Abdullah Musleh
W21006726
Department of Computer and Information Sciences
KF5042
Intelligent Systems
Submission Date: 10/5/2023
Word Count: 2440

# A Deep Learning Study: Comparing Faster R-CNN, SSD, and YOLOv2 for Human Detection in CCTV Surveillance Systems

Abdullah Musleh (W21006726)
Department of Computer and Information Sciences
Northumbria University
Newcastle Upon Tyne, United Kingdom
abdullah.musleh@northumbria.ac.uk

***Abstract* – In the broad field of computer vision, human detection has emerged as an essential task for ensuring the safety and security of the public. With the advent of deep learning, several object detection models have been developed with the intention of enhancing the accuracy of object detection. This paper presents a comparative study of three renowned deep learning models: Faster R-CNN, SSD, and YOLOv2 for human detection in Closed Circuit Television systems. The study involved the evaluation of the named models based on their Precision-Recall Curve, Average Precision, and Average Miss Rate. The experimental results showed that SSD outperformed the other models in both average precision and average miss rate. All in all, this study provided valuable insight into selecting the appropriate deep learning model for use in human detection in CCTV surveillance systems.**

***Keywords* –** Deep Learning, Object Detection, CCTV, Faster R-CNN, SSD, YOLOv2, Precision-Recall Curve, Average Precision, Average Miss Rate

## I. INTRODUCTION

Parallel to the rapid progression of technology, the implementation of Closed Circuit Television (CCTV) systems has been utilized in many different sectors in an attempt to ensure public safety and security. One crucial constituent of these systems is object detection and more specifically; human detection which is accomplished with the aid of Artificial Intelligence technologies such as Deep Learning.

Object detection is a vital constituent of computer vision which deals with detecting instances of visual objects in digital media [21] and essentially tells us what objects are where. Due to the recent rapid development of deep learning techniques and models, there have been great improvements in object detection, facilitating its use in many real-world utilizations such as video surveillance and autonomous driving [21]. These models utilize a wide range of techniques to be able to identify humans in digital media such as feature extraction and classification [20]. Faster R-CNN, SSD, and YOLOv2 are some of the most popular and renowned deep-learning models used for object detection.

This study presents a comparative analysis of the three models: Faster R-CNN, SSD, and YOLOv2 on their ability to detect people found in CCTV media. The study aims to evaluate their performance in terms of detection accuracy, speed, and computational memory consumption with the aims of gaining insight into selecting the most appropriate deep learning model to implement in surveillance systems. To perform an evaluation of the models, the metrics of precision-recall curves, average precision, and average miss rate will be used.

Section II of this paper provides an in-depth review of the domain of the models: Faster R-CNN [14], SSD [18], and YOLOv2 [9], utilization of these models in the world, and social/ethical concerns. Furthermore, Section III describes the methodology used throughout this study, Section IV covers the experimental results of the study alongside an analysis. Section V presents the conclusion alongside the discussion of approaches toward future improvements in this field.

## II. RELATED WORK

### *A. Object Detection Models*

Faster R-CNN: Faster- R-CNN [14] is an object detection system that is considered an evolved version of its predecessors R-CNN and Fast R-CNN [12], improving on their drawbacks while performing better in object detection tasks. Faster R-CNN is comprised of two modules, the first being a deep fully convolutional network proposing regions by utilizing regional proposal networks (RPN) and the second module is a Fast R-CNN detector which utilizes the proposed regions [14].

To go further into detail, Fast R-CNN models consist of a deep neural network that classifies object proposals within an image and simultaneously refines object proposals. It takes an image input and then object proposals are obtained by using a selective search algorithm. Then, convolutional neural networks (CNN) process each region to create feature maps that are in turn fed into fully connected layers that perform classification and bounding box regression [12]. Furthermore, pooling layers in

Fast R-CNN networks down-sample the feature maps by reducing their dimensionality while retaining the important features.

Region Proposal Networks are used to generate region proposals to be used in object detection. RPNs take an image for an input and outputs sets of rectangular object proposals with an objectness score for each, indicating the likeliness that an object proposal contains an object. To achieve this, RPNs implement a sliding window approach and slide a small fully convolutional network over the feature map and predict the objectness score and the coordinates of the object proposal [14].

In turn, Faster R-CNN uses the RPN to generate region proposals and then feeds them to a Fast R-CNN network. Thus, it can be said that RPN may serve as a preprocessing tool while the Fast R-CNN network performs object detection.

