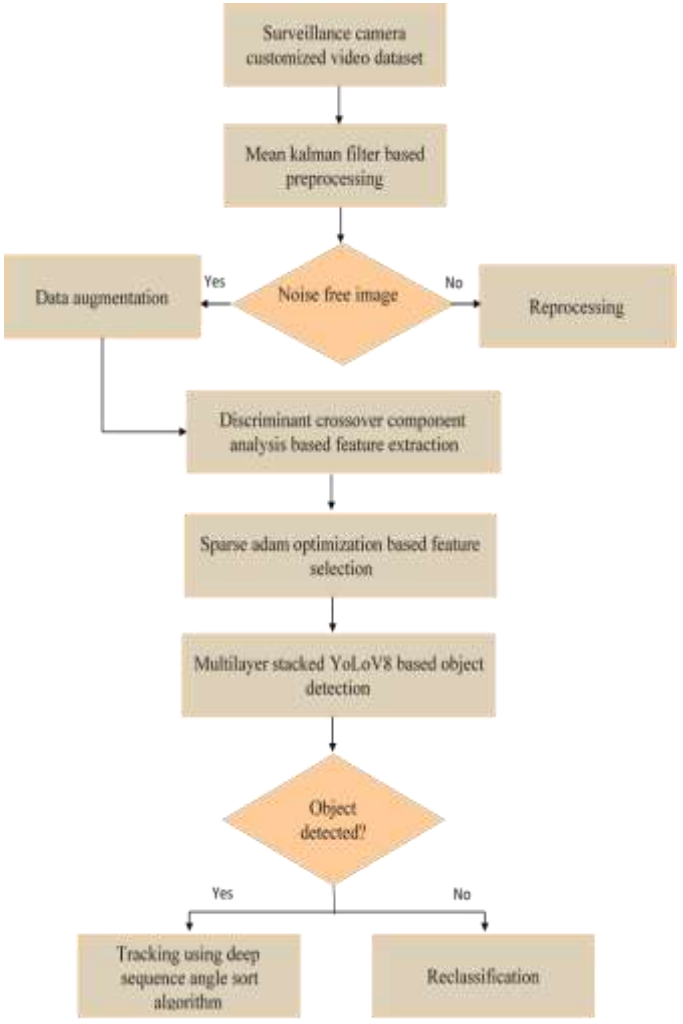


Figure 1 Pictorial representation of the suggested methodology

a. Dataset

The video captured by a highway-mounted surveillance camera is what is used for statistical analysis. Each dataset is unique since it was shot at a different time of day and under varied lighting conditions.

b. Preprocessing

"The Mean Kalman filter is a method for linear least-squares estimation. The MKF algorithm consists of two loops: one to calculate the gain, and another to calculate the filter. Gain is computed through a feedback loop that also factors in estimate and prediction errors. Both state prediction and state estimation are part of the filter's calculating loop.

To express the covariance of the observation noise R, we use the mean squared deviation of the data at each landmark. At this one time in the experiment, put Q = 0.001, = 1, and H = 1 for the system process noise, state transition, and observation vectors, respectively. The initial estimated error covariance of a reference point is derived using MKF filtering.

$$Q_1 = \frac{1}{N}\sum_{l=1}^{N}\left(Y_1^l - E\left[Y^l\right]\right)^2 \quad (1)$$

where $Y_1^l$is the initial information from the sample at l. $E\left[Y^l\right]$is the number of points used to calculate the expectation of l's data, and N represents the total number of points used. For l, we get the following results. The formula for the t-th sample's covariance of prediction errors is

$$Q_{u,u-1}^l = Q_{u-1}^l + R \quad (2)$$

where Q_{t-1}^l represents the $(t-1)$ th estimation sample variance correlation. Gain in a filter for data from the t th sample, called J_t^l, is

$$J_t^l = Q_{u,u-1}^l\left[Q_{u,u-1}^l + R\right]^{-1} \quad (3)$$

For the t^{th} set of samples, we have an estimated error covariance of

$$Q_{u-1}^l = \left[I - J_u^l\right]Q_{u,u-1}^l\left[I - J_u^l\right]^T + JRJ_u^{l^T} \quad (4)$$

where I symbolizing a vector unit. The expression for the t th data's anticipated value is

$$Y_{u,u-1}^l = \Phi Y_{u-1}^l \quad (5)$$

The estimated value of the t-th sample is obtained by inserting the filter gain into the filter calculation loop.

$$Y_u^l = Y_{u,u-1}^l + J_u^l\left[Y_u^l - Y_{u,u-1}^l\right] \quad (6)$$

where Y_u^l is the u th sample.

In order to get more precise data, the measuring process' error may be decreased by the MKF. We guarantee real-time accuracy for the spots to be found for the real-time video database.

c. Feature extraction

The features were extracted in our study using the DCCA-based unsupervised FE approach. There aren't many approaches that have been presented in the literature that deal with PCA-based unsupervised feature extraction; nonetheless, we are enhancing the DCCA method,

which is briefly explained below. Let's say that feature profiles are presented as a matrix, $y_{ij} \in \mathbb{R}^{N\times M}$, which reflects the histone modification of the i th areas in the j th samples. We'll assume $y_{ij}$is normalized as in this case.

$$\begin{aligned} &\textstyle\sum_{i=1}^{N} y_{ij} = 0 \\ &\textstyle\sum_{i=1}^{N} y_{ij}^2 = N \end{aligned} \tag{7}$$

Applying DCCA to x_{ij} should obtain

$$y_{ij} = \textstyle\sum_{\ell=1}^{min(N,M)} \lambda_\ell v_{\ell i} u_{\ell j} \tag{8}$$

where

$$\sum_i v_{\ell i} v_{\ell' i} = \sum_j u_{\ell j} u_{\ell' j} = \delta_{\ell\ell'}$$

To get $v_{\ell i}$, the vector and eigen feature values of$\sum_j y_{ij} y_{i'j} \in \mathbb{R}^{N\times N}$ must be obtained as

$$\textstyle\sum_{i'} \left(\sum_j y_{ij} y_{i'j}\right) v_{\ell i'} = \lambda_\ell u_{\ell i} \tag{9}$$

and $v_{\ell j}$ can be obtained as

$$v_{\ell j} = \textstyle\sum_i v_{\ell i} x_{ij} \tag{10}$$

To choose areas of interest, it is necessary to first identify the $u_{\ell j}$linked to the desired property. The property of importance in this research is object outline independence from samples, or that $u_{\ell j}$should be independent of j. After locating the l of interest, an ideal standard deviation (SD) of $u_{\ell i}$is discovered, ensuring that $u_{\ell i}$ as closely resembles the Gaussian distribution as feasible (null hypothesis).

