Object Detector for Visually Impaired with Distance Calculation for Humans

Eliganti Ramalakshmi, Dixitha Kasturi, Gouthami V

Abstract: Object detection is a computer vision technique for locating instances of objects in videos. When we as humans look at images or videos, we can recognize and locate objects within a matter of moments. The main goal of this project is to clone the intelligence of humans in doing that using Deep Neural Networks and IOT, Raspberry Pi and a camera. This model could be used for visually disabled people for improved navigation and crash free motion. When we consider real time scenarios, numerous objects come into a single frame. To identify different items simultaneously as they are captured, a strong model needs to be developed. YOLO (You Only Look Once) is a clever convolutional neural network (CNN) that helps in reaching that objective. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. The bounding boxes are nothing but weighted by the predicted probabilities. The second objective of this model is to calculate distance of humans from the camera, to achieve that haar classifier is created and used. This classifier also helps in enhancing human detection along with distance calculation. Haar is just like a kernel in CNN where the kernel values are determined by training while in Haar they are determined manually. Whenever a person is detected by both YOLO and Haar classifier, a formula which considers height and width of human contours is applied to calculate the distance of it from the camera. As the objects are identified they will be read out using a text-to-speech engine known as gTTS(google text-to-speech) and ,which stores the text in an mp3 file. The package known as Pygame will load and play the mp3 file dynamically as the objects are detected. This developed Deep Learning model is integrated with Raspberry Pi using OpenCV. Though this project is primarily developed to aid visually disabled people, it can have various other applications such as, self-driving cars, video surveillance, pedestrian detection, face detection.

Keywords* : *Deep learning, IOT,YOLO.

I. INTRODUCTION

With the world moving towards automation, there's a rising demand for an efficient Obstacle (Object) detector [1], [2], [7]. There are no low cost object detectors available which are efficient and most of the available obstacle detectors use ultrasonic sensor which doesn't take a broader frame into its line of sight. This project focuses on integrating Deep learning algorithms with Raspberry Pi and Camera to detect the type of object and its distance efficiently.

Revised Manuscript Received on April 25, 2020.
* Correspondence Author

Eliganti Ramalakshmi*, Asst. Professor, Department of Information Technology, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, India

Dixitha Kasturi, Bachelor of Engineering, Information Technology, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, India

Gouthami V, Bachelor of Engineering,Information Technology, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, India

The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detect obstacles [9]. With the availability of huge amounts of data, faster GPUs, and better algorithms, computers can now be trained to detect and classify multiple objects within an image with high accuracy [10].In this project we will explore Object detection and localization using convolutional networks and integrate it with Raspberry Pi and camera. A raspberry pi board with camera and speaker will be used to detect the obstacle and alert the user by reading out the distance and type of obstacle/object in the path of motion. Machine Learning algorithm will be used to train the system. The underlying algorithm is CNN(Convolutional Neural Network) [16]. The model will be trained to identify the type of object and calculate the approximate, if not exact, distance of the humans from the camera. Once the object is detected, the type of object and the distance only for humans will be read out using a speaker

II. DATASET

COCO data set is considered for object detection. It is a large-scale object detection, segmentation, and captioning dataset. It contains images and bounding boxes for the objects in the images. Compared to the previous version it uses different train/val/test splits . In general 91 classes are defined by COCO by only 80 are used.

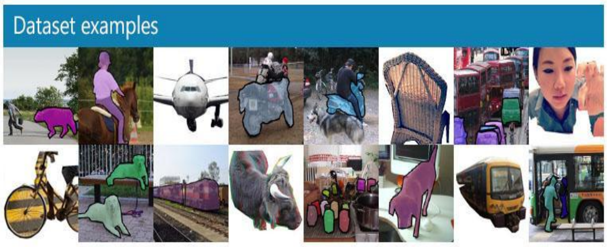

Fig 1. Dataset Representation [21]

III. METHODOLOGY

As shown in Fig 2 the phases of the proposed system design are described as follows :

Phase 1*:* The raspberry pi camera captures the input video live to detect objects in real time.

Phase 2*:* The video captured is broken down into frames and the model trained extracts specific features from the frames.

Phase 3: The boundary boxes are drawn and object proposals are generated [1].