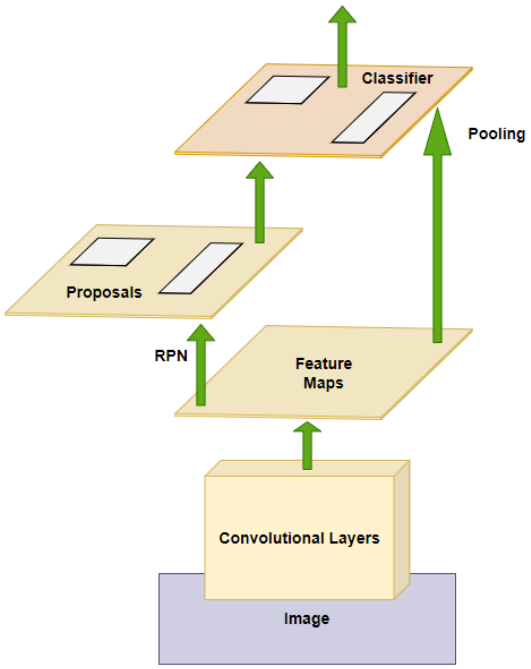


Figure 1: Faster R-CNN architecture showing how Region Proposal Networks and Feature Maps are utilized.

SSD: Single Shot MultiBox Detector is another object detection algorithm that works by using a single deep neural network that is divided into several convolutional layers of decreasing sizes that extract feature maps at multiple resolutions. In turn, every feature map has a set of default anchor boxes defined alongside the objectness score which is calculated by the network. Since these anchor boxes have different aspect ratios and sizes, the network uses feature maps and anchor boxes to predict the object class and refine the bounding box coordinates simultaneously for each detected object [18]. SSD utilizes a base network pre-trained on the ImageNet dataset, typically being VGG-16 or ResNet-50, which extracts feature maps from 300x300 pixel input images.

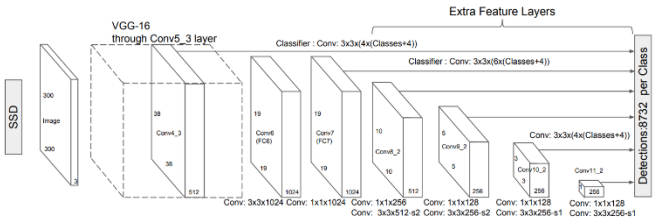


Figure 2: SSD Architecture [7].

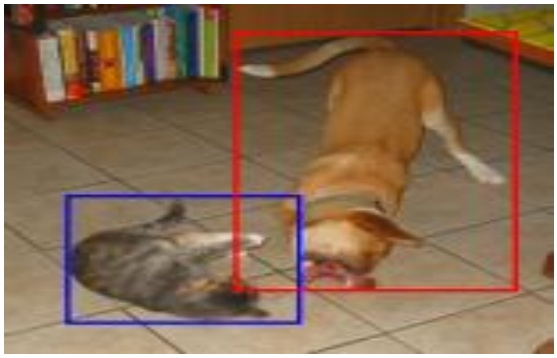

Figure 3: Utilization of different-sized feature maps in SSD [18].

As Figure 3 shows, the cat (blue box) was detected by the 8x8 feature map layer while the dog (red box) was detected by the 4x4 feature map layer [18].

YOLOV2: YOLOv2 is the second version of the You Only Look Once object detection algorithm that improves upon its predecessor YOLO through the addition of several modifications. YOLOv2 works by processing the input image using a single CNN, dividing the image into a grid of cells responsible for predicting bounding boxes and their associated class probabilities. Its architecture is based on Darknet-19, a neural network architecture that contains 19 convolutional layers followed by 5 max-pooling layers and 2 fully connected layers. Furthermore, the network's inputs are images of 416x416 pixels and output a tensor of size 13x13x425 [9].

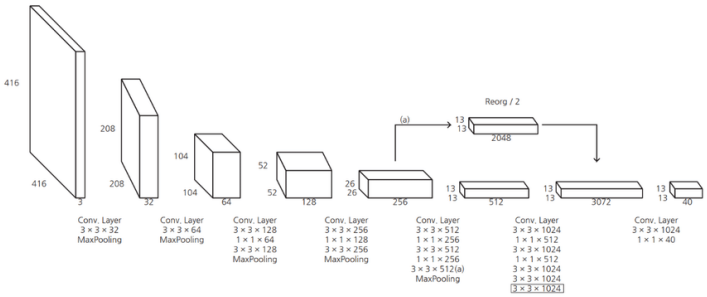


Figure 4: YOLOv2 architecture [7].

