RAMAIAH
Institute of Technology

Drone Detection Using Deep Learning

Submitted as part of the final year project (ECP)

BACHELOR OF ENGINEERING
In
Electronics and Communication

Name of the students:

Sinchana S R	1MS17EC106	sinchu.sr@gmail.com	9538917997
Soma Rohith	1MS17EC109	mintu1842000@gmail.com	8639370617
Sushmith T	1MS17EC119	sushmith.thuluva@gmail.com	9740594943
Chiranthan K	1MS17EC135	chinnuk1729@gmail.com	9916313945

Under the Guidance of
Dr. Raghuram S
Associate Professor
Dept. of E&C,
RIT, Bengaluru

Department of Electronics and Communication

RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)
Accredited by National Board of Accreditation & NAAC with 'A' Grade
MSRIT Post, MSR Nagar, Bengaluru, Karnataka 560054
www.msrit.edu

2021

Second Review Report

Introduction:

The rapid growth of technology in the field of Unmanned Aerial Vehicles (UAV's) is making the process of building and controlling drones very easy and accessible, thus giving rise to a plethora of drones flying around in the neighborhood. However, such a development when fallen into the wrong hands creates an alarming danger for the safety and security of an organization and society. This potential danger arises as a result of the wide range of designs and capabilities of the drones. At one end of the scale, they are just a few feet in diameter and able to carry payloads like a camera or small Raspberry Pi computer whereas at the other end they have a wingspan of 20 meters and can carry 500-pound laser-guided bombs or Hellfire missiles.

The above-mentioned issue can be of major concern for the military of a country and hence needs continuous and diligent monitoring of the surrounding to prevent and prohibit the infiltration of such drones, hence leading to a requirement of a low-cost and effective technique for drone detection. This drone detection mechanism must uniquely detect drones and omit any other flying objects such as birds, airplanes, leaves, and other miscellaneous objects in the surroundings.

As a result of which, the idea of our project is to employ a deep learning model that will be trained exclusively to detect drones and build supporting hardware to acquire the data. The hardware of our project is intended to incorporate a video recording device providing a continuous flow of the surrounding information that will be immediately checked with the model so developed for spotting any drones in the footage.

Motivation:

Drones have drastically changed the way we use them for surveillance, exploring the skies, and film breathtaking landscapes. As much as this revolutionary product changes our lives, however, it can pose a threat if it's used for sinister purposes.

It's disturbing to discover all the things that criminals can do with technology as advanced as this. With drones, they can exploit various kinds of privacy laws that aren't as strict as they should be. This makes it

easier for them to commit their devious acts of malevolence and get away with it.

There have been instances of the use of drones for Drug delivery, illegal weapon transfer, surveillance of restricted areas, warfare, and burglaries. This poses a serious threat, So there is a crucial need for drone detection to prevent these activities from happening in the future.

Methodology:

The shortcomings of manual capabilities and time constraints in detecting objects as drones from footages or images have inspired the intentions of our project to develop a simple computer algorithm that could locate the drones in a matter of milliseconds embodying the power of object detection algorithms. Its applications span multiple and diverse industries, from round-the-clock surveillance to real-time vehicle detection in smart cities. In short, these are powerful deep learning algorithms.

Image classification involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all of these problems are referred to as **object recognition**.

Hardware model for edge-based detection:

