International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 08 | Aug 2022 www.irjet.net p-ISSN: 2395-0072

Real time ship detection using YOLOv5

Dr. R Guru[1], Shrinidhi T S[2], Thryambak M V[3], Shylesh N[4], V Hemanth[5]

[1]Associate Professor, Dept. of Computer Science & Engineering, Sri Jayachamarajaendra College of Engineering, JSSS&TU, Mysuru, Karnataka, India
[2,3,4,5] Dept. of Computer Science & Engineering, Sri Jayachamarajaendra College of Engineering, JSSS&TU, Mysuru, Karnataka, India

---

Abstract - *Ship detection and classification is critical for national maritime security and national defense. As massive optical remote sensing images of high resolution are available, many digital image processing methods have been proposed to detect ships in optical remote sensing images, but most of them face difficulty in terms of accuracy, performance and complexity. In this paper we propose to use YOLOv5 for real time detection of ships.*

Key Words: Ship detection, Machine learning, Deep learning, YOLOv5, Object detection.

1. INTRODUCTION

Automatic detection of ships in a satellite image has gained increased importance in today's military and political scenario. Detecting intrusion of enemy ships in restricted areas, effective enforcement of ceasefire, and protecting the waters of a country are the major military applications of ship detection. Politically, this area is of interest in enforcing agreement of countries with respect to shared water boundaries and monitoring of illegal fishing and maritime activities. Accurate mechanisms for detecting ships in the satellite feed can be game changer in the field. Failing to detect ships and detecting a ship falsely can both cause huge damage in terms of money, infrastructure and international relations. Hence detection of ships has to be carried out with a software that not only detects ships but also does not detect false ones. Many methods have been proposed in the past that include basic histogram processing of image, machine learning and deep learning. We propose to use YOLOv5, an object detection framework which has the capability of detecting ships in milliseconds.

2. LITERATURE REVIEW

In this section we review ship detecting methods for optical images obtained from satellites mainly related to object detection and ship detection.

2.1 Object detection

Object detection is a common task in real life. Finding all objects of interest in an image is called target detection. It has two subtasks: determining the object's categorization and location. Traditional approaches and methods based on convolutional neural networks are the two types of target detection algorithms. Traditional object detection algorithm involves three steps in the traditional object detection algorithm: (1) Using sliding window frames of various sizes and proportions to slide on the input picture with a specific step length and as a candidate area; (2) extracting features from the local information of each candidate region using traditional methods such as colour-based, texture-based, shapebased methods, and some middle-level or high-level semantic features. The algorithm's ultimate output result is the final target that must be discovered. (3) For recognition, use classifiers such as SVM [1] models. Following the evaluation of the check box, a set of candidate boxes that could be the detection target will be generated, with some overlapping conditions. Traditional detection algorithms can produce good results in specific settings, but their performance is difficult to guarantee in complicated environments such as cloudy conditions, unequal ship distribution in an image and different ship sizes. It has a limited ability to generalise. Furthermore, typical manual design elements necessitate a significant amount of prior knowledge. The three-part detection procedure is time-consuming and computationally costly, and it cannot keep up with real-time monitoring.

2.2 Object detection using deep learning

Object detection based on deep learning can be divided into two types. There are two steps in category one. Candidate frames are created first, and then classified. The first is centered on the creation of regions. R-CNN [2], Faster RCNN ([3], [4]), and SPP-net [5] are examples of common approaches. This approach offers a high level of detecting precision, but it takes a long time. The other type employs an end-to-end approach to anticipate the entire image while also detecting and classifying the target position.

SSD ([6], [7]) and YOLO ([8], [9], [10], [11]) are two examples of representative approaches. The end-to-end training is genuinely achieved with the one-stage target detection technique. The regression-based target identification system, represented by YOLO, determines the target category and positions the target all at once. Only convolutional layers and the input picture make up the full network structure. The target category and

position are returned directly after the convolution procedure. As a result, the one-stage target detection method, particularly YOLOv5, which has advanced speed and accuracy, is faster than the two-stage target detection algorithm.

2.3 Related work

Detecting ships in satellite images has been approached by many methods including machine learning and image segmentation. Yifan Li et al. [12] compared the performances of several machine learning methods on binary classification tasks involving images that contained ships and those that did not contain ships, then the results were improved by HOG feature extraction in data preprocessing. Furthermore, they implemented a convolutional neural network (CNN) and compared it with machine learning methods.