Model Comparison: According to experiments conducted by S. Ren et al [14], W. Liu et al. [18], and J. Redmon and A. Farhadi [9] seen in Table 1, in terms of mAP scores; YOLOv2 performs the best on VOC2007 while SSD performs the best on COCO. It is worth noting that the differences in mAP are

relatively small. In terms of speed, SSD performs the best with a speed of 59fps on a GPU while Faster R-CNN is the slowest with a speed of 5fps on a GPU. These results are reinforced by Sojasingarayar [4] in which SSD was the only detector achieving a mAP level above 70% while maintaining good speed (45fps). Also, Faster R-CNN's accuracy level was 73% but at a speed of 7fps [4]. Additionally, in [17] Sowmya & Radha's results show that Faster R-CNN achieves a higher mAP than YOLOv2 with almost a 7% difference but an FPS of 17, a difference of 28fps compared to YOLOv2's 45fps.

| Models | mAP (mean Average Precision) | | Speed (FPS) |
|---|---|---|---|
| | VOC2007 | COCO | |
| Faster R-CNN | 73.2% | 76.4% | 5 |
| YOLOv2 | 78.6% | 73.4% | 40 |
| SSD | 74.3% | 76.8% | 59 |

Table 1: Precision and speed measurements of object detection models on VOC2007 and COCO datasets [3-5].

### *B. Real-World Utilizations*

According to N. Thakur et al [16] deep learning models such as SSD and Faster R-CNN are used to assist UAV-based surveillance systems to aid in real-time surveillance, acquire crime evidence, prevent, and reduce theft, pedestrian detection, etc. Models trained on the MOT20 dataset showed that the highest mAP value of 0.48 was obtained from SSD. Furthermore, A paper by Ingle and Kim [11] showed that YOLO, SSD, and Faster R-CNN models were able to accurately detect normal scenarios such as walking, and abnormal scenarios such as holding a handgun that would be used in surveillance. With mAP levels of 70.1, 69.5, and 80.47 respectively. These findings suggest that the models could be effectively utilized in surveillance systems to enhance their performance.

### *C. Ethical, Social, and Legal Issues*

The use of deep learning models in human detection could raise various ethical, social, and legal issues. One major ethical concern would be privacy invasion where the use of deep learning models would result in the identification and tracking of people, causing possible privacy violations and misuse of personal data [5]. Furthermore, possible bias and discrimination could be fed into models based on training data and in turn, cause inaccuracy and injustice. Additionally, a paper by Saheb [15] showed that there is an intricate relationship between humans, computer authority, humans right breaches, and threats to civil liberties. Finally, there is yet to be any universally applicable regulatory framework for these technologies. These concerns highlight the need for consideration and regulation of the use of deep learning models in human detection.

## III. METHODOLOGY

To gain a clearer insight into how the three models: Faster R-CNN, SSD, and YOLOv2 compared to one another, the code for the models was developed by utilizing the programming language MATLAB. Furthermore, the dataset used to train the detectors [10] was found online. The dataset is comprised of various CCTV images with people in the foreground. But due to computational limitations, a subset of the data representative of the overall dataset that was appropriate for the experiment was used. Furthermore, by utilizing the built-in image labeler in MATLAB, ground truth annotations were obtained in which the regions of interest were manually highlighted and paired with a label. Moving on, datastores were created for the images and labels, and then pre-processing and augmenting the person dataset was done by applying a series of transformations to the images, also the images were standardized to a specific input size through MATLAB's supporting functionality. The preprocessing and augmentation aided in improving the robustness of the models.

As for the hyperparameters of the models, a few tweaks were necessary to collect the results. All three models used the Stochastic Gradient Descend with Momentum optimization algorithm since it allows for more stable convergence and overcome local minima [19]. For Faster R-CNN, due to its large computational requirements, it was continuously hindering the training process until the number of epochs dropped to 3. Additionally, the MiniBatchSize parameter was set to 2 as a larger one would require more computational resources [13]. A common Initial learning rate of $1e^{-3}$ was used to avoid the risk of overshooting the minimum loss function while keeping a good rate of convergence [8]. Moving on to SSD, after a series of trials and errors, 20 epochs achieved the most optimal results, and a similar learning rate of $1e^{-3}$ was used. The mini-batch size of 16 was suitable and chosen as a tradeoff between the computational resources and a faster convergence rate. Finally, YOLOv2's hyperparameters were similar to SSD but with a bigger learning rate of $1e^{-2}$ which is a relatively high learning rate but, enabled a faster convergence rate during training while maintaining stability.

After the completion of training, the testing dataset was used to test the detectors, and MATLAB's 'evaluateDetectionPrecision' function was used to obtain the average precision, recall, and precision, Furthermore, the 'evaluateDetectionMissRate' function was used to obtain the log-average miss rate and false positives. Finally, both the precision-recall curve alongside the log-average miss rate and false positives plots were plotted. This study utilized the average precision and recall generating a precision-recall curve for each model. Additionally, the average miss rate and false positives were used to plot a log-average miss rate vs false positives plot. These chosen metrics provide a comprehensive overview of the models' performances including their ability to detect positive instances alongside false positives. In turn, these metrics will aid in obtaining insight into the performance of Faster R-CNN, SSD, and YOLOv2 in detecting humans in CCTV media.