The following steps may be taken to optimize SD. P-values are first assigned to the ith area as

$$Q_i = Q_{\chi^2}\left[> \left(\frac{v_{\ell i}}{\sigma_\ell}\right)^2\right] \tag{11}$$

"where $P_{\chi^2}[> x]$ is the cumulative χ^2 distribution where the argument is larger than x and σ_ℓ is the SD. Then, the feature error h_s of P_i is computed, which is the number of i s that satisfy"

$$\frac{t}{N_t} \le 1 - Q_i \le \frac{t+1}{N_t}, 0 \le t \le N_t - 1 \tag{12}$$

In this case, it is anticipated that h_s will have constant values for $t \leq t_0$, with some s_0, and a strong peak at$k_0 < k$. Is chosen to be part of $h_t, t > t_0$in this instance are chosen based on correlation with major histone modification. After that, Q_i s are once again calculated using an optimized SD, the results are rectified, and i s with altered Q_i (features) are chosen.

d. Feature selection

Sparse Adam optimization algorithm is an adaptive learning rate optimization technique. This optimization algorithm has been designed for deep learning approaches. The great importance of the algorithm is that it finds individual adaptive learning rates for various parameters. The name of algorithm based on the adaptive moment estimation procedure of the algorithm. The first and second moment estimations of gradients are used for adapting learning rate for every weight of the deep neural network. The first and second moments are known as mean and variance, respectively. Sparse Adam algorithm uses exponentially moving averages for estimation of moments in each batch at every iteration. The update feature selection rule for Adam optimizer can explain some mathematical formulas as follows:

$$N_t = \beta_1 N_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 U_{t-1} + (1-\beta_2) g_t^2 \quad (13)$$

where m and U are moving averages, g is gradient on current batch, t is iterations, and β_1 and β_2 are specialised batch characteristics of the technique. Eq. (13) shows gradient and squared gradient moving averages.

Zero moving averages start the iteration. One may make sense of the association of moving averages with the moments by studying the anticipated values of moving averages. Since our zero initialization biases the optimizer towards zero, the first and second moments must be corrected.

After bias correction for specialised feature estimators, the anticipated value becomes the desired value. Eq.(14) gives bias-corrected first and second moment estimators.

$$\widehat{N_t} = \frac{N_t}{1-\beta_1^t}$$
$$\widehat{U_t} = \frac{U_t}{1-\beta_2^t} \quad (14)$$

The technique ends with moving averages scaling the learning rate for each parameter. For implementing this step, calculate the weighted feature update as Eq. (15).

$$U_t = U_{t-1} - \alpha \frac{\overline{N_t}}{\sqrt{\widehat{U_t}+\epsilon}} \quad (15)$$

where w is feature weight, α is learning rate or step size, t is the number of iteration and ε is to avoid division by zero and the default value is 〖10〗^(-8). Eq. selects specialised features. (16).

$$Y_n\left(e^{j\Omega}\right) = \sum_{m=-\infty}^{\infty} Y[N]U[n-m]e^{-j\Omega m} \quad (16)$$

where $\Omega$is frequence point variable, n and m are time indicators and $w[n]$ is the window function.

e. Object detection

Our multi-layer stacked YOLO V8 model is described in the following comprehensive analysis. To construct the first input layer, we assume that the input is an image, r represents the number of rows in the image, c represents the number of columns in the image, and s represents the sliding step which is is T_1; An equation illustrating the computational cost of obtaining a feature map is presented below:

$$(Y_1X_1)\frac{r-Y_1+T_1}{T_1}\frac{c-X_1+T_1}{T_1}. \quad (17)$$

Taking the computing area to be the size of the convolution kernel area, we obtain (17), and then we assume that the first layer has feature maps, so the first layer is calculated as follows:

$$\frac{r-Y_1+T_1}{T_1}\frac{c-X_1+T}{T_1}(Y_1X_1n_1) \quad (18)$$

After convolution, the feature map will have a size of,

$$\begin{aligned} r_1 &= \left\lceil\frac{r-Y_1+T_1}{T_1}\right\rceil, \\ c_1 &= \left\lceil\frac{c-Y_1+T_1}{T_1}\right\rceil. \end{aligned} \quad (19)$$

Since the maximum downsampling layer does not change the number of feature maps, the number of feature maps is equal to the number of previous feature maps. Based on the downsampling window size, the downsampled feature map would have a size of n_1

$$\begin{aligned} Y_2 &= \frac{r_1}{r_2}, \\ X_2 &= \frac{D_1}{D_2}. \end{aligned} \quad (20)$$

In order to calculate the total number of feature maps,

$$n_2r_2D_2Y_2X_2 \quad (21)$$

The following is the second convolution layer assuming the number of features is large

$$\begin{aligned} r_3 &= \left\lceil\frac{r_2-Y_3+T_3}{T_3}\right\rceil, \\ D_3 &= \left\lceil\frac{D_2-Y_3+T_3}{T_3}\right\rceil. \end{aligned} \quad (22)$$

The following is how the computation for the convolution operation will look when using the top layer of the feature map.

$$(Y_3 X_3 n_3)\left\lceil\frac{r_2 - Y_3 + T_3}{T_3}\right\rceil\left\lceil\frac{D_2 - X_3 + T_3}{T_3}\right\rceil. \quad (23)$$

Assuming that the output of the second-layer maximum downsampling layer is defined by the downsampling window size and the step size s_4, computation of the entire quantity of the layer may be accomplished using the same approach.

$$\begin{aligned} Y_4 &= \frac{r_3}{r_4}, \\ X_4 &= \frac{D_3}{D_4}. \end{aligned} \quad (24)$$

As can be seen above, the MaxPool2 output feature size is n_4. When using the inception structure, calculations are performed from left to right with a step size of 1. The conceptual model and mathematical demonstration of the inception structure of the third layer are presented in.