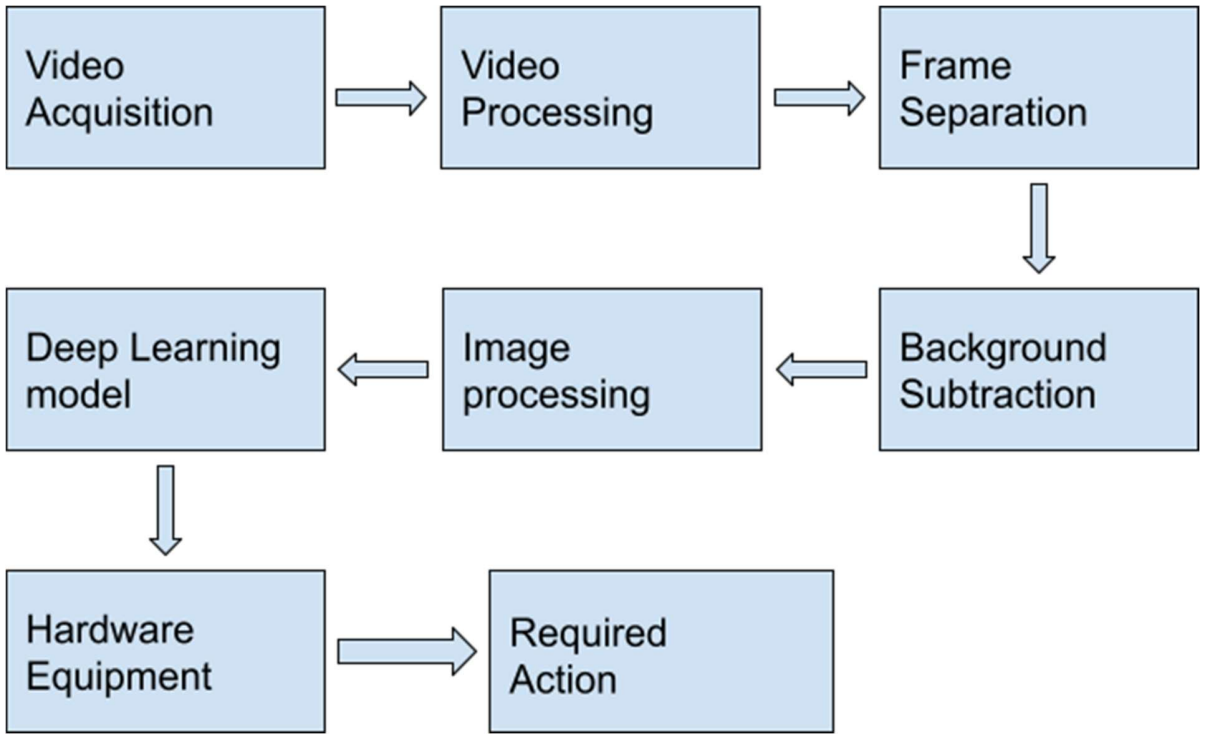
Edge detection includes a variety of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed *edges*. Edge detection is a fundamental tool in image processing, machine vision, and computer vision, particularly in the areas of feature detection. This algorithm when implemented with suitable hardware equipment can give us efficient results for drone detection.

The real-time implementation of drone detection and classification can be done by implementing an object detector on a single board computer. Here the object detector system will consist of a Raspberry Pi camera interfaced with Raspberry Pi 3 B+, which is capable of maintaining real-time frame rate keeping high precision. The Pi camera module is capable of taking both high-definition videos as well as still photographs.

Raspberry Pi is a small single-board computer capable of doing everything along with a great ability to interact with the outside world for real-time application of many projects. Among the different models, we would be using Raspberry Pi 3 B+ for the real-time detection of drones. This is the most recent version of Pi boards with CPU, GPU, USB ports, I/O pins, WiFi, Bluetooth, USB, and network boot onboard.

A Raspberry Pi camera module interfaced with this Pi 3 B+ board will help in obtaining video of the area under surveillance. The captured video has to be processed by separating different frames of it and by eliminating the background, these processed frames will be given to the deep learning model to identify the presence or absence of drones among each.

