SEVAS Publishing

Computing Technology Research Journal
Compute. Tech. Res. J. (2022) 1(1):5-11

CATTLE DETECTION AND MISHAP AVOIDANCE SYSTEM USING YOLO v5 ALGORITHM

Vinayak Sutar
Department of Electronics and Telecommunication Engineering, DKTE Society's Textile and Engineering Institute, Rajwada, Ichalkaranji, India

Kshama Kulhalli
Dr. D.Y. Patil College of Engineering & Technology, Kolhapur, India
*Email: kvkulhalli@gmail.com

Received: 10th September 2022, Accepted: 3rd October 2022 and Published: 3rd October 2022

ABSTRACT

Aim: We propose a robust vehicular cattle detection model using YOLO version 5 for vehicles to avoid mishap on Indian roads.
Results: The research was conducted with the help of training and validation on-road cattle image dataset and tested the model for various epoch values.
Conclusion: The model has predicted 82% to 85% of true positive result with 90.5% accuracy and fruitful test results observed on real world video samples.

Keywords: Object detection, Road Safety, CNN, YOLO, Mishap Avoidance

HIGHLIGHTS:

1. **YOLO v5 is used for developing a robust cattle detection model.**
2. **Real world video samples and on-road cattle image datasets showed 90.5% accuracy.**

INTRODUCTION

In our research work experimentation has been carried out on Google colab Jupyter notebook environment with customized object detection model for our cattle dataset. This environment provides Nvidia Tesla K80 GPU with 12GB of memory. More than five hundred image samples consisting of stray cattle found on Indian roads are used for training and validation.

The research work focuses on building a cattle detection model [1] especially for Indian stray animals observed on roadways using single stage object detector YOLO version 5 [2]. The YOLO network architecture mainly divided in three parts model backbone, model neck and model head. The first part model backbone of network mainly utilized to extract the important features available in the input image. The Cross Stage Partial Networks (CSP) [3] at this backbone part extracts rich in informative features from an image. The CSP network improves the model in terms of processing time with deeper network. The next part of single stage detector is model neck which generates feature pyramids by using Path Aggregation Network (PANet)[4]. The generated feature pyramids help the model to generalize well on object scaling. Therefore model could effectively identify the objects with different sizes and scales. Also it helps to perform well on unseen data. The model head at final stage performs the object detection by applying anchor boxes on features. Also it generates resultant output vectors with objectness scores, class probabilities and bounding boxes. The neural network in the model uses Leaky ReLU activation function at middle and hidden layers and at final detection layer sigmoid activation function. To optimize the network model uses Stochastic Gradient Descent (SGD) default optimization function. The compound loss calculation in model is based on class probability score, objectness score and bounding box regression score. Binary Cross Entropy with Logits Loss function from PyTorch for loss calculation of class probability and object score used by Ultralytics.

Methods:

We proposed nine stages procedures to build the model for cattle detection for mishap avoidance system as shown in Figure 1.

Collection of Dataset for Training

Selections of suitable machine learning oriented data have been a crucial step in the analysis of the data. As the model dealing with stray cattle detection on Indian roads, factors such as number of cattle, their classes and dimensions and other object class influence the study directly. As mentioned in previous section, to train the model more than five hundred images of roadways where stray animals present are considered for training and hundreds of samples are utilized for validation. The dataset of stray animals like cows, bulls, calf, buffalos, etc. are collected from website such as kaggle.com, indiatimes.com, indianexpress.com and indiatoday.in.

Annotation of Dataset

In machine learning and object detection models it is necessary to supervise its learning with the help of bounding box annotations. Therefore in this stage all the images including training and validation are annotated. As proposed work focused on cattle detection model, all cattle in the images are annotated and assigned label for cattle class and obtained bounding box (x1, y1) and (x2, y2) coordinates. Figure 2 (a) shows the one of the annotated image during batch processing where 'cattle' over the bounding box represents the class name of object and four annotated bounding box over cattle. In Figure 2(b) the first digit '0' represents the class of object and respective XY coordinates of cattle in the image.

All the annotated images of dataset are categorized into training and validation subdirectory and respective labels are also compiled in training and validation subdirectory. The final directories of training and validation which includes annotated images and respective labels are now ready to build the model. The program is available as supplementary data.