Shanlan Nie et al. [13] utilise an instance segmentation model for inshore ship detection. Mask R-CNN is regarded as the baseline model for its high performance in instance segmentation. Meanwhile they introduced Soft-NMS into the Mask R-CNN to improve object detection performance. Therefore, the method is able to obtain the masks of ships, which is useful for predicting the extra information of ships, like area, circumference, direction, etc.

In a paper by Ying Liu et al. [14] ship candidates are coarsely extracted by image segmentation methods first, then actual ships are detected from all the ship candidates and finally classified into 10 different ship classes by deep learning.

A paper by Sagar Karki et al. [15] investigates training methods with different models using the Unet model. The EfficientNet encoder trained Tversky loss on further finetuning on higher resolution can help in tackling false alarms. The smaller and close ships are better segmented by the Googlenet encoder model. Both these models clubbed with a good classifier of whether there is a ship in the image or not further boosts their F2 score. The research in this paper provides a solution that can ease the work of analysing the satellite images and with further fine- tuning, the model can be deployed for maritime surveillance purposes.

After analysing the above works, we decided to propose an object detection framework for ship detection using YOLOv5 so that it is reliable and also fast enough for real time detection.

3. METHODOLOGY

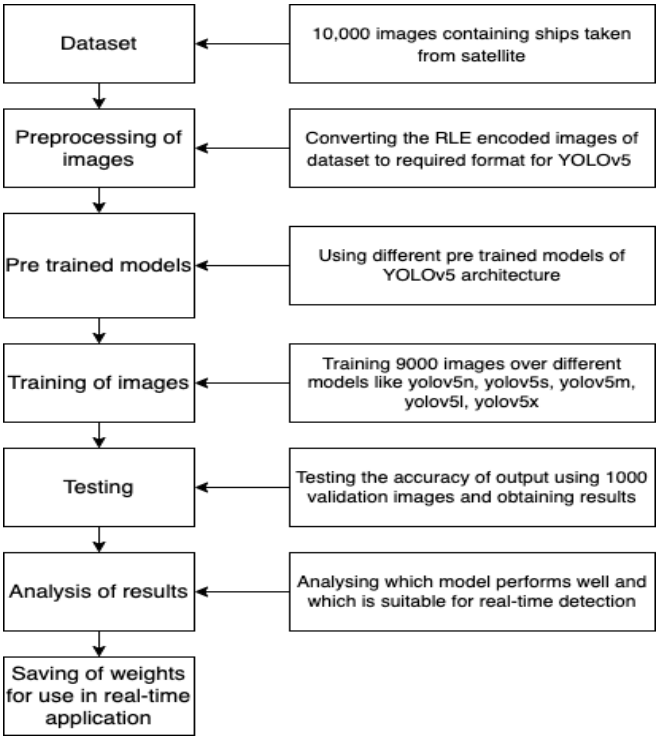


Fig -1: Methodology workflow

We propose a real-time ship detection application that makes use of the YOLOv5 object detection model. Before applying the YOLOv5 model, we also analysed how our dataset performs when trained on image segmentation models like MaskRCNN and UNet.

As shown in fig -1, the preprocessing step involves processing the image so that it can be fed as input for the training process. The images in the dataset come along with RLE encoded strings that indicate pixels where ships are present in an image. This can be used for training by converting RLE encoded string to binary masks for input for training on image segmentation models and by converting them to YOLO annotation format in a text file for training on YOLOv5 object detection model.

YOLOv5 expects annotations for each image in the form of a .txt file where each line of the text file describes a bounding box. Each line represents one of the objects to be detected. The specification for each line is as follows. (1) One row per object. (2) Each row is class x center y center width height format. (3) Box coordinates must be normalized by the dimensions of the image (i.e., have values between 0 and 1). (4) Class numbers are zero-indexed.