Despite the successful training of the detectors and the obtainment of successful results, several limitations need to be

considered. To begin with, the computational resources available in the completion of this study were limited which restricted the size of the data set, affected the hyperparameters used and as a result, the models may have not reached their full potential. Additionally, the data set used in this study could have not been fully representative of the diversity of all CCTV footage and may have introduced some bias. Finally, since the annotation process for the ground truth was done manually, there could be discrepancies due to human error although efforts were made to minimize these errors.

## IV. EXPERIMENTAL RESULTS

The experimental results were analyzed using MATLAB's built-in evaluation metrics such as average precision, recall, miss rate, and false positives allowing for an overview of the performance of Faster R-CNN, SSD, and YOLOv2. As seen in Figures 5 and 6, SSD had the highest average precision of 0.64, followed by Faster R-CNN with 0.6 and YOLOv2 with 0.5. Also, the log average miss rate for Faster R-CNN and SSD is 0.6 performing better than YOLOv2's 0.7. The obtained results suggest that SSD performs the best out of all models in terms of average precision, while Faster R-CNN is slightly less precise with YOLOv2 not being up to par. However, with consideration of the log-average miss rate, Faster R-CNN and SSD performed better than YOLOv2.

In terms of the significance of the results, these results can aid in informing the selection of an object detection model in real-world applications. SSD would be the recommended model due to its high average precision and its performance in log-average miss rate with fewer missed detection than YOLOv2.

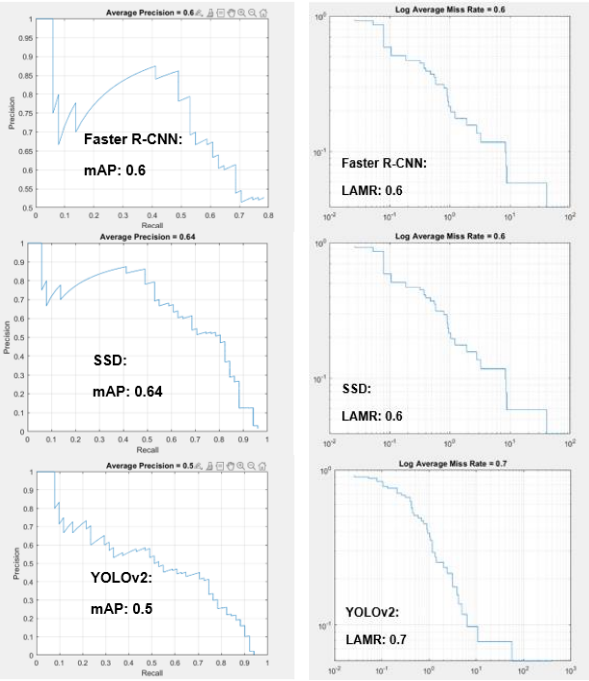


Figure 5: Faster R-CNN, SSD, and YOLOv2's mAP and LAMR.

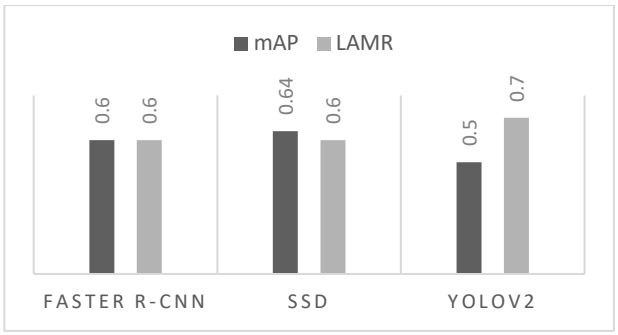


Figure 6: Bar Chart of Faster R-CNN, SSD, and YOLOv2's mAP and LAMR.

## V. CONCLUSION

This paper presents a comparative analysis of three object detection models for human detection in closed-circuit television (CCTV) systems. The three models evaluated are Faster R-CNN, SSD, and YOLOv2. The study aimed to evaluate the models based on their detection accuracy by using metrics such as average precision, average miss rate, and precision-recall curves. The study concluded that SSD outperformed the other models in terms of precision and miss rate. Overall, the study provided useful insight into selecting the most appropriate deep learning model for use in surveillance systems. In the future, further research can be done on improving the performance of the models, increasing their speed, and decreasing the miss rate to aid in real-time surveillance. Furthermore, there are various ethical and social implications that exist in the domain of object detection surveillance and future work needs to be done to address these implications and concerns.

## REFERENCES


[1] "Object Detection Using Faster R-CNN Deep Learning," MATLAB & Simulink, https://www.mathworks.com/help/deeplearning/ug/object-detection-using-faster-r-cnn-deep-learning.html (accessed May 10, 2023).