$$\begin{aligned} & n_4 \times 1 \times 1 \times 64 \\ & n_4 \times 1 \times 1 \times 96 + 96 \times 3 \times 3 \times 128 \\ & n_4 \times 1 \times 1 \times 16 + 16 \times 5 \times 5 \times 32 \\ & n_4 \times r_4 \times c_4 + n_4 \times 1 \times 1 \times 32 \end{aligned} \quad (25)$$

All four layers of Inception can be calculated in this manner. Next is the fifth layer of the convolution, and the total calculation is,

$$\begin{aligned} 1 \times \quad & 1 \times r_5 \times c_5 \times 512 + 2 \times 3 \times 3 \times (r_5 - 3 + 1) \\ & \times (D_5 - 3 + 1) \times 1024 \\ & +3 \times 3 \times \left(\frac{r_5 - 3 + 2}{2}\right)\left(\frac{c_5 - 3 + 2}{2}\right) \times 1024. \end{aligned} \quad (26)$$

Due to structural similarities between the sixth and fifth layers, the result is the same as in (26).

If n is an integer from 1 to n-1, then L represents the pyramid layer, and so on. This pyramid's base layer's calculated quantity is

$$\sum_{n=1}^{L}\left(n_5 r_{6,n} D_{6,n}\left\lceil\frac{r_5}{r_{6,n}}\right\rceil\left\lceil\frac{D_5}{D_{6,n}}\right\rceil\right) \quad (27)$$

The eighth layer is entirely joined. Let's assume n_6 input features and n_7 output features are available. Since the previous layer serves as its input, this layer will be handled after all the map's features have been collected as a vector, and so n_6 ,

$n_6 = \sum_{n=1}^{L} n_5 r_{6,n} D_{6,n}$ (28)

The full-connection layer's computational cost is low since it is based on the original YOLO V8 network and uses the same calculation approach as the neural network.

$$n_7 = \left(2\sum_{n=1}^{L} n_5 r_{6,n} D_{6,n} + 1\right). \quad (29)$$

Overall network calculation, input layer image size, convolution kernel size, and number of convolutional layers all reveal that network depth and breadth are having a significant influence, as described above in the examination of network design. Now it's possible to do multi-object detection.

f. Object tracking

One such improved method of tracking is deep sequence angle (DSA) -SORT. Short-term and permanent occlusions are no problem for deep sequence angle SORT. All feasible assignments (trajectories) between the target object in f_t and all existing track-lets in earlier frames are first calculated. Second, each target objects has its optimal assignment projected. When all classes' track-lets from earlier frames are taken into account, it slows down tracking and expands the search area. This problem is solved by DSA-SORT, which only considers assignments between the target object and existing track-lets of the same class.

Let $\{G: G_1, G_2, \dots, G_n\}$ be the collection of n trajectories having varying numbers of track-lets. Consider the appearance descriptor of the trajectory to be the set $\left\{G_\beta: G_\beta^1, G_\beta^2, \dots, G_\beta^\xi\right\}$. ξ implies the total number of track-lets present in β^th" " trajectory. Let $\{S: S_1, S_2, \dots, S_v, \dots, S_m\}$be the set of m new objects identified in the current frame, as acquired via multilayer layered YOLO V8. In DSA -SORT, we find the trajectory with the highest similarity to the β^{th} newly detected object by computing the similarity (with a metric Q). Q is the distance between the β^{th} trajectory $(N_\beta)$and the β^{th} th newly identified object(o_v),

$$Q(t_\beta, o_v) = \begin{Bmatrix} (\lambda_1)E(N_\beta, o_v) + (1-\lambda_1)A(N_\beta, o_v), if\ o_v^k = N_\beta^k \\ 0, otherwise \end{Bmatrix} \quad (30)$$

where $k \in c, \beta \in n, v \in m, E(N_\beta, S_v)$, and $A(N_\beta, S_v)$are two distinct measures of separation in a state space that is 8 dimensions deep (including position, orientation, size, and speed). These measures are designed to account for both temporary and permanent obstructions while monitoring o_v. The more similar two things are, the lower the value of Q should be. The influence of E and A on Q is regulated using using λ_1. Metric $E(N_\beta, S_v)$ between N_β and o_v is:

$$E(N_\beta, S_v) = \sqrt{S_v - N_\beta^1} \quad (31)$$

where t_β^1 denotes the predicted 8-dimensional Kalman state of the latest track-let of β^{th} trajectory. Metric $A(t_\beta, S_\nu)$, in contrast, is characterised by the smallest possible cosine distance among ξ track-lets of t_β and S_ν, as

$$A(N_\beta, S_\nu) = min\left\{1 - cos\ \left(S_\nu, N_\beta^{k_1}\right)\right\} \qquad (32)$$

where $cos\ \left(S_\nu, N_\beta^{k_1}\right)$ represents the cosine similarity between o_ν and $N_\beta^{k_1}$, and $k_1 = 1,2, \dots, \xi$.

The relationship between $S_\nu$and t_β is addressed in Eq. (31), while in Eq. (32) it is only done for the most recent track-let of t_β. To further reduce the search space and thus the computation time while computing $Q(N_\beta, S_\nu)$ one may put some thresholds, say, $N_E$and N_A on E and A values, respectively, for excluding the unlikely association of the newly detected object. The distance between $S_\nu$and N_β is defined by the metric $Q(N_\beta, S_\nu)$ (Eq. (10). So, if there are n trajectories, there will be n Q-values for theS_ν. The $S_\nu$is predicted to follow the path with the lowest Q value. In other words, the o_ν association rule is ,

$$D(N_\beta, S_\nu) = \begin{Bmatrix} 1, if\ Q(N_\beta, S_\nu) = min\{Q(N_\beta, S_\nu) \neq 0\} \\ 0, otherwise \end{Bmatrix} \qquad (33)$$

Each identified object undergoes the aforementioned tracking procedures in order to determine whether or not its trajectory has already been matched with it. If an existing trajectory does not get associated with any newly detected object, then, one null track-let record is added to its history. If a trajectory has more than some threshold number of null track-lets (call itN_{max}), it is assumed to have left the scene and removed from the trajectory set N. Finally the object can be tracked precisely".