3.1 YOLOv5 network

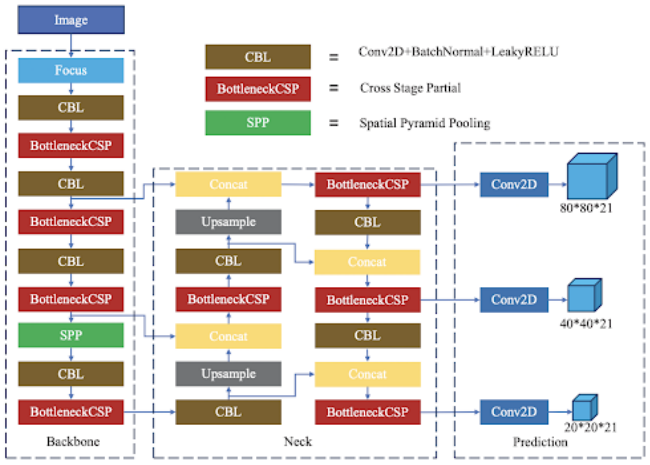


Fig -2: YOLOv5 network structure [16]

The network structure of YOLOv5 is divided into three components, as depicted in fig -2, backbone, neck, and output. The input image with a resolution of 640*640*3 passes via the Focus structure in the backbone. It first becomes a 320*320*12 feature map via the slicing process, and then a 320*320*32 feature map after a convolution operation with 32 convolution kernels. A basic convolution module is the CBL module. Conv2D + BatchNormal + LeakyRELU is represented by a CBL module. The BottleneckCSP module extracts rich information from the image by performing feature extraction on the feature map. The BottleneckCSP structure can reduce gradient information duplication in convolutional neural networks' optimization process when compared to other large-scale convolutional neural networks, in the optimization process of convolutional neural networks, the BottleneckCSP structure can reduce gradient information duplication. Its parameter amount accounts for the majority of the network's parameter quantity. Four variants with varying parameters can be obtained by altering the width and depth of the BottleneckCSP module, namely YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The SPP module primarily improves the network's receptive field and acquires features of various scales.

In addition, based on the FPN structure, YOLOv5 adds a bottom-up feature pyramid structure. The FPN layer sends powerful semantic characteristics from top to bottom, whereas the feature pyramid conveys robust positional features from the bottom up, thanks to this combination operation. To boost the network's ability to detect targets at different scales, use feature aggregation from several feature layers. Output the categorization results and object coordinates at the bottom of the figure. We used the opensource version of YOLOv5 by ultralytics[17]. This contains different models that have been pre trained on the MS COCO dataset for 300 epochs. The following is a short description of each of these:

- YOLOv5n: This is the new nano model, smallest in the family and meant for the edge, IoT devices, and with OpenCV DNN support as well. It takes less than 2.5MB in INT8 format. It is ideal for mobile solutions.
- YOLOv5s: This is the small-sized model in the family. It has around 7.2 million parameters and is best suited for CPU inference.
- YOLOv5m: This is the medium-sized model in the family, it has 21.2 million parameters. It is an ideal model for many datasets and training since it provides a good tradeoff between speed and accuracy.
- YOLOv5l: It is the large model in the YOLOv5 family, it has 46.5 million parameters. It is ideal for datasets where target objects are small in size.
- YOLOv5x: It is the largest and has the highest mAP among the five models. It is slower compared to the others in terms of inference time and has 86.7 million parameters.

3.2 Dataset

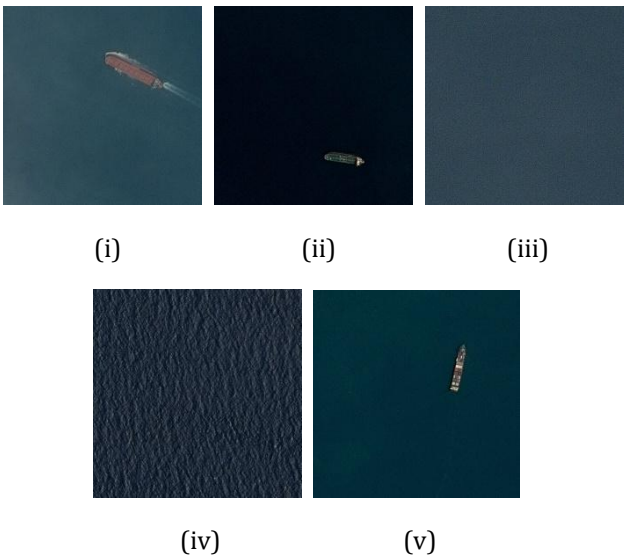


Fig -3: Sample images from dataset [18], i, ii & v contain ships and iii & iv do not