[2] "Object Detection Using SSD Deep Learning," MATLAB & Simulink - MathWorks United Kingdom, https://uk.mathworks.com/help/vision/ug/object-detection-using-single-shot-detector.html (accessed May 10, 2023).

[3] "Object detection using Yolo V2 Deep Learning," Object Detection Using YOLO v2 Deep Learning - MATLAB & Simulink - MathWorks United Kingdom, https://uk.mathworks.com/help/deeplearning/ug/object-detection-using-yolo-v2.html (accessed May 10, 2023).

[4] A. Sojasingarayar, "Faster R-CNN vs Yolo vs SSD-object detection algorithms," Medium, https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc

[5] B. L., "Ethical concerns of combating crimes with AI surveillance and Facial Recognition Technology," Medium, https://towardsdatascience.com/ethical-concerns-of-combating-crimes-with-artificial-intelligence-surveillance-and-facial-a5eb7a09abb1

[6] Barreiros, Marta & Dantas, Diego & Silva, Luis & Ribeiro, Sidarta & Barros Filho, Allan Kardec. (2021). Zebrafish tracking using YOLOv2 and Kalman filter. Scientific Reports. 11. 3219. 10.1038/s41598-021-81997-9.

[7] Convolutional Neural Network-Based Real-Time Object Detection and Tracking for Parrot AR Drone 2 - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Basic-architecture-of-SSD_fig2_333407340

[8] J. Brownlee, "Understand the impact of learning rate on neural network performance," MachineLearningMastery.com, https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

[9] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv.org, https://arxiv.org/abs/1612.08242

[10] K. Verner, "Human detection dataset," Kaggle, https://www.kaggle.com/datasets/constantinwerner/human-detection-dataset

[11] P. Y. Ingle and Y.-G. Kim, "Real-time abnormal object detection for video surveillance in Smart Cities," *Sensors*, vol. 22, no. 10, p. 3862, 2022. doi:10.3390/s22103862

[12] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[13] R. Rotenberg, "How to break GPU memory boundaries even with large batch sizes," Medium, https://towardsdatascience.com/how-to-break-gpu-memory-boundaries-even-with-large-batch-sizes-7a9c27a400ce

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," arXiv.org, https://arxiv.org/abs/1506.01497

[15] T. Saheb, "'ethically contentious aspects of Artificial Intelligence Surveillance: A social science perspective,'" *AI and Ethics*, 2022. doi:10.1007/s43681-022-00196-y

[16] Thakur, Dr. Narina & Nagrath, Preeti & Jain, Rachna & Saini, Dharmender & Sharma, Nitika & D, Jude. (2021). Object Detection in Deep Surveillance. 10.21203/rs.3.rs-901583/v1.

[17] V. Sowmya and R. Radha, "Comparative analysis on Deep Learning Approaches for heavy-vehicle detection based on data augmentation and transfer-learning techniques," *Journal of Scientific Research*, vol. 13, no. 3, pp. 809–820, 2021. doi:10.3329/jsr.v13i3.52332

[18] W. Liu et al., "SSD: Single shot multibox detector," arXiv.org, https://arxiv.org/abs/1512.02325

[19] Y. Liu, Y. Guo, and W. Yin, An Improved Analysis of Stochastic Gradient Descent with Momentum, https://proceedings.neurips.cc/paper/2020/file/d3f5d4de09ea19461dab00590df91e4f-Paper.pdf

[20] Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object Detection in 20 Years: A Survey," in Proceedings of the IEEE, vol. 111, no. 3, pp. 257-276, March 2023, doi: 10.1109/JPROC.2023.3238524.

[21] Zhao, Zhong-Qiu & Zheng, Peng & Xu, Shou-Tao & Wu, Xindong. (2019). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems. PP. 1-21. 10.1109/TNNLS.2018.2876865.