Algorithm: MS-YOLOV8_((DSA) –SORT)

"**Input:** Size of feature space, training set, size of feature subspace, feature set, number of feature subspace, one test sample, and number of object classes.
Output: Classification of camera surveillance data. Process:
For y = $\{G: G_1, G_2, \dots, G_n\}$classes Label the samples of i th class.
Train the feature subsets using YOLO.
Train {70%) Test {30%)
Neuron Layers {code input}
Neural size{Number of layers}
Ranking feature set utilizing
$\sum_{n=1}^{L} \left(n_5 r_{6,n} D_{6,n} \left\lfloor \frac{r_5}{r_{6,n}} \right\rfloor \left[\frac{D_5}{D_{6,n}}\right]\right)$'s last ranked featur
Begin

conv{

$$n_7\left(2\sum_{n=1}^{L} n_5 r_{6,n} D_{6,n} + 1\right).$$

}

For

output $\left\{G_\beta : G_\beta^1, G_\beta^2, \ldots, G_\beta^\xi\right\}$

Else

$$\{S : S_1, S_2, \ldots, S_v, \ldots, S_m\}$$

Out {

(DSA) -SORT }

Trajectory distance calculation$\{\beta^{th}\}$

$$Q(t_\beta, o_v) = \begin{Bmatrix}(\lambda_1)E(N_\beta, o_v) + (1-\lambda_1)A(N_\beta, o_v), if\ o_v^k = N_\beta^k \\ 0, otherwise\end{Bmatrix}$$

For

Updation;

{Seperation distance $k \in c, \beta \in n, v \in m, E(N_\beta, S_v)$, and $A(N_\beta, S_v)$}

{Tracklets similarity{ $A(N_\beta, S_v) = min\left\{1 - cos\ \left(S_v, N_\beta^{k_1}\right)\right\}$ }

Track objects{1…….n}

Trajectory (Anal{1…n})

End for

End

Calculate the value of counter Output.

Return

End

End"

V. PERFORMANCE ANALYSIS

To assess the effectiveness of the proposed system, performance tests and evaluations were conducted using large datasets with many items. The roadway dataset is derived from these data. This information includes a variety of situations, including video clips recorded by a moving camera, surveillance footage from various settings and occasions, etc. Then, as shown in Fig. 1, MS-YOLOV8_((DSA) -SORT) was trained networks were used to improve object detection performance.The test was conducted on a machine with an Intel Core i3-3370, 2 GB of RAM, and a 2.73 GHz clock speed. The approach was created using Python, a widely known and free programming language. Python has a large library of pre-existing modules. The video frame packages for computer vision are used to accomplish the required work in Python. Its cheap cost is partly due to the resources required to conclude the procedure.

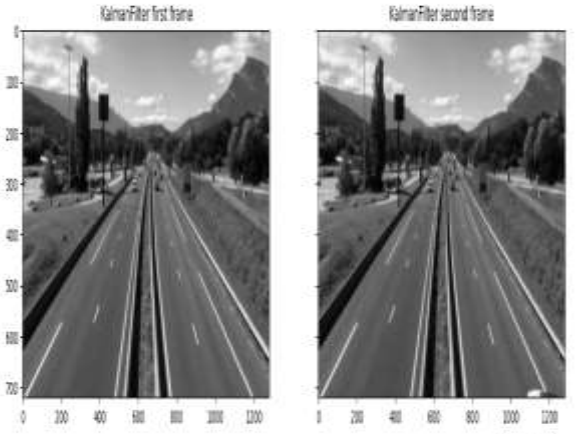


Figure 2 Preprocessed output

As of from the figure 2 the noises are removed and the processed image are illustrated in grey scale.

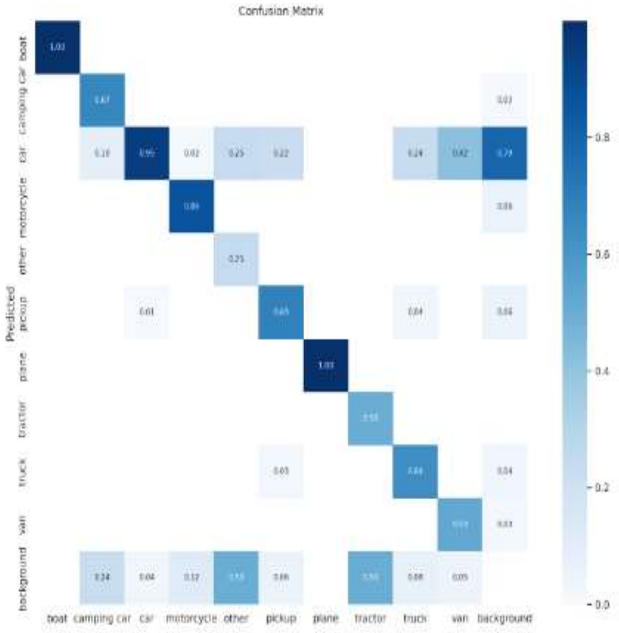


Figure 3 Confusion matrix for object detection

Figure 3 shows the confusion matrix for the actual classes and the anticipated classes. While estimates are presented in the horizontal axis, the ground truths for the item labels are listed in the vertical axis.