Phase 4: Object proposals from phase 3 and the surrounding proposals from the vicinity are grouped together. Each object proposal grouped together detects a particular part of the object. These object proposals are refined to get the actual object that needs to be detected. Phase 5: The refined objects from phase 4 are classified according to the trained categories. The number of categories that the model can identify us 80 [13], [14], [15]. Phase 6: Based on the contours of the humans detected, the distance between the detector and human is calculated [4]. A Haar classifier is used to identify the contours [11]

Phase 7: The type of object along with the distance for humans is read out using the speaker attached to the Raspberry Pi

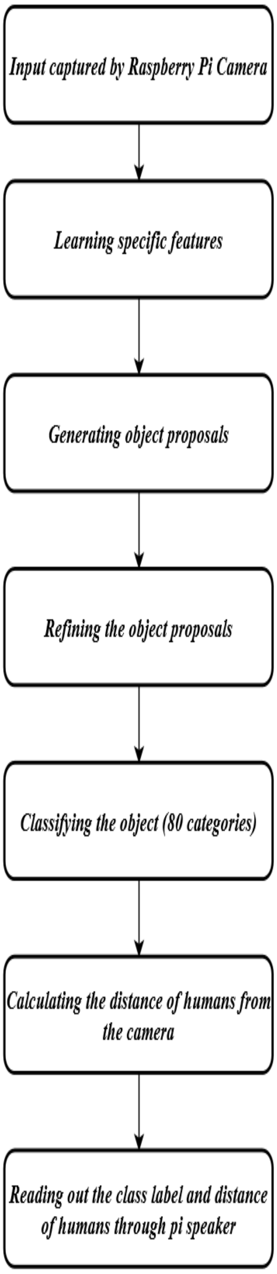
Input captured by Raspberry Pi Camera
Learning specific features
Generating object proposals
Refining the object proposals
Classifying the object (80 categories)
Calculating the distance of humans from the camera
Reading out the class label and distance of humans through pi speaker

Fig 2. Design

A. Object detection Algorithm(YOLO)

The underlying neural network for YOLO is a Convolutional Neural Network which predicts multiple bounding boxes and class probabilities for those boxes simultaneously. YOLO trains on full images and directly optimizes detection performance [5], [6]. It has many benefits over traditional method for object detection, three of which are mentioned below :

YOLO is extremely fast. To predict detections at test time, the neural network is run on a new image. The network runs at 45 frames per second with no batch processing on a Titan X GPU. This means the streaming video in real-time can be processed with less than 25 milliseconds of latency.

ii. In contrast to the region based proposals and sliding window method, YOLO considers the entire image during training and test time where the information about classes as well as their appearance is encoded. In Fast R-CNN the larger context is not considered and the smaller objects in the background are identified as images. Compared to Fast R-CNN, YOLO makes very few background errors.

iii. Generalizable representations of objects are easily learnt by YOLO which makes it less likely to breakdown when applied on unexpected and new domain inputs.

The network uses features from the entire image to predict each bounding box. For all classes for an image, bounding boxes are simultaneously predicted. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision [8].

The following points explain the process of object detection and how bounding boxes are drawn [12].

As shown in Fig 3, An image is split into SxS grid, within each of the grid m bounding boxes are taken.

2. For each of the bounding box generated, a class probability and confidence scores values for the bounding box are given as output.
3. These confidence scores show how confident the model is that the box contains an object .
4. Each of the bounding box generated consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the boundaries of the grid cell. Relative to the whole image, the width and height are predicted.
5. Only those bounding boxes that have the class probability greater than or equal to the set threshold value are selected. These bounding boxes are used to locate the object in the image/frame.

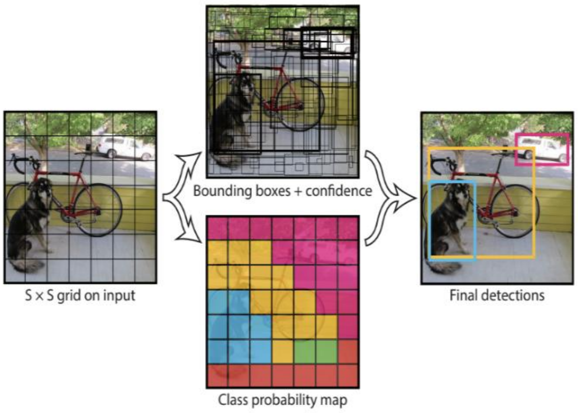

Fig 3. YOLO Working [22]