## Code: (Note that supporting functions need to be placed in the working directory, or in the script)

**Faster R-CNN Model Code** [1]:

```
clc;
clear;

% Setting doTraining variable to true
doTraining = true;

% Loading the custom dataset
GTruthData = load('PERSONS225DS.mat');
source = GTruthData.gTruth.DataSource.Source;
labelData =  GTruthData.gTruth.LabelData.Person;
personDataset = table(source, labelData);

% Display first 6 rows personDataset.
personDataset(1:6,:)

% Setting rng to 0, shuffle the random indices of personDataset into
% training validation and testing sets
rng(0);
shuffledIndices = randperm(height(personDataset));
idx = floor(0.6 * height(personDataset));

% Indicies of training set
trainingIdx = 1:idx;
trainingDataTbl = personDataset(shuffledIndices(trainingIdx),:);

% Getting the indicies of validation set 10%
validationIdx = idx+1 : idx + 1 + floor(0.1 * length(shuffledIndices) );
validationDataTbl = personDataset(shuffledIndices(validationIdx),:);

% Getting indices for test set
testIdx = validationIdx(end)+1 : length(shuffledIndices);
testDataTbl = personDataset(shuffledIndices(testIdx),:);

% Creating image datastore and box label datastore for training set
imdsTrain = imageDatastore(trainingDataTbl{:,'source'});
bldsTrain = boxLabelDatastore(trainingDataTbl(:,'labelData'));

% Creating image datastore and box label datastore for validation set
imdsValidation = imageDatastore(validationDataTbl{:,'source'});
bldsValidation = boxLabelDatastore(validationDataTbl(:,'labelData'));

% Creating image datastore and box label datastore for test set
imdsTest = imageDatastore(testDataTbl{:,'source'});
bldsTest = boxLabelDatastore(testDataTbl(:,'labelData'));

% Combine image/box label datastores to form training validation and test
% datasets
```

```
trainingData = combine(imdsTrain,bldsTrain);
validationData = combine(imdsValidation,bldsValidation);
testData = combine(imdsTest,bldsTest);

% Creating Faster R-CNN network
inputSize = [224 224 3];

% Preprocess training data and estimate the anchor boxes
preprocessedTrainingData = transform(trainingData, @(data)preprocessData(data,inputSize));
numAnchors = 3;
anchorBoxes = estimateAnchorBoxes(preprocessedTrainingData,numAnchors);

% Defining feature extractio network, feature layer and number of classes
featureExtractionNetwork = resnet50;
featureLayer = 'activation_40_relu';
numClasses = width(personDataset)-1;

% Faster R-CNN layer graph
lgraph =
fasterRCNNLayers(inputSize,numClasses,anchorBoxes,featureExtractionNetwork,featureLayer);

% Use augment data function
augmentedTrainingData = transform(trainingData,@augmentData);
augmentedData = cell(4,1);
for k = 1:4
    data = read(augmentedTrainingData);
    augmentedData{k} = insertShape(data{1},'rectangle',data{2});
    reset(augmentedTrainingData);
end
figure
montage(augmentedData,'BorderSize',10)

% Pre processing augmented training and validation data with preprocess
% function
trainingData = transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
validationData = transform(validationData,@(data)preprocessData(data,inputSize));

% Display one preprocessed training image
data = read(trainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,'rectangle',bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)

% Training Faster R-CNN with custom hyperparameters
options = trainingOptions('sgdm',...
    'MaxEpochs',3,...
    'MiniBatchSize',2,...
    'InitialLearnRate',1e-3,...
    'ValidationData',validationData);

if doTraining
    % Train the Faster R-CNN detector.
    [detector, info] = trainFasterRCNNObjectDetector(trainingData,lgraph,options, ...
```

```
        'NegativeOverlapRange',[0 0.3], ...
        'PositiveOverlapRange',[0.6 1]);
else
    % Load the pretrained detector from the example if needed.
    detector = pretrained.detector;
end

% Read one test image and resize to input size
I = imread(testDataTbl.source{5});
I = imresize(I,inputSize(1:2));
[bboxes,scores] = detect(detector,I);

% Insert bounding box with score on image
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)

% Load test data
GTruthData2 = load('PERSONS40DS.mat');
source = GTruthData2.gTruth.DataSource.Source;
labelData =  GTruthData2.gTruth.LabelData.Person;
Dataset2 = table(source, labelData);

% Image datastore
imageDS2 = imageDatastore('PERSONS45DSresized\');
% label datastore
boxLabelDS2 = boxLabelDatastore(Dataset2(:,'labelData'));
% Combine the image and label datastores
testingDs = combine(imageDS2,boxLabelDS2);

% Detect in testing dataset
detectionResults = detect(detector,imageDS2,'MinibatchSize',4);

% Evaluation, get average precision, recall, miss rate, fppi etc
[ap, recall, precision] = evaluateDetectionPrecision(detectionResults,boxLabelDS2);

% Plot precision-recall curve
figure
plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f', ap))

% Plot log-average miss rate vs false positives
[am, fppi, missRate] = evaluateDetectionMissRate(detectionResults, boxLabelDS2);
figure
loglog(fppi, missRate);
grid on
title(sprintf('Log Average Miss Rate = %.1f', am))
```