RESULTS

After setting all program options (See Supplementary Data) the file path for our custom dataset, training of model has been carried out for various training epoch values in the range of 10 to 150. Finally the weights are obtained by achieving greater validation mean average precision. After the training of model, performance of model evaluated. Figure 3 shows the results obtained after training of the model. All the cattle in the image samples are predicted and percentage of prediction shown in each diagram.

Figure 4a shows the box loss and objectness loss which represents how well the model can locate the centre of the predicted cattle and how well it can covered by the predicted bounding box. Objectness is the term deals with measure of the probability that the cattle exists in a ROI. High objectivity represents the image window is likely to contain an object. The improvement in the model can be observed in terms of precision, recall (Figure 4b) and mean average precision before flattening after about 100 epochs. All the losses in the validation data show a rapid decline until around 100 epochs. Early stopping was used for selecting best weights.

The results obtained with optimization accuracy of 0.81 at confidence value 0.390. The confidence value indicates the probability that an anchor box contains an object and as per graph its value is 0.8 (Figure 5a). The precision verses confidence plot shown in Figure 5b where precision is 1 at 0.8 confidence score. The precision verses recall plot serves as an evaluation of the performance of model. The prediction of model is better if the precision stays high as the recall increases. Figure 6a shows the good response of precision and recall of our model. Figure 6b depicts the plot for recall and confidence. Accuracy, Sensitivity and F-measure are considered to evaluate the performance of cattle detection.

Accuracy = (Number of TP + Number of TN) / (Number of TP + Number of TN + Number of FP + Number of FN)
Sensitivity = Number of TP / (Number of TP + Number of FN)
F-Measure = 2 ∗ Number of TP / (2 ∗ Number of TP + Number of FN + Number of FP)

The performance of cattle detection model measured with the help of four matrices.

True Positive (TP):	This means model predicted positive when the ground truth is indeed positive,
False Negative (FP):	specifies the wrong positive prediction of model
False Negative (FN):	specifies an actual instance that is not predicted by the model
True Negative (TN):	specifies a negative prediction given that the actual instance is also negative.

The test video samples which consist of cattle on roadway are tested by the trained model. The obtained weights after training are used in inference to test the video samples for cattle detection. The test videos with 25 frames

per second are uploaded one by one on Colab environment and run the inference. The cattle detection shows sufficient true positive detection accuracy up to 82% with 81% of F score. The tests are carried out on Intel(R) Core2 Duo CPU with 2.93 GHz and 2 Gb of RAM. The Table 1 depicts the summary of mean precision, recall and mAP acquired during the trained of model. Table 1 also shows the results of three performance metrics F measure, Sensitivity and Accuracy considered to measure the performance of model.

Our proposed model is efficient one in terms average accuracy of 90.5%, average sensitivity of 81.5% and average F-measure of 89.5%. Here we have trained and tested for various batch and epoch values by considering various poses of cattle and on road conditions, we observed that our model is robust to detect the cattle in occluded situation as well as in diverse scene also. Fig. 14 shows the randomly collected frames from the tested video samples. Following frames clearly shows our model can detect the herd of cattle on road effectively at poor lighting conditions also. One of the frame shows the model has detected the occluded cow with 78% by the auto rickshaw.

CONCLUSION

Here we propose an efficient cattle detection model with 90.5% detection accuracy which can detect the domestic animals usually found on Indian roads. The cattle detection model using single stage object detector YOLO version 5 can be utilized in the vehicles which can autonomously identify and quantify with the help of onboard camera arrangement. The proposed can help in reducing the number of collisions occurring between stray animals and vehicle on roadways. The proposed model can be upgraded with other classes of animals, pedestrian and different classes of vehicles. This upgrade can give a complete solution to prevent animal to vehicle collisions and major injuries and fatalities on highways.