B. Distance Calculation

To enhance human detection and to calculate distance for humans, Haar cascade classifier is used [17], [18]. To train the haar-cascade classifier a set of 800 positive images and a set of 400 negative images were taken [19]. Here, the number of positive images should be double the number of negative images. Positive image set that was created contains humans and has 800 images, as shown in Fig 4 and the negative image set that was created contains random images that do not contain humans in any form and has 400 images, as shown in Fig 5. From the positive images, a vector file (pos.vec) is created which stitches the positive images together. A training command is run on server which takes positive images vector, a text file that contains locations of negative images, number of positive images (in his case: 800), number of negative images (in this case: 400), number of stages (stages determine the number of times the classifier should run to extract the features from the images given), and height and width of the training images.

Yesterday
Show Less
1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg 8.jpg
9.jpg 10.jpg 11.jpg 12.jpg 13.jpg 14.jpg 15.jpg 16.jpg
17.jpg 18.jpg 19.jpg 20.jpg 21.jpg 22.jpg 23.jpg 24.jpg
25.jpg 26.jpg 27.jpg 28.jpg 29.jpg 30.jpg 31.jpg 32.jpg
33.jpg 34.jpg 35.jpg 36.jpg 37.jpg 38.jpg 39.jpg 40.jpg
41.jpg 42.jpg 43.jpg 44.jpg 45.jpg 46.jpg 47.jpg 48.jpg
49.jpg 50.jpg 51.jpg 52.jpg 53.jpg 54.jpg 55.jpg 56.jpg

Fig 4. positive set of images

1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg 8.jpg
9.jpg 10.jpg 11.jpg 12.jpg 13.jpg 14.jpg 15.jpg 16.jpg
17.jpg 18.jpg 19.jpg 20.jpg 21.jpg 22.jpg 23.jpg 24.jpg
25.jpg 26.jpg 27.jpg 28.jpg 29.jpg 30.jpg 31.jpg 32.jpg
33.jpg 34.jpg 35.jpg 36.jpg 37.jpg 38.jpg 39.jpg 40.jpg
41.jpg 42.jpg 43.jpg 44.jpg 45.jpg 46.jpg 47.jpg 48.jpg
49.jpg 50.jpg 51.jpg 52.jpg 53.jpg 54.jpg 55.jpg 56.jpg

Fig 5. Negative set of images

After running the training command on the server, an XML file is returned which contains all the information about the features that is required to detect a human. This classifier is integrated into the YOLO model that was developed as explained in the section above and the following formula is used to calculate distance for humans from the camera dynamically:

(2 *3.14 *180)/ (width + height * 360) * 1000 + 3

C. Reading out labels

As the objects are identified, the class labels of the objects are used to create a set. These set labels are converted into a string with a user defined function. After the string is formed, GTTS engine is used which converts the string that contains object classes into an audio mp3 file which gets saved under a name.

A wrapper module called pygame is used to play the mp3 file that is created by GTTS module dynamically. Mixer package from this module is imported and load function of mixer is used to load the mp3 file and play it as the objects are getting detected. Whenever the object detected is of person class, distance of it from the camera is also read out along with the class label. For example, person: 21 inches.

D. Integration with Raspberry Pi

The Raspberrypi was connected to the laptop by getting the IP address of the PI [3]. This IP address was used to connect to it through the putty. The default login credentials have to be entered while accessing the pi. SSH and Camera interfaces have to be enabled. The python script , cfg file and .weight files must be placed in a single folder. The python file has to be run on the optimised IDE known as Thonny.

1. Multiple packages like OpenCV, gTTS, SSH, pygame were installed
2. The object detection program was run using optimised Python IDE, Thonny
3. Pi was accessed on the laptop using Putty , Xming Server, Windows Remote Control
4. Camera was interfaced to the pi through the CSI port

For audio output, a speaker/headphone was connected through the audio jack provided on the Raspberrypi