**SSD Model Code** [2]:

```
clc;
clear;

% Setting doTraining variable to true
doTraining = true;

% Loading the custom dataset
GTruthData = load('PERSONS225DS.mat');
source = GTruthData.gTruth.DataSource.Source;
labelData =  GTruthData.gTruth.LabelData.Person;
personDataset = table(source, labelData);

% Display first 6 rows personDataset.
personDataset(1:6,:)

% Setting rng to 0, shuffle the indices of personDataset and then reorder
rng(0);
shuffledIdx = randperm(height(personDataset));
trainingData = personDataset(shuffledIdx,:);

% Creating image and label datastores
imageDS = imageDatastore(trainingData.source);
boxLabelDS = boxLabelDatastore(trainingData(:,2:end));

% Combining the two datastores
cds = combine(imageDS, boxLabelDS);

% Creating SSD Object Detection Network, defining input size of imgs
inputSize = [300 300 3];

% Defining object classes to detect
classNames = {'Person'};

% Number of object classes to detect
numClasses = width(personDataset)-1;

% Creating SSD network with resnet50 feature extraction
lgraph = ssdLayers(inputSize, numClasses, 'resnet50');

% Data augmentation on the combined datastore
augmentedTrainingData = transform(cds,@augmentData);

% Preprocess the augmented training data to prepare for training
preprocessedTrainingData =
transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));
data = read(preprocessedTrainingData);

% Custom SSD Object Detector Hyperparameters
options = trainingOptions('sgdm', ...
```

```
    'MiniBatchSize', 16, ....
    'InitialLearnRate',1e-3, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropPeriod', 30, ...
    'LearnRateDropFactor', 0.8, ...
    'MaxEpochs', 20, ...
    'VerboseFrequency', 50, ...
    'CheckpointPath', tempdir, ...
    'Shuffle','every-epoch');

if doTraining
    % Train the SSD detector with the custom data.
    [detector, info] = trainSSDObjectDetector(preprocessedTrainingData,lgraph,options);
else
   % Load the pretrained detector from the example if needed.
     detector = pretrained.detector
end

% Read data from combined ds, resize and detect, display results
data = read(cds);
I = data{1,1};
I = imresize(I,inputSize(1:2));
[bboxes,scores] = detect(detector,I, 'Threshold', 0.4);
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
figure
imshow(I)

% Loading test data
GTruthData2 = load('PERSONS40DS.mat');
source = GTruthData2.gTruth.DataSource.Source;
labelData =  GTruthData2.gTruth.LabelData.Person;
Dataset2 = table(source, labelData);

% Image and Label datastores
imageDS2 = imageDatastore(source);
boxLabelDS2 = boxLabelDatastore(Dataset2(:,'labelData'));
% Combine the image and label datastores
testingDs = combine(imageDS2,boxLabelDS2);

% Detect in testing dataset
detectionResults = detect(detector, imageDS2, 'Threshold', 0.01);

% Evaluation, get average precision, recall, miss rate, fppi etc
[ap,recall,precision] = evaluateDetectionPrecision(detectionResults, boxLabelDS2);

% Plot precision-recall curve
figure
plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f',ap))

% Plot log-average miss rate vs false positives
[am, fppi, missRate] = evaluateDetectionMissRate(detectionResults, boxLabelDS2);
figure
```

```
loglog(fppi, missRate);
grid on
title(sprintf('Log Average Miss Rate = %.1f', am))
```