The dataset is taken from Kaggle [18], which comprises 1,93,000 pictures, with just 40,000 images containing 80,000 ships [fig -3]. There may be one or more ships in the photos with ships. To generate masks for the image, the corresponding run-length encoded data is provided in a separate CSV file. The data contains 15,000 photos for testing, and the ground truth values are available on

Kaggle. The models that perform well on the test data show their robustness and provide an approximate approximation of how well they will work when deployed. All of the images are satellite images with a resolution of 768*768 pixels. Due to computational resource limits, all training is done using 10,000 images from the original dataset that contain varying number of ships from 0 to 16 and scaling down the size to 416*416 pixels. We considered 9000 images for training and 1000 images for validation and used 600 images from the original 15,000 test images for testing. The text annotations for all the 10,000 images along with the images make our dataset which is further trained on YOLOv5 network.

3.3 Modification of classifier

The models provided by [17] are pre trained on the COCO dataset [19] for 300 epochs each. In [19] there are 80 object categories, and the output tensor's dimension is 3(5 + 80) = 255, where 3 represents each grid prediction's three template boxes. The coordinates (x, y, w, h) and confidence of each prediction box are represented by the number 5. (confidence, c). Because there are two sorts of items in the ship detection scenario, one with ships and one without, the YOLOv5 classifier must be modified. 3(5+2) = 21 becomes the output dimension. Adapting to ship detection, we can reduce the amount of network parameters, lower computing overhead, and increase detection accuracy and speed by altering them. The number of classes is modified to 1 from 80 to reflect our training process [16].

3.4 Metrics used for evaluation

1) Precision - It is a measure of how accurate the predictions are. It is the percentage of correct predictions out of all predictions.

$$Precision = \frac{TP}{(TP + FP)}$$

TP is the number of true positives and FP is the number of false positives.

2) Recall - Recall is the number of positives predicted divided by the total number of existing positives in the data.

$$Recall = \frac{TP}{(TP + FN)}$$

TP is the number of true positives and FN is the number of false negatives.

3) mAP (mean Average Precision) compares the groundtruth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

mAP@ 0.5: It is the average of AP of all pictures in each category when IoU is set to 0.5.

mAP@ 0.5:0.95: This is the average of mAP considering different IoU thresholds (from 0.5 to 0.95 in steps of 0.05)

4) IoU (Intersection over Union) - It is a measure of the overlap between two boundaries. It is used to measure how much the predicted boundary overlaps with the ground truth. For example how much of the picture the predicted bounding box covers.

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}$$

5) AP (Average precision) - It is a metric used to measure the accuracy of object detection algorithms. It is the average precision value for recall value over the range from 0 to 1.

$$AP = \int_0^1 p(r)\, dr$$

$$AP = \frac{1}{11} \times \left(AP_r(0) + AP_r(0.1) + \cdots + AP_r(1.0)\right)$$

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} AP_r$$

Where

$$p_{interp}(r) = \tilde{r} \geq rmax\ p(\tilde{r})$$

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Implementation

In this section, we introduce the experiment's details, and then we show the results of training from scratch and that of using pretrained weights and compare the five models of YOLOv5. Finally, we show the real time performance of each model and analyse the results.

In our experiment, all the training is done by scaling image to 416*416 pixels size. Data enhancement methods such as random flip, geometric distortion, illumination distortion, image occlusion, random erase, cutout, mixup, etc.,. are performed on them. Due to unavailability of a powerful machine that meets the needs of our training, we performed all our analysis on a virtual machine with GPU provided by Google Colab [20] with a batch size of 64. The specifications of the instance are shown in table -1. For training our dataset on YOLOv5 models, the hyperparameters as shown in table -2 were used.

Table -1: Experimental environment configuration.