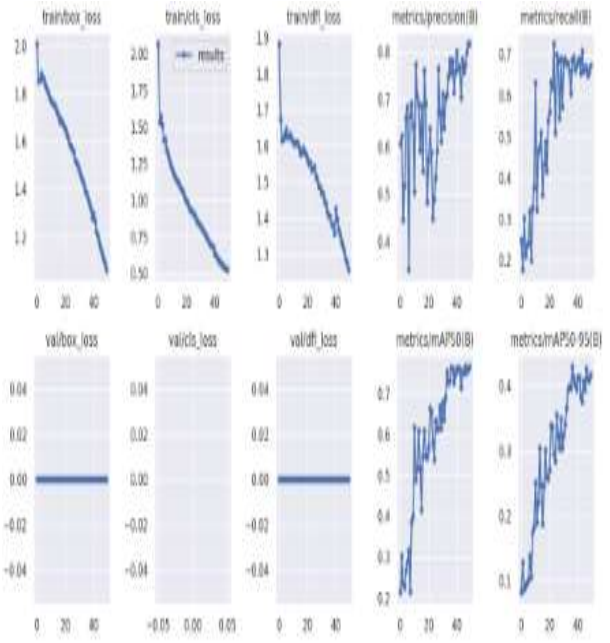
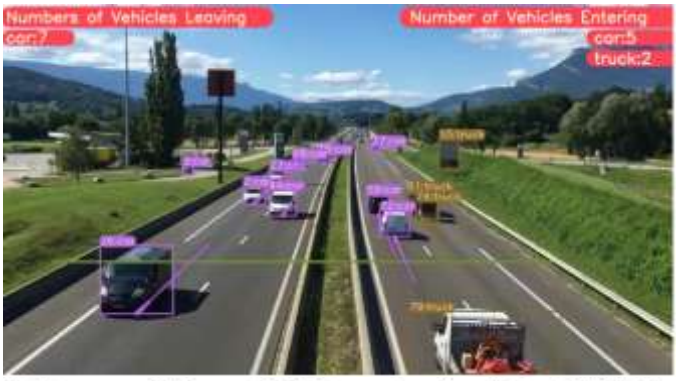


Figure 4 Epochs Vs. Loss

The loss function is a quantifiable measure of loss for all of the time and is constructed from all of the data in that era. It is inevitable that some data will be lost throughout the iterative process of creating a curve. The resultant curve shows that the costs associated with classifier training, validation, and testing are manageable when compared to alternative methods. Our model is likely underfitted if there is just a little discrepancy between the training loss and the validation loss. Maximising the size (either the number of layers or the raw number of neurons in each layer) might reduce the training loss. Figure 5 depicts the data used to calculate the loss estimate. The proposed method beats comparable approaches because it suffers from far less severe levels of level loss.




```
Model summary: 268 layers, 43614318 parameters, 0 gradients, 164.9 GFLOPs
video 1/1 (1/808) /content/YOLOv8-DeepSORT-Object-
Tracking/ultralytics/yolo/v8/detect/test4.mp4: 384x640 11 cars, 41.5ms
video 1/1 (2/808) /content/YOLOv8-DeepSORT-Object-
Tracking/ultralytics/yolo/v8/detect/test4.mp4: 384x640 11 cars, 40.7ms
video 1/1 (3/808) /content/YOLOv8-DeepSORT-Object-
Tracking/ultralytics/yolo/v8/detect/test4.mp4: 384x640 11 cars, 40.6ms
video 1/1 (4/808) /content/YOLOv8-DeepSORT-Object-
Tracking/ultralytics/yolo/v8/detect/test4.mp4: 384x640 11 cars, 36.2ms
video 1/1 (5/808) /content/YOLOv8-DeepSORT-Object-
Tracking/ultralytics/yolo/v8/detect/test4.mp4: 384x640 11 cars, 26.9ms
video 1/1 (6/808) /content/YOLOv8-DeepSORT-Object-
Tracking/ultralytics/yolo/v8/detect/test4.mp4: 384x640 12 cars, 26.9ms
```

Figure 5(a)

```
val: Scanning /content/YOLOv8-DeepSORT-Object-Tracking/ultralytics/yolo\v8/detect/detection-1/valid/labels.cache... 619 images,
1 backgrounds, 0 corrupt: 100% 619/619 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 39/39 [00:27<00:00,  1.42it/s]
                   all        619       1185      0.757      0.667      0.764      0.429
                  boat        619          1      0.472          1      0.995      0.796
           camping car        619         21      0.866      0.667      0.735      0.379
                   car        619       1963      0.882      0.918      0.911      0.422
            motorcycle        619         58      0.859      0.678      0.844      0.326
                 other        619          4      0.32       0.25       0.373      0.232
                pickup        619         67      0.773      0.66       0.723      0.383
                 plane        619          1      0.89           1      0.995      0.697
               tractor        619          6          1      0.482      0.757      0.275
                 truck        619         15      0.673      0.56       0.629      0.394
                   van        619         38      0.831      0.658        0.7      0.393
Speed: 0.5ms pre-process, 36.3ms inference, 0.0ms loss, 1.4ms post-process per image
```

Figure 5(b)
Figure 5 Simulated output

The overall simulated output for the suggested classified was depicted in figure 6.

Dice score :

N and D represent the found data qualities and the attributes of the underlying truth data, respectively. This allows us to calculate the Dice coefficient using the formula given in Eq. (21).

$$D(N,D) = \left(2D \cap \frac{N}{D} + N\right) = \frac{2(True\ Positive)}{2\ (True\ Positive + False\ Negative + False\ Positive)} \quad (34)$$

Jaccard score:

The Jaccard score, which is obtained using Eq. (22), is used to compare how similar the two groups are,

$$J(N,D) = \frac{N \cap D}{N \cup D} = \frac{N \cap D}{o+p-(o \cap p)} = \frac{True\ Positive}{True\ Positive + False\ Negative + False\ Positive} \quad (35)$$

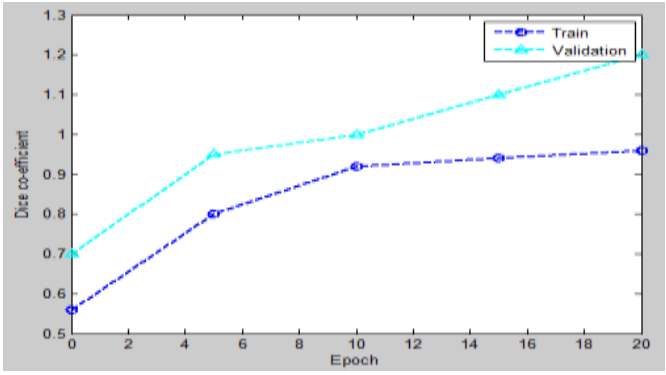


Figure 6(a) Dice Co-efficient over epochs

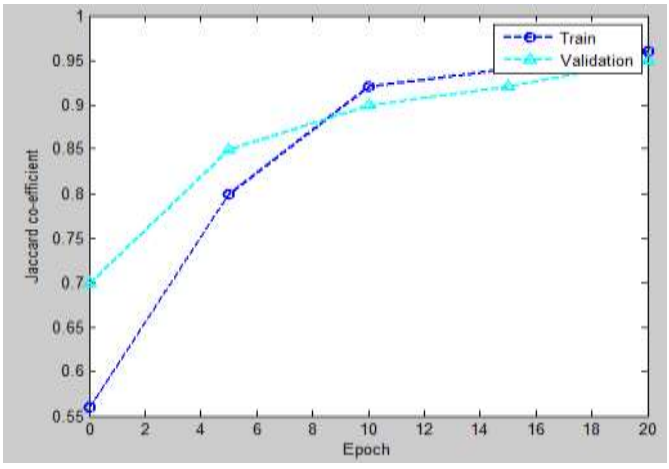


Figure 6 (b) Jaccard score over epochs

In medical image processing, the Dice score and Jaccard index are often used metrics for assessing classification tasks. The results demonstrate that the suggested approach has high Jaccard and dice values when compared to other methods.