IV. EXPERMENTAL RESULTS

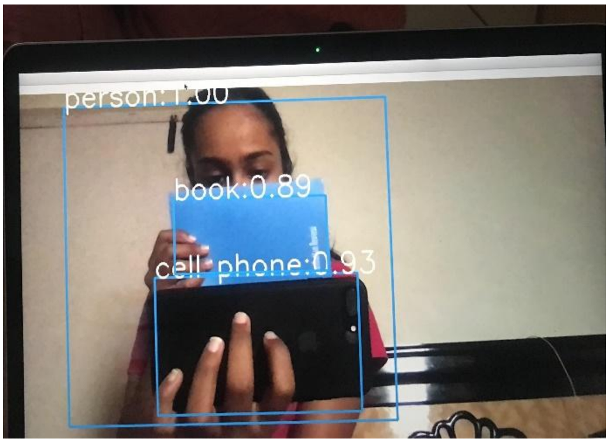

Fig 6. Running Object detector on Laptop

YOLO object detector was run on a CPU. For an object to be identified and classified, the threshold value was set to a standard value of 0.25. If for the object detected the threshold value was greater than or equal to 0.25, bounding boxes were generated and the class label was predicted according to the trained network. In Fig 6 the objects detected were person, book, cell phone with their corresponding confidence values of 1.00, 0.89,0.93 respectively. On the CPU the frames were processed at a rate of 3 FPS**.**

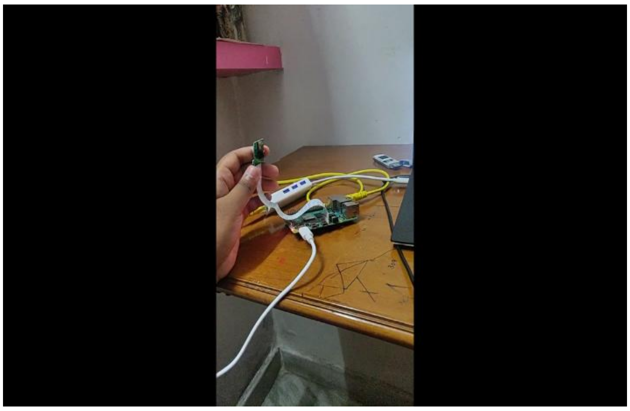

Fig 7. Connecting Raspberry Pi to laptop

To access the Operating System of Raspberry Pi and run the object detection model, it was connected to the laptop. The camera was connected by enabling the Camera Serial Interface.

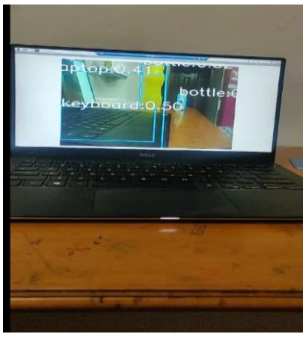

Fig 8. object detection running on Raspberry Pi viewed through laptop

The object detection model was run on Raspberry Pi using the python IDE 'Thonny'. The frames were processed at a rate of 0.6fps. As shown in Fig 8 the objects identified were laptop with a confidence of 0.41,keyboard with a confidence of 0.50, two bottles with a confidence of 0.32 each. The time taken to identify objects was much slower compared to the regular CPU as the fps on Raspberry Pi is much lower than that of the CPU

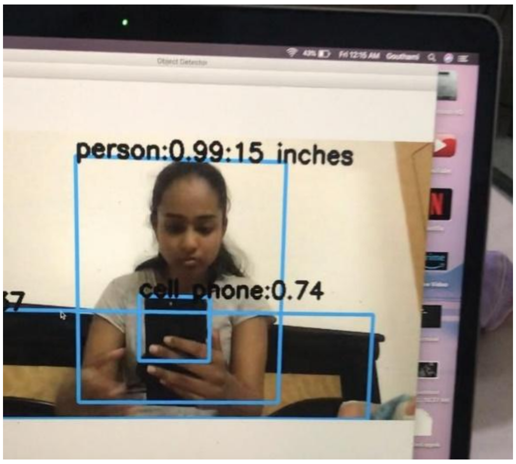

Fig 9. Distance measurement for humans along with detection

As shown in Fig 9. the objects detected were cell phone, bed and person with confidences of 0.74, 0.99, 0.67 respectively. The distance for a human from the monocular camera was calculated by running the Haar classifier which takes the contours of the human. The distance is shown in inches. For the person above the distance calculated by the classifier was 15 inches whereas in real time it was between 16-17inches. Therefore the accuracy of distance calculation is high.