Parameter	Configuration
CPU	2 Intel(R) Xeon(R) CPU @ 2.00GHz
GPU	Tesla P100-PCIE-16GB
RAM	25GB
Language	Python 3.6
Acceleration environment	CUDA 11.2

Table -2: Some hyperparameters used for training

Parameter	Value
Learning rate	0.01
Learning rate decay	0.999
Weight decay rate	5e-4
Momentum	0.937
Batch size	64

Initially, the different YOLOv5 models i.e yolov5n, yolov5s, yolov5m, yolov5l, yolov5x which vary in network size and model size (in MB) with yolov5n having least network size and model size and yolov5x having the largest were trained on the dataset without using any pretrained weights. The results obtained after training from scratch for 100 epochs batch size of 64, image size of 416*416 pixels and hyperparameters as shown in table -2 are shown in fig -4 and table -3 shows the best mAP/0.5 obtained on training each model.

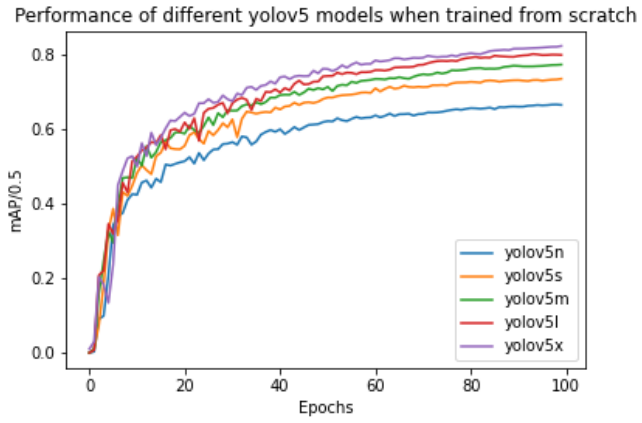


Fig -4: This figure shows how different models perform over 100 epochs when trained from scratch without using pretrained weights.

Table -3: Best mAP/0.5 achieved by training models from scratch without using pretrained weights.

Model	Best mAP/0.5 after 100 epochs
yolov5n	0.66436
yolov5s	0.73315
yolov5m	0.77113
yolov5l	0.79944
yolov5x	0.82100

Later, YOLOv5 models that were pretrained on the COCO dataset[19] were used for training. We used yolov5n, yolov5s, yolov5m, yolov5l, yolov5x along with their pretrained weights and trained them for our dataset by modify- ing the configuration file for 100 epochs with other configurations like batch size, hyperparameters, image input size same as that of training from scratch. The results obtained were as expected i.e. they performed way better than the models that were trained from scratch and yolov5x performed the best when compared to other models owing to its large network size. table -4 shows the best mAP/0.5 achieved on each model. The highest mAP/0.5 achieved was 0.89 using yolov5x. fig -5 shows how each model performed on each epoch.

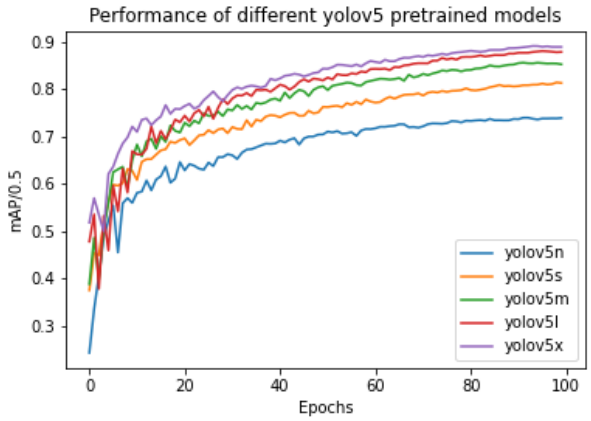


Fig -5: This figure shows how different models perform over 100 epochs when trained using pretrained weights

Table -4: Best mAP/0.5 achieved during transfer learning using each pretrained model.

Model	Best mAP/0.5 after 100 epochs
yolov5n	0. 73989
yolov5s	0. 81424
yolov5m	0. 85593

yolov5l	0. 88049
yolov5x	0. 89119

As the model size of yolov5x is very large and requires more computational resources to train for further epochs after 100, we trained yolov5l for more than 100 epochs by training the model for 100 epoch once a time and then using the saved weight of last epoch and training further for 100 more epoch. Even though after each 100 epochs the mAP reduces during training due to usage of saved weights of previous 100 epochs and different learning rate, eventually the model converges to one value of mAP/0.5 after around 400 epochs. The mAP/0.5 best achieved after 400 epochs was 0.894. As a balancing act, we consider both yolov5s and yolov5l models for our real time detection of ships in satellite images as yolov5l produced higher accuracy and yolov5s produced satisfactory performance with high speed detection. table -5 lists the time taken by each model for inference.