Block Diagram :



An image depicting the video monitoring of a drone

Technology Stack:

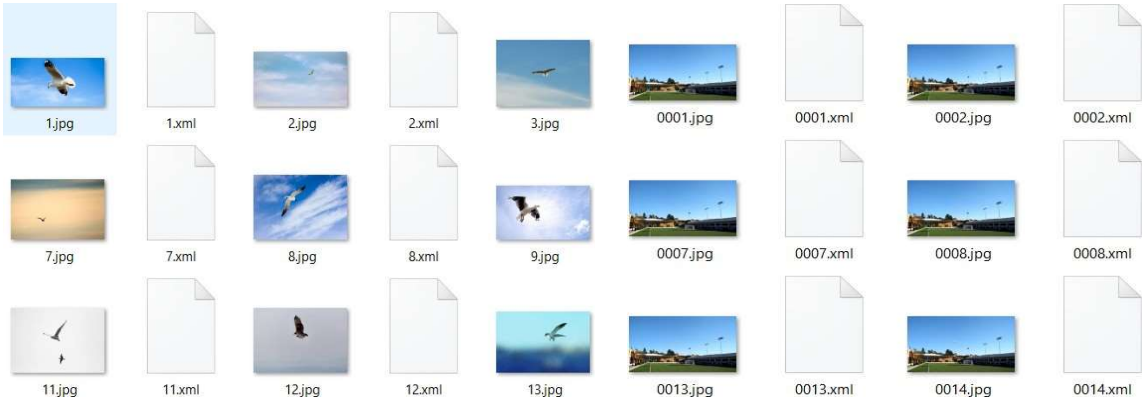
❑ Software

- OS
  - Windows
  - Ubuntu Version-16(Raspberry Pi)
- IDE
  - Anaconda
  - Jupyter Notebook/ Google Colab
- Frameworks
  - PyTorch
  - Keras
  - Open CV

❑ Hardware

- Raspberry Pi 3 B+
- Raspberry Pi Camera
- Jetson Nano

Dataset:



- https://www.kaggle.com/dasmehdixtr/drone-dataset-uav (image dataset + annotations in XML and txt format)
- Dataset(of birds, drones, mixed) from video footages

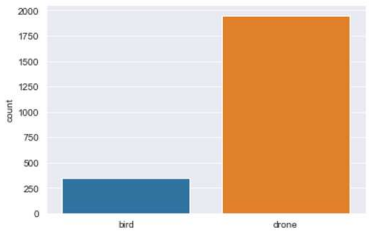
Dataset-new

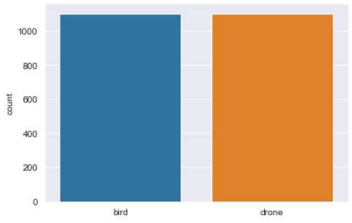
Manual dataset prepared,
sourcing images from Google, Kaggle, Pinterest.
Train - Drone: 1100 ; Bird: 1100
Test – Drone: 280 ; Bird: 280

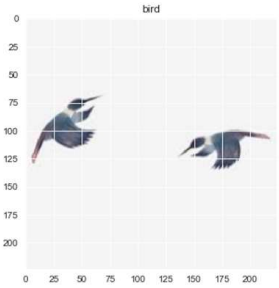
Results and Inference:

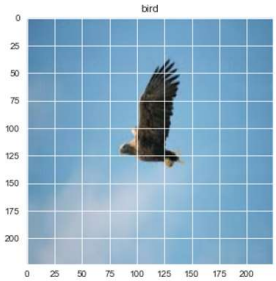
Results and Inference ▸ CNN-Classification

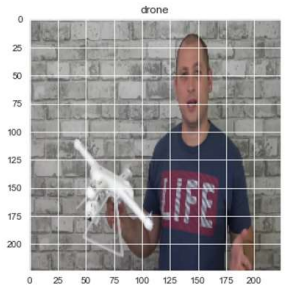
No. of images in train & test

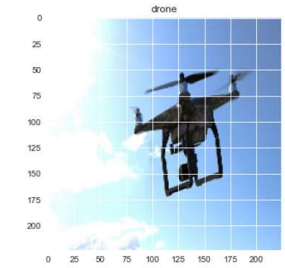












Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_2 (Conv2D)	(None, 56, 56, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 28, 28, 64)	0
dropout (Dropout)	(None, 28, 28, 64)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 128)	6422656
dense_1 (Dense)	(None, 2)	258