It may be compared to the many existing mechanisms in order to show that the suggested technique is efficient [5],[17]. The proposed system's performance is assessed in terms of accuracy, precision, F1 score, and recall.

Accuracy

It determines the proportion of items that accurately classify their object status. It establishes how closely the results match actual ground truth discoveries.

ACCU =(TP+TN)/(TP+TN+FP+FN) (36)

Precision

It evaluates the behaviour of the suggested approach's accuracy and makes use of that knowledge by deleting the dataset's aberrant object status.

Precision= TP/(TP+FP) (37)

Recall

The ratio of correctly predicted instances and all instances.

Recall= TP/(TP+FN) (38)

F-Score

"Only two measurements are needed to determine the F-measure (precision and recall)

"F measure =

$$2* \frac{(Precision * Recall)}{(Precision+Recall)}\text{,,} \quad (39)$$

Table 1 Performance analysis of the suggested methodology

"Method	Type	Accuracy(%)	Precision(%)	FM(%)	Recall(%)
KCF [17]	Batch	51.43	76.83	42.33	62.66
IOU [17]	Online	44.00	76.13	57.54	54.57
DeepSORT [17]	Online	54.8	79.78	46.73	55.38

Our Tracker1 [17]	Online/Real	60.95	78.00	72.07	66.36
FCNN[17]	Online/Real	63.11	78.62	58.74	72.82
Our Proposed	Real	96.8	97	90	94”

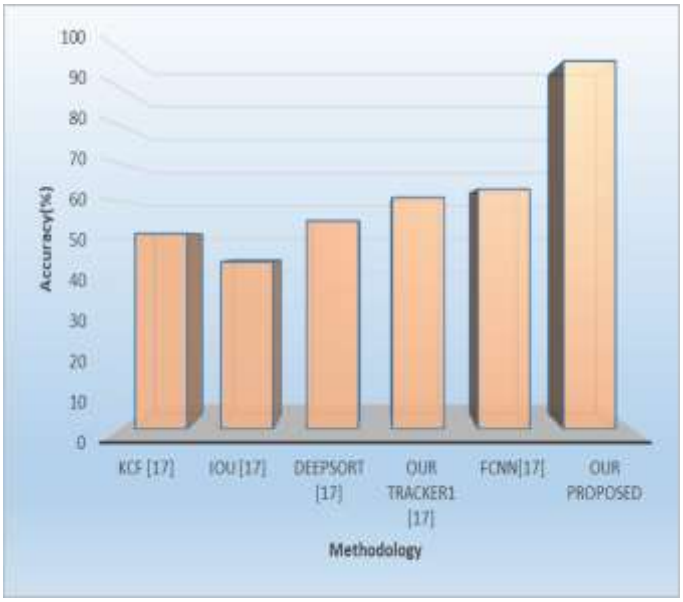


Figure 7 Accuracy percentile calculation

A classifier's ability to accurately identify all expected positives and projected negatives should be multiplied by the total of all real positives and real negatives, and that number should then be divided by the total. The accuracy of the suggested technique, on the other hand, is 96.8%, which is more than the accuracy of the current procedures (see Figure 7).

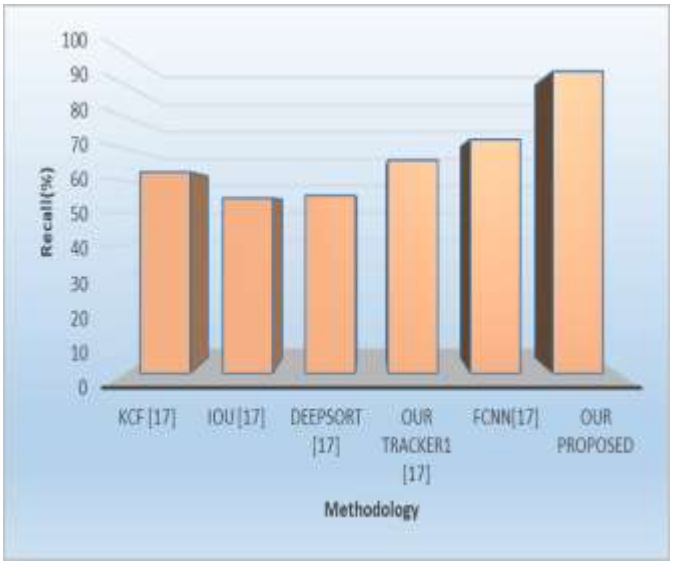


Figure 8 Recall percentile calculation

The proposed MS-YOLOV8 and ((DSA) -SORT) approach has a recall of up to 94 percent, which is exceptionally high compared to that of the current mechanisms, based on the data displayed in Figure 8.