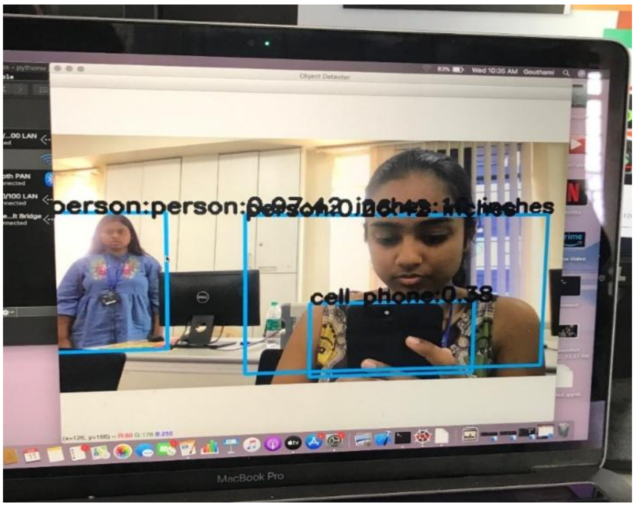

Fig 10. Distance measurement for two humans along with detection

As shown in Fig 10, the distances for two humans detected were 42 inches and 11 inches respectively

V. CONCLUSION

To conclude the paper, the obstacle detector that was be implemented will have diverse uses, the main one being an obstacle detector for the blind as their walking aide for smooth navigation. The other one being object detection in automated/driver less cars. The model trained classifies objects with high accuracy. After testing under different circumstances, the objects were accurately identified by the model developed using YOLO .The distance calculation and human detection enhancement was done using Haar Cascading Classifier which worked fairly well for multiple distances. As the objects were identified, a text-to-speech engine was used to store the labels of objects and distance in inches for humans as an MP3 file and read out dynamically using Pygame. Under the future scope of this project the following two modules have to be looked into and improvised

i. An efficient method for the distance calculation for multiple objects monocular camera is yet to be formulated. 3D reconstruction which is under research ,can be used to calculate the distance for multiple objects.

ii. The processing of frames on the Raspberry Pi is less compared to that of a CPU and GPU. This makes it very slow with a delay of 50 seconds. A method to increase the speed of processing of the frames must be implemented.