Total params: 6,451,554
Trainable params: 6,451,554
Non-trainable params: 0

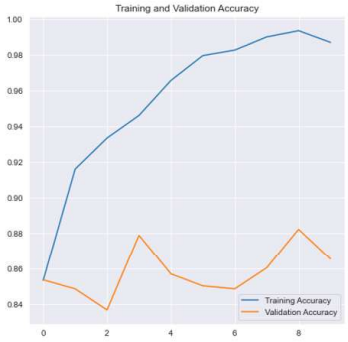
▸ CNN-Classification-dataset

```
Epoch 7/10
72/72 [==============================] - 149s 2s/step - loss: 0.0375 - accuracy: 0.9832 - val_loss: 0.4613 - val_accuracy: 0.8487
Epoch 8/10
72/72 [==============================] - 149s 2s/step - loss: 0.0282 - accuracy: 0.9889 - val_loss: 0.3834 - val_accuracy: 0.8605
Epoch 9/10
72/72 [==============================] - 149s 2s/step - loss: 0.0186 - accuracy: 0.9928 - val_loss: 0.4011 - val_accuracy: 0.8824
Epoch 10/10
72/72 [==============================] - 150s 2s/step - loss: 0.0420 - accuracy: 0.9858 - val_loss: 0.6840 - val_accuracy: 0.8655
```

▸ CNN-Classification-dataset-new

```
Epoch 22/25
69/69 [==============================] - 102s 1s/step - loss: 0.0380 - accuracy: 0.9900 - val_loss: 0.4154 - val_accuracy: 0.8875
Epoch 23/25
69/69 [==============================] - 100s 1s/step - loss: 0.0286 - accuracy: 0.9950 - val_loss: 0.4099 - val_accuracy: 0.8839
Epoch 24/25
69/69 [==============================] - 99s 1s/step - loss: 0.0333 - accuracy: 0.9921 - val_loss: 0.4805 - val_accuracy: 0.8750
Epoch 25/25
69/69 [==============================] - 100s 1s/step - loss: 0.0350 - accuracy: 0.9902 - val_loss: 0.3873 - val_accuracy: 0.8768
```

- CNN-Classification-dataset



	precision	recall	f1-score	support
Bird (Class 0)	0.58	0.75	0.65	100
Drone (Class 1)	0.95	0.89	0.92	495
accuracy			0.87	595
macro avg	0.76	0.82	0.78	595
weighted avg	0.88	0.87	0.87	595

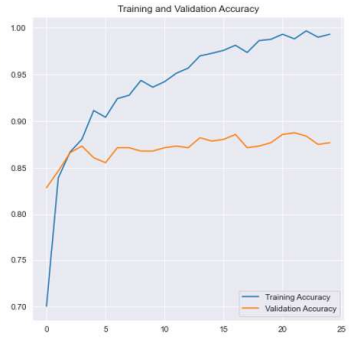
```
[ ] img = image.load_img('E:/FYP/Classification/dataset/test/drone/0293.jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))
```

```
[[0. 1.]]
[1]
```

```
[ ] img = image.load_img('E:/FYP/Classification/dataset/test/bird/155.jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))
```

```
[[1. 0.]]
[0]
```

- CNN-Classification-dataset-new



	precision	recall	f1-score	support
Bird (Class 0)	0.87	0.88	0.88	280
Drone (Class 1)	0.88	0.87	0.88	280
accuracy			0.88	560
macro avg	0.88	0.88	0.88	560
weighted avg	0.88	0.88	0.88	560

- CNN-Classification-dataset

```
[ ] img = image.load_img('E:/FYP/Classification