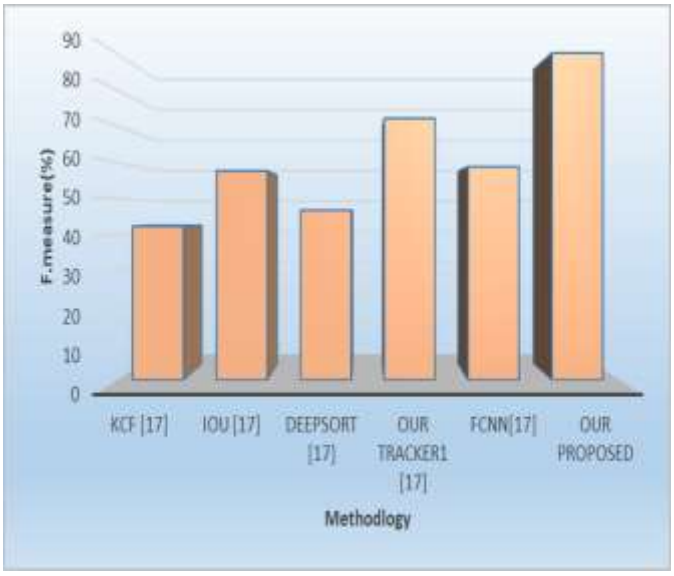


Figure 9 F-measure calculation

The proposed MS-YOLOV8 and ((DSA) -SORT) approach has a high rate of F1 score (90%) compared to the current mechanisms, as can be shown in Figure 13.

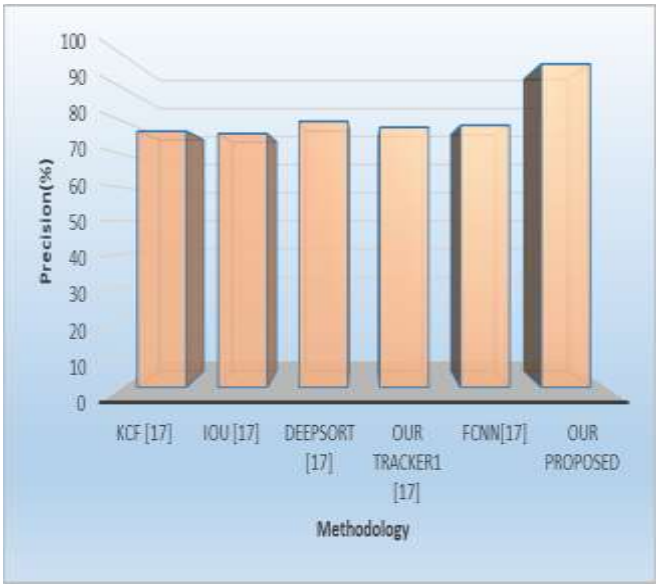


Figure 10 Precision rate calculation

As of from the figure 15 the suggested MS-YOLOV8 and ((DSA) –SORT) technique have the high rate of precision (97%) over object prediction and tracking which is very high that that of the other existing mechanisms in use.

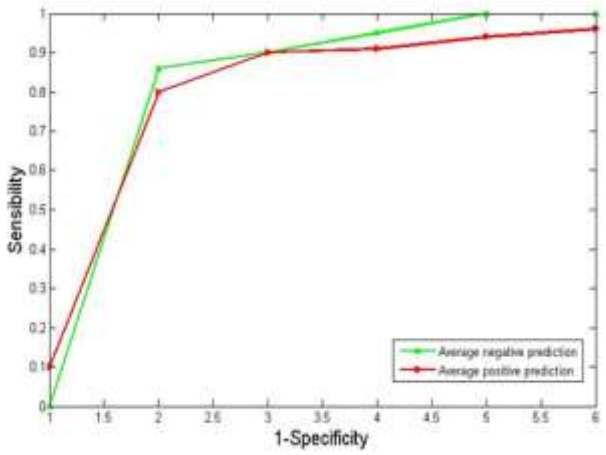


Figure 11 AUC rate calculation

An area under the receiving operating curve (AUC) with a value of 0.90 is shown in Figure 11. The classifier has properly classified the item category if the AUC is 0.9. By looking at the sensitivity and specificity values shown at each location along the AUC, decision

thresholds are identified. The area under the ROC is a measure of a parameter's object detection and tracking performance.

Table 2 Feature extraction comparative analysis

"Featureextraction"	**"Classifier"**	**"False detection rate (FDR)(%)"**	**"False omittion rate (FOR) (%)"**
	"MSVM"	22	18.03
	"KNN"	12	17
"SPT"[5]	"DNN"	9.72	10.74
	"LSTM"	8	3.20
	"Ensemble"	6	1.29
	"MSVM"	11.91	9.5
	"KNN"	6.03	5.93
"WLD"[5]	"DNN"	3	3.07
	"LSTM"	2.09	1.02
	"Ensemble"	1.72	0.86
	"MSVM"	4	3.29
	"KNN"	1.20	1.95
"Hybrid (SPT + WLD)"[5]	"DNN"	0.98	0.83
	"LSTM"	0.79	0.35
	"Ensemble"	0.44	0.32
Proposed DCCA-SAO	"MS-YOLOV8_((DSA) –SORT) (proposed)"	0.2	0.2

Table 3 Classifier with tracker comparative analysis

	"Datast"	"Precision (%)"	"Recall (%)"	"Accuracy (%)"
"GAN-based deep ensemble	"MI	96.4	—	—

technique" [5]	O-TCD"	1		
"Tiny YOLO with SVM" [5]	"BIT Vehicle"	97.90	99.60	—
"Semisupervised CNN model" [5]	"BIT Vehicle"	—	—	88.11
"PCN with softmax classifier [5]	"BIT Vehicle"	—	—	88.52
"TC-SF-CNNLS [5]	"BIT Vehicle"	90.52	90.41	93.80
"proposed"	"Real time"	97	94	96.8

An study of Tables 2 and 3 reveals that the optimisation and deep learning approach provided a mini- mum FDR of 0.2% and a FOR of 0.2%, which are effective in contrast to other classification methods. A large number of security camera images are transformed for usage in testing and the real-time dataset. On the Dataset, the FDR and FOR of the proposed deep learning technique are compared visually. Additionally, the Dataset's proposed ensemble deep learning strategy achieves 96.8% accuracy for each frame. Furthermore, the outcomes of the simulation revealed that the developed model had an AUC of 90%. The developed approach outperforms other presently used techniques in terms of recall, precision, and accuracy throughout the testing period.