Table -5: Inference time per image in milliseconds of different models when trained using pretrained weights.

Model	Inference time per image in milliseconds
yolov5n	8.4
yolov5s	7.2
yolov5m	10.2
yolov5l	14.6
yolov5x	21.1

4.2 Result analysis

The experiment clearly shows that there is around 10% improvement in the performance after pretrained weights were used. We can easily infer from table -5 that yolov5s is very fast and best suited for real time detection. It is also observable that yolov5l has a good trade off between speed and accuracy making it a good choice for scenarios where both accuracy and speed are important. Fig -6 shows some examples of detection of ships using our proposed method. This method detects even small ships in an image and also almost all ships in the image. As the dataset contains more images of ships that are oriented in some angle and less images in which ships are oriented horizontally or vertically, the results for these ships are not that good as the bounding box does not cover the ship completely as in the case of ships that are oriented in an angle different than 90 degree or 180 degree. This is evident from fig -7.

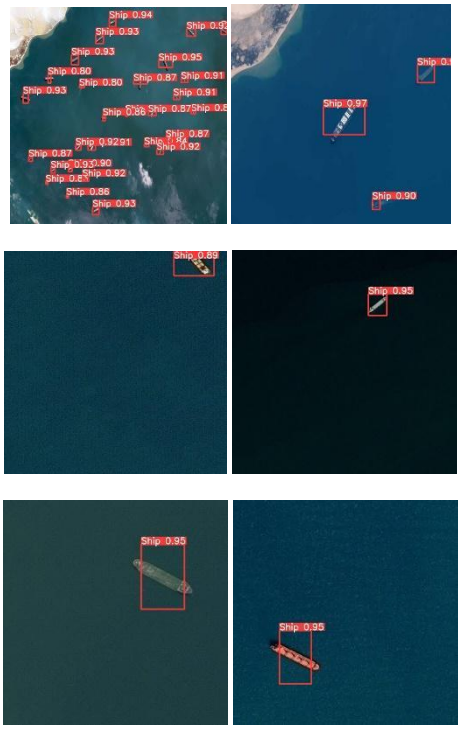


Fig -6: Samples of detection using the proposed method.

Fig -7: Results for images that contain ships oriented horizontally and vertically.

5. CONCLUSION AND FUTURE WORK

Owing to the need of detecting ships accurately and with high speed the proposed method seems a very fit candidate for large scale application for real time ship detection. We obtained a high detection mAP/0.5 of 0.89114 with an inference time of 14.6 milliseconds which can be used in maritime regulatory authorities and defence sector usage. The future work includes adding more images that contain ships oriented in a horizontal or vertical fashion, improving the accuracy using higher batch size, training model from scratch without transfer learning and trying different sets of hyperparameters for a large number of epochs. These things require more computational resources and time. Nevertheless it can improve the accuracy of detecting ships.

REFERENCES

[1] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplarSVMs for object detection and beyond," in 2011 International Conference on Computer Vision, Barcelona, Spain, Nov. 2011, pp. 89–96. doi: 10.1109/ICCV.2011.6126229.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 2014, pp. 580–587. doi: 10.1109/CVPR.2014.81.

[3] C. Han, G. Gao, and Y. Zhang, "Real-time small traffic sign detection with revised faster-RCNN," Multimed. Tools Appl., vol. 78, May 2019, doi: 10.1007/s11042-018-6428-0.

[4] Z. Liu, Y. Lyu, L. Wang, and Z. Han, "Detection Approach Based on an Improved Faster RCNN for Brace Sleeve Screws in High-Speed Railways," IEEE Trans. Instrum. Meas., vol. PP, p. 4395, Jan. 2020, doi: 10.1109/TIM.2019.2941292.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," vol. 8691, 2014, pp. 346–361. doi: 10.1007/978-3-319-10578-9_23.

[6] Z. Chen, K. Wu, Y. Li, M. Wang, and W. Li, "SSD-MSN: An Improved Multi-Scale Object Detection Network Based on SSD," IEEE Access, vol. 7, pp. 80622–80632, 2019, doi: 10.1109/ACCESS.2019.2923016.