REFERENCES:

1. https://doi.org/10.1109/TMM.2016.2594138
2. https://doi.org/10.1109/INISTA52262.2021.9548440
3. https://doi.org/10.13031/aim.202200120
4. https://doi.org/10.1117/12.2589473

TABLES & FIGURES:

Table 1. Performance of cattle detection model at various batch and epoch values

Iterations	Batch Value	Epoch Value	Precision	Recall	mAP 0.5	TP	TN	FP	FN	Accuracy	Sensitivity	F Measure
1	02	60	0.6415	0.6644	0.6282	0.85	1.0	0.0	0.15	0.92	0.85	0.91
2	02	100	0.8818	0.7002	0.7786	0.77	1.0	0.0	0.23	0.88	0.77	0.87
3	02	150	0.8706	0.7649	0.8276	0.82	1.0	0.0	0.18	0.91	0.82	0.90
4	03	150	0.8430	0.7314	0.7838	0.82	1.0	0.0	0.18	0.91	0.82	0.90

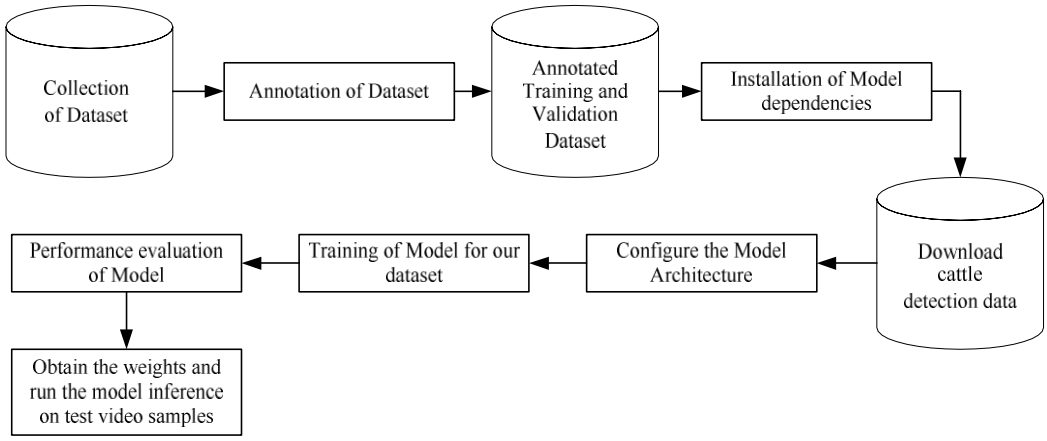
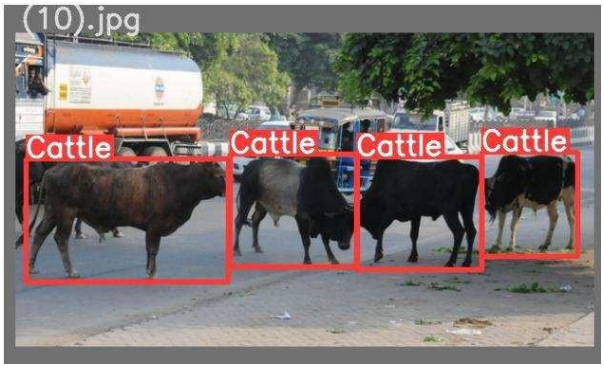


Figure 1: Stages involved in cattle detection model




```
0 0.888643 0.546225 0.166667 0.334454
0 0.698378 0.574776 0.216814 0.358926
0 0.481563 0.566619 0.219764 0.358926
0 0.192478 0.597889 0.349558 0.394275
```