**YOLOv2 Model Code** [3]:

```
clc;
clear;

% Setting doTraining variable to true
doTraining = true;

% Loading the custom dataset
GTruthData = load('PERSONS225DS.mat');
source = GTruthData.gTruth.DataSource.Source;
labelData =  GTruthData.gTruth.LabelData.Person;
personDataset = table(source, labelData);

% Display first 6 rows personDataset.
personDataset(1:6,:)

% Setting rng to 0, shuffle the indices of personDataset and then reorder
rng(0);
shuffledIdx = randperm(height(personDataset));
trainingData = personDataset(shuffledIdx,:);

% Creating image and label datastores
imageDS = imageDatastore(trainingData.source);
boxLabelDS = boxLabelDatastore(trainingData(:,2:end));

% Combining the two datastores
cds = combine(imageDS, boxLabelDS);

% Define input size and number of classes for object detection
inputSize = [224 224 3];
numClasses = width(personDataset)-1;

% Preprocess the data for training, resize and normalize
trainingDataForEstimation = transform(cds,@(data)preprocessData(data,inputSize));

% Estimate anchor boxes for detection
numAnchors = 7;
[anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingDataForEstimation, numAnchors);

% Set feature extraction network and feature layer to use
featureExtractionNetwork = resnet50;
featureLayer = 'activation_40_relu';

% Define YOLOv2 network architecture
lgraph = yolov2Layers(inputSize,numClasses,anchorBoxes,featureExtractionNetwork,featureLayer);

% Augment training data
augmentedTrainingData = transform(cds,@augmentData);

% Visualize the augmented training data
```

```
augmentedData = cell(4,1);
for k = 1:4
    data = read(augmentedTrainingData);
    augmentedData{k} = insertShape(data{1},'rectangle',data{2});
    reset(augmentedTrainingData);
end
figure
montage(augmentedData,'BorderSize',10)

% Preprocess the augemnted training data, resize and normalize
preprocessedTrainingData =
transform(augmentedTrainingData,@(data)preprocessData(data,inputSize));

% Read data, resize and detect, display results
data = read(preprocessedTrainingData);
I = data{1};
bbox = data{2};
annotatedImage = insertShape(I,'rectangle',bbox);
annotatedImage = imresize(annotatedImage,2);
figure
imshow(annotatedImage)

% Custom YOLOv2 Object Detector Hyperparameters
options = trainingOptions('sgdm', ...
        'MiniBatchSize',16, ....
        'InitialLearnRate',1e-2, ...
        'MaxEpochs',20, ...
        'CheckpointPath',tempdir);

if doTraining
    % Train the YOLO v2 detector.
    [detector,info] = trainYOLOv2ObjectDetector(preprocessedTrainingData,lgraph,options);
else
    % Load pretrained detector for the example.
    detector = yolov2ObjectDetector('darknet19-coco');
end


% Load test data
GTruthData2 = load('PERSONS40DS.mat');
source = GTruthData2.gTruth.DataSource.Source;
labelData =  GTruthData2.gTruth.LabelData.Person;
Dataset2 = table(source, labelData);

% Image and Label datastores
imageDS2 = imageDatastore(source);
boxLabelDS2 = boxLabelDatastore(Dataset2(:,'labelData'));
% Combine the image and label datastores
testingDs = combine(imageDS2,boxLabelDS2);

% Detect in testing dataset
detectionResults = detect(detector, imageDS2,Threshold=0.001);

% Evaluation, get average precision, recall, miss rate, fppi etc
[ap,recall,precision] = evaluateDetectionPrecision(detectionResults, boxLabelDS2);
```

```
% Plot precision-recall curve
figure
plot(recall,precision)
xlabel('Recall')
ylabel('Precision')
grid on
title(sprintf('Average Precision = %.2f',ap))

% Plot log-average miss rate vs false positives
[am, fppi, missRate] = evaluateDetectionMissRate(detectionResults, boxLabelDS2);
figure
loglog(fppi, missRate);
grid on
title(sprintf('Log Average Miss Rate = %.1f', am))
```

**Supporting Functions Code [1-3]:**

preprocessData:

```
function data = preprocessData(data,targetSize)
% Resize image and bounding boxes to the targetSize.
sz = size(data{1},[1 2]);
scale = targetSize(1:2)./sz;
data{1} = imresize(data{1},targetSize(1:2));

% Sanitize boxes, if needed. This helper function is attached as a
% supporting file. Open the example in MATLAB to access this function.
%data{2} = helperSanitizeBoxes(data{2});

% Resize boxes to new image size.
data{2} = bboxresize(data{2},scale);
end
```

augmentData:

```
function B = augmentData(A)
% Apply random horizontal flipping, and random X/Y scaling. Boxes that get
% scaled outside the bounds are clipped if the overlap is above 0.25. Also,
% jitter image color.

B = cell(size(A));

I = A{1};
sz = size(I);
if numel(sz)==3 && sz(3) == 3
    I = jitterColorHSV(I,...
        'Contrast',0.2,...
        'Hue',0,...
        'Saturation',0.1,...
        'Brightness',0.2);
end

% Randomly flip and scale image.
tform = randomAffine2d('XReflection',true,'Scale',[1 1.1]);
rout = affineOutputView(sz,tform,'BoundsStyle','CenterOutput');
B{1} = imwarp(I,tform,'OutputView',rout);

% Sanitize boxes, if needed. This helper function is attached as a
% supporting file. Open the example in MATLAB to access this function.
%A{2} = helperSanitizeBoxes(A{2});

% Apply same transform to boxes.
[B{2},indices] = bboxwarp(A{2},tform,rout,'OverlapThreshold',0.25);
B{3} = A{3}(indices);

% Return original data only when all boxes are removed by warping.
if isempty(indices)
    B = A;
end
```

```
end
```

helperSantizeBoxes:

```
%helperSanitizeBoxes Sanitize box data.
% This example helper is used to clean up invalid bounding box data. Boxes
% with values <= 0 are removed.
%
% If none of the boxes are valid, this function passes the data through to
% enable downstream processing to issue proper errors.

% Copyright 2020-2022 The Mathworks, Inc.

function boxes = helperSanitizeBoxes(boxes, ~)
persistent hasInvalidBoxes
valid = all(boxes > 0, 2);
if any(valid)
    if ~all(valid) && isempty(hasInvalidBoxes)
        % Issue one-time warning about removing invalid boxes.
        hasInvalidBoxes = true;
        warning('Removing ground truth bounding box data with values <= 0.')
    end
    boxes = boxes(valid,:);
end
end
```