[7] X. Hu, H. Li, X. Li, and C. Wang, "MobileNet-SSD MicroScope using adaptive error correction algorithm: real-time detection of license plates on mobile devices," IET Intell. Transp. Syst., vol. 14, no. 2, pp. 110–118, 2020, doi: 10.1049/iet-its.2019.0380.

[8] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection," Inf. Sci., vol. 522, pp. 241–258, Jun. 2020, doi: 10.1016/j.ins.2020.02.067.

[9] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection," IEEE Trans. Very Large Scale Integr. VLSI Syst., vol. 27, no. 8, pp. 1861–1873, Aug. 2019, doi: 10.1109/TVLSI.2019.2905242.

[10] D. Sadykova, D. Mussina, M. Bagheri, and A. James, "IN-YOLO: Real-time Detection of Outdoor High Voltage Insulators using UAV Imaging," IEEE Trans. Power Deliv., vol. PP, pp. 1–1, Sep. 2019, doi: 10.1109/TPWRD.2019.2944741.

[11] Y. Yin, H. Li, and W. Fu, "Faster-YOLO: An accurate and faster object detection method," Digit. Signal Process., vol. 102, p. 102756, Jul. 2020, doi: 10.1016/j.dsp.2020.102756.

[12] Y. Li, H. Zhang, Q. Guo, and X. Li, "Machine Learning Methods for Ship Detection in Satellite Images," p. 6.

[13] S. Nie, Z. Jiang, H. Zhang, B. Cai, and Y. Yao, "Inshore Ship Detection Based on Mask R-CNN," in IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, Jul. 2018, pp. 693–696. doi: 10.1109/IGARSS.2018.8519123.

[14] Y. Liu, H.-Y. Cui, Z. Kuang, and G. Li, "Ship Detection and Classification on Optical Remote Sensing Images Using Deep Learning," ITM Web Conf., vol. 12, p. 05012, Jan. 2017, doi: 10.1051/itmconf/20171205012.

[15] S. Karki and S. Kulkarni, "Ship Detection and Segmentation using Unet," in 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Feb. 2021, pp. 1-7. doi: 10.1109/ICAECT49130.2021.9392463.

[16] F. Zhou, H. Zhao, and Z. Nie, "Safety Helmet Detection Based on YOLOv5," 2021 IEEE Int. Conf. Power Electron. Comput. Appl. ICPECA, 2021, doi: 10.1109/ICPECA51329.2021.9362711.

[17] ultralytics/yolov5. Ultralytics, 2022. Accessed: Jun. 13, 2022. [Online]. Available: https://github.com/ultralytics/yolov5

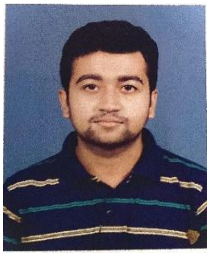
[18] "Airbus Ship Detection Challenge." https://kaggle.com/competitions/airbus-ship-detection (accessed Jun. 13, 2022).

[19] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in Computer Vision – ECCV 2014, Cham, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

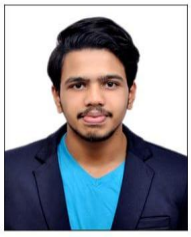
[20] "Making the most of your colab subscription - Colaboratory." https://colab.research.google.com/?utm_source=scs-index (accessed Jun. 13, 2022).

BIOGRAPHIES

Dr. R Guru Dr. R Guru is working as Assistant Professor in the Department of Computer Science and Engineering. He graduated from Bangalore University. He obtained his Masters Degree from B M S College of Engineering, Bangalore, affiliated to Visvesvaraya Technological University. He obtained Doctoral Degree from Visvesvaraya Technological University, Belagavi. His areas of interest are Computer Network, wireless sensor networks, Cloud Computing, IoT.



Shrinidhi T S, is studying B.E in computer science at JSS Science and Technology University. His current research interests include deep learning and computational complexity.



Thryambak M V is studying B.E in computer science at JSS Science and Technology University. His current research interest is deep learning.

Shylesh N is studying B.E in computer science at JSS Science and Technology University. His current research interest is machine learning.

V Hemanth is studying B.E in computer science at JSS Science and Technology University. His current research interest is machine learning.