Fig. 2(a) Test batch labeled image and (b) Contents of labeled file

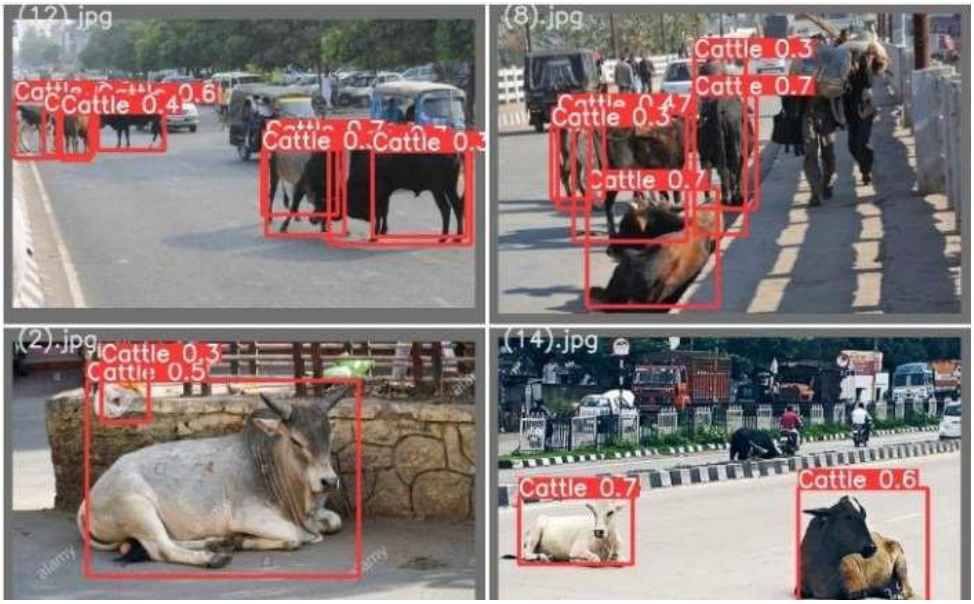


Figure 3: Test batch prediction results

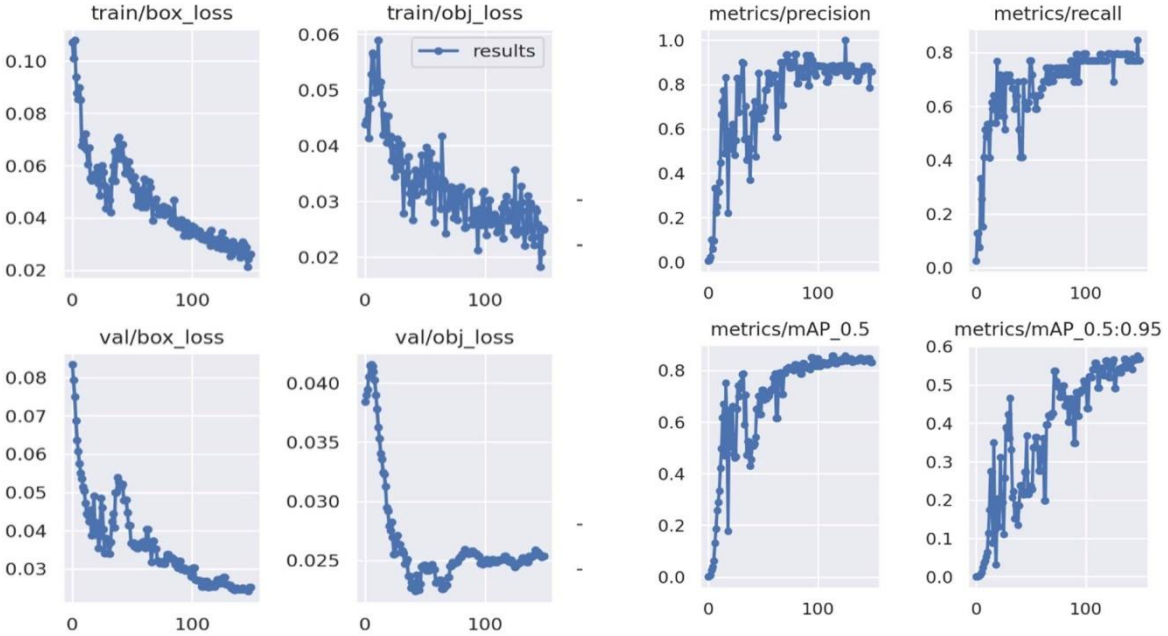


Fig. 4 (a) shows box loss, objectness loss and (b) shows precision recall and mean average precision (mAP) over the training epochs for the training and validation set