VI. CONCLUSION

In this work, it is proposed to use a deep learning-based method, which has been widely utilised in surveillance systems, to video surveillance. In the midst of the COVID-19 epidemic, video surveillance is now being employed for a number of other reasons throughout the world. Our programme uses a deep learning method that may be divided into many discrete

phases, including object identification and tracking. In the context of this work, active feature vectors are extracted using DCCA feature descriptors to reduce training time, improve classification accuracy, and reduce the possibility of overfitting problems. The deep learning technology is utilised in this study to classify different objects based on their attributes. When compared against other categorization techniques, the deep learning approach performed better in terms of precision, recall, accuracy, FDR, and FOR. The Experimental Results and Discussion section made this clear. In comparison to currently used benchmark methodologies, the proposed method showed a classification accuracy improvement of no more than 10%. Future study will apply a segmentation technique based on clustering to improve item type recognition and tracking according to the recommended manner. A competent intelligent transportation system will also pay close attention to three-dimensional modelling, vehicle tracking, and occlusion handling.

REFERENCES

[1] S. A. Najeeb, R. H. Raza, A. Yusuf, and Z. Sultan, "Fine-grained vehicle classification in urban traffic scenes using deep learning," in Proceedings of the 11th International Conference on Robotics, Vision, Signal Processing and Power Applications, 2022, pp. 902-908.

[2] H. Gupta and O. P. Verma, "Monitoring and surveillance of urban road traffic using low altitude drone images: a deep learning approach," Multimedia Tools and Applications, vol. 81, pp. 19683-19703, 2022.

[3] A. Kherraki and R. El Ouazzani, "Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring," IAES International Journal of Artificial Intelligence, vol. 11, p. 110, 2022.

[4] A. Aboah, "A vision-based system for traffic anomaly detection using deep learning and decision trees," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4207-4212.

[5] P. Jagannathan, S. Rajkumar, J. Frnda, P. B. Divakarachari, and P. Subramani, "Moving vehicle detection and classification using gaussian mixture model and ensemble deep learning technique," Wireless Communications and Mobile Computing, vol. 2021, 2021.

[6] W. Liu, Z. Luo, and S. Li, "Improving deep ensemble vehicle classification by using selected adversarial samples," Knowledge-Based Systems, vol. 160, pp. 167-175, 2018.

[7] H. Fu, H. Ma, Y. Liu, and D. Lu, "A vehicle classification system based on hierarchical multi-SVMs in crowded traffic scenes," Neurocomputing, vol. 211, pp. 182-190, 2016.

[8] A. Şentaş, İ. Tashiev, F. Küçükayvaz, S. Kul, S. Eken, A. Sayar, et al., "Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type and color classification," Evolutionary Intelligence, vol. 13, pp. 83-91, 2020.

[9] X. Wang, W. Zhang, X. Wu, L. Xiao, Y. Qian, and Z. Fang, "Real-time vehicle type classification with deep convolutional neural networks," Journal of Real-Time Image Processing, vol. 16, pp. 5-14, 2019.

[10] L. Zhuo, L. Jiang, Z. Zhu, J. Li, J. Zhang, and H. Long, "Vehicle classification for large-scale traffic surveillance videos using convolutional neural networks," Machine Vision and Applications, vol. 28, pp. 793-802, 2017.

[11] V. Murugan and V. Vijaykumar, "Automatic moving vehicle detection and classification based on artificial neural fuzzy inference system," Wireless Personal Communications, vol. 100, pp. 745-766, 2018.

[12] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," IEEE transactions on intelligent transportation systems, vol. 16, pp. 2247-2256, 2015.
[13] M. A. Hedeya, A. H. Eid, and R. F. Abdel-Kader, "A super-learner ensemble of deep networks for vehicle-type classification," IEEE Access, vol. 8, pp. 98266-98280, 2020.
[14] F. C. Soon, H. Y. Khaw, J. H. Chuah, and J. Kanesan, "Semisupervised PCA convolutional network for vehicle type classification," IEEE Transactions on Vehicular Technology, vol. 69, pp. 8267-8277, 2020.
[15] S. Awang, N. M. A. N. Azmi, and M. A. Rahman, "Vehicle type classification using an enhanced sparse-filtered convolutional neural network with layer-skipping strategy," IEEE Access, vol. 8, pp. 14265-14277, 2020.
[16] N. Nasaruddin, K. Muchtar, and A. Afdhal, "A lightweight moving vehicle classification system through attention-based method and deep learning," IEEE Access, vol. 7, pp. 157564-157573, 2019.
[17] N. H. Abdulghafoor and H. N. Abdullah, "A novel real-time multiple objects detection and tracking framework for different challenges," Alexandria Engineering Journal, vol. 61, pp. 9637-9647, 2022.
[18] Y. Xiong, Y. Ge, and P. J. From, "An improved obstacle separation method using deep learning for object detection and tracking in a hybrid visual control loop for fruit picking in clusters," Computers and Electronics in Agriculture, vol. 191, p. 106508, 2021.
[19] X. Lin, C.-T. Li, V. Sanchez, and C. Maple, "On the detection-to-track association for online multi-object tracking," Pattern Recognition Letters, vol. 146, pp. 200-207, 2021.
[20] H. Chen, W. Cai, F. Wu, and Q. Liu, "Vehicle-mounted far-infrared pedestrian detection using multi-object tracking," Infrared Physics & Technology, vol. 115, p. 103697, 2021.
[21] L. Nalpantidis, F. Schöller, M. Blanke, and M. Plenge-Feidenhans, "Vision-based Object Tracking in Marine Environments using Features from Neural Network Detections," 2020.
[22] F. E. T. Schöller, M. Blanke, M. K. Plenge-Feidenhans, and L. Nalpantidis, "Vision-based object tracking in marine environments using features from neural network detections," IFAC-PapersOnLine, vol. 53, pp. 14517-14523, 2020.
[23] M. Elhoseny, "Multi-object detection and tracking (MODT) machine learning model for real-time video surveillance systems," Circuits, Systems, and Signal Processing, vol. 39, pp. 611-630, 2020.
[24] M. Wieczorek, J. Siłka, M. Woźniak, S. Garg, and M. M. Hassan, "Lightweight convolutional neural network model for human face detection in risk situations," IEEE Transactions on Industrial Informatics, vol. 18, pp. 4820-4829, 2021.
[25] I. V. Pustokhina, D. A. Pustokhin, T. Vaiyapuri, D. Gupta, S. Kumar, and K. Shankar, "An automated deep learning based anomaly detection in pedestrian walkways for vulnerable road users safety," Safety science, vol. 142, p. 105356, 2021.