REFERENCES

1. Jianan Li , Xiaodan Liang, Jianshu Li , Yunchao Wei , Tingfa Xu , Jiashi Feng, and Shuicheng Yan, "Multistage Object Detection with Group Recursive Learning", IEEE TRANSACTIONS ON MULTIMEDIA, Vol. 20, Pages 1645-1655, 2018.
2. Shuai Zhang (Member, IEEE), Chong Wang (Member, IEEE), Shing-Chow Chan (Member, IEEE), Xiguang Wei, and Check-Hei Ho, "New Object Detection, Tracking, and Recognition Approaches for Video Surveillance Over Camera Network", IEEE SENSORS JOURNAL, Vol. 15, Pages 2679-2692, 2015.
3. Radhika Kamath, Mamatha Balachandra (Member, IEEE), and Srikanth Prabhu(Member, IEEE), "Raspberry Pi as Visual Sensor Nodes in Precision Agriculture: A Study", IEEE Access, Vol. 7, Pages 45110 - 45122, 2019.
4. Sachin Umesh Sharma, Dharmesh J. Shah, "A Practical Animal Detection and Collision Avoidance System Using Computer Vision Technique", IEEE Access, Vol. , Pages 347-359, 2017
5. Duy Thanh Nguyen, Tuan Nghia Nguyen, Hyun Kim, Hyuk-Jae Lee, "A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 27, Pages 1861 - 1873, 2017
6. Xing Wang, Tingfa Xu, Jizhou Zhang, Sining Chen, Yizhou Zhang, "SO-YOLO Based WBC Detection With Fourier Ptychographic Microscopy", IEEE Access, Vol. 6, Pages 51566 - 51576, 2018
7. Wei Fang, Lin Wang, Peiming Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments", IEEE Access, Vol. 8, Pages 1935 - 1944, 2020
8. Hualin Yang ; Long Chen ; Miaoting Chen ; Zhibin Ma ; Fang Deng ; Maozhen Li ; Xiangrong Li, "Tender Tea Shoots Recognition and Positioning for Picking Robot Using Improved YOLO-V3 Model", IEEE Access, Vol. 7, Pages 180998 - 181011, 2019
9. German, Parisia, Stefan Warmter, "Continual lifelong learning with neural networks: A review", Elsevier Neural Networks, Vol. 113, pages 54-71, 2019
10. Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, Xindong Wu, "Object Detection with Deep Learning: A Review", IEEE Paper, Volume 30, Pages 20, 2019
11. Li Cuimei, Qi Zhiliang , Jia Nan , Wu Jianhua, "Human face detection algorithm via Haar cascade classifier combined with three additional classifiers", IEEE paper, volume 13,Pages 5, 2017
12. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", IEEE Paper, Volume , Pages 10, 2016
13. Qichang Hu, Sakrapee Paisitkriangkrai, Chunhua Shen, Anton van den Hengel, Fatih Porikli, "Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework", IEEE Transactions on Intelligent Transportation Systems, Vol. 17, Pages 1002 - 1014, 2016
14. Bastian Leibe, Konrad Schindler, Nico Cornelis, Luc Van Gool, "Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 30, Pages 1683 - 1698, 2008
15. Dmitry Batenkov, "Real-time detection with webcam", ACM, Vol. 16, Pages 1528-1972, 2010
16. Ruan, Jiyang & Wang, Zhili "An Improved Algorithm for Dense Object Detection Based on YOLO". 10.1007/978-3-030-14680-1_65 ,2020
17. Jürgen Schmidhuber, "Deep learning in neural networks: An overview", Elsevier International Neural Network Society, Vol. 61, Pages 85-117, 2015
18. Jun Nishimura, Tadahiro Kuroda, "Versatile Recognition Using Haar-Like Feature and Cascaded Classifier", IEEE Sensors Journal, Vol. 10, Pages 942 - 951, 2010
19. Archit Gajjar,Xiaokun Yang,Lei Wu,Hakduran Koc,Ishaq Hasanali Unwala, Yunxiang Zhang,Yi Feng "An FPGA Synthesis of Face Detection Algorithm using HAAR Classifier" , ACM ICACS'18, pages 133-137, july 2018.
20. Junguk Cho,Shahnam Mirzaei,Jason K Oberg,Ryan Charles Kastner "Fpga-based face detection system using Haar classifiers", FPGA'09,pages 102-112, february 2009
21. Dataset representation taken from the open source official website - http://cocodataset.org/Yolo working and bounding boxes generation is taken from the website-https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

AUTHORS PROFILE

Eliganti Ramalakshmi, pursued her B.Tech from JNTU, Kakinada and M.Tech in C.S.E from JNTU, Anatapur. She is currently working as an Asst. Professor in the Department of Information Technology, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, India. She has teaching experience of 18 years. She guided many undergraduate projects in computer vision and Computer Graphics. Her research interests include Internet of Things, Digital Image Processing and Computer Graphics. She is also a panel member to assess major projects done by students of the Department of Information technology, CBIT. She has around 15 publications in various reputed journals and is a member of ISTE.

Dixitha Kasturi is a student of Bachelor of Engineering in Information Technology, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, India. She was born in 1998. Her research interests include Data Science , Artificial Intelligence, Machine Learning ,Big Data, Data Mining, IoT , and Cloud Computing. She has done projects in different fields such as Web development, building Android application and an Arduino project which alerts the blind person using a buzzer if there is an obstacle in the way which is detected by an ultrasonic sensor. One of her projects related to machine learning was developing a CNN model to classify the sign language numbers from 0 to 9 using images which could be used for people with hearing impairment.

Gouthami V is a student of Bachelor of Engineering in Information Technology, Chaitanya Bharathi Institute of Technology (CBIT), Hyderabad, India. She was born in 1998. Few of her notable projects include Classification of Clothing using deep learning and Image Steganography. 1. Fashion classification model was developed using MNIST dataset. The model developed could identify clothing items and attach a class label to them (a total of 9 class labels and 60,000 images were using for training). 2. Image Steganography provides true secrecy. This project allows a user to hide confidential information inside an image for secure transmission. Due to very minor pixel changes, third person will not be able to notice even the existence of data inside the image.

Retrieval Number: D7868049420/2020©BEIESP
DOI: 10.35940/ijeat.D7868.049420
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication
© Copyright: All rights reserved.