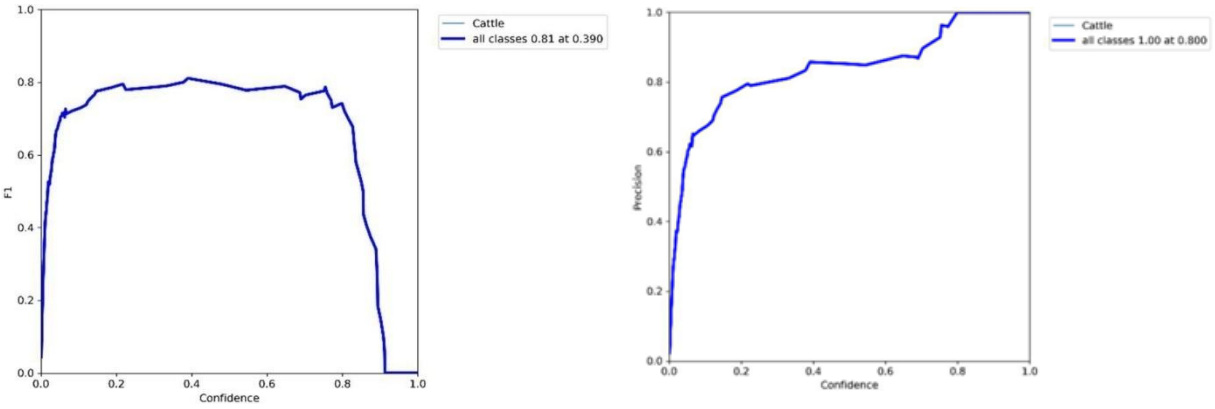


Figure 5 (a) F1 vs Confidence Graph and (b) Precision vs Confidence Graph

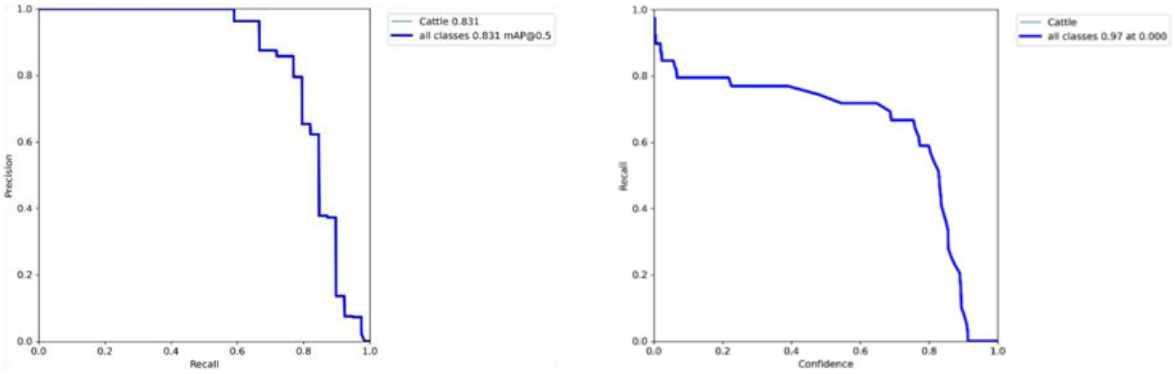


Figure 6(a) Precision vs Recall Graph (b) Recall vs Confidence Graph

SUPPLEMENTARY DATA

Installation of Model Dependencies
In the proposed work the object detection model based is based on YOLO version 5. Therefore in this stage we have cloned the YOLO repository and installed the necessary dependencies. This will sets up the programming environment to build our cattle detection training and inferences command. Following are commands to invoke to clone the repository and installs required dependencies.

```
!git clone https://github.com/ultralytics/yolov5
!pip install -U -r yolov5/requirements.txt
```

After this Google Colab provides training environment for our custom cattle dataset.

```
import torch
from IPython.display import Image # To display images
from utils.google_utils import gdrive_download #
To download models/datasets
print('torch %s %s' % (torch. version , torch.cuda.get_device_properties(0)
if torch.cuda.is_available () else 'CPU'))

YOLOv5 v5.0-288-g8ee9fd1 torch 1.9.0+cu102 CUDA:0 (Tesla K80, 11441.1875MB)
```

Above line shows the details of GPU Tesla K80, received from Google Colab. This cloud based GPU allows to accelerate training times. The Colab provides all necessary preinstalled packages of Torch and Cuda.

Download Cattle Detection Dataset
At this stage the annotated training and validation dataset and their label directories are downloaded in the Colab environment. Now the model must read our custom cattle dataset for training in the environment. Therefore to specify the path of our custom dataset following commands are used in yaml extension file (yet another markup language).

```
# Train and Validation data
train: ../train_data/images/train/ # train images val: ../train_data/images/val/ #
val images test: # test images (optional)

# Classes
```

```
nc: 1 # Defines number of classes names: [ 'Cattle' ] # Defines class name
```

The first line specifies the path of directory where all training samples are present and second line is for validation directory path. Third line is the optional one where testing of images could be done. The number of object classes and their names can be specified under classes. In our work cattle is one of the class to detection, therefore it's '1' and class name is Cattle.

Configure the Model Architecture

YOLO version 5 model comes with various options like v5s, v5m, v5x etc. so in this stage model architecture configured for fastest YOLOv5s using yolov5s.yaml file. This stage could be omitted and directly we can move towards next training stage.

Training of Model

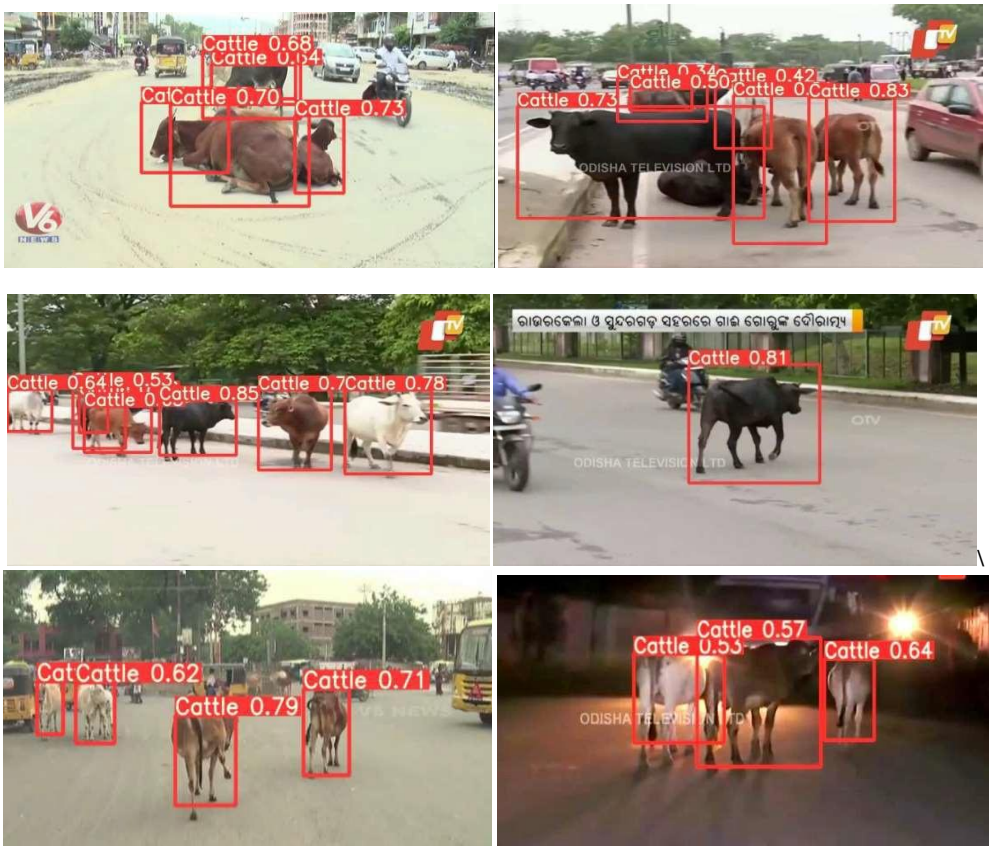
At this stage model is going to be trained for our custom cattle dataset. Before training of model following options are set before train command.

```
!python train.py --img 720 --batch 2 --epochs 150 --data custom_data.yaml --weights
yolov5s.pt –cache

Here,
img: specifies input image size batch: specifies batch size
epochs: specifies the number of training epochs. data: specifies the path to our
yaml file
cfg: specify our model configuration weights: specify a custom path to weights.
cache: specifies cache images for faster training
```

Stray animals on road

Frames in the tested video samples



Frames in the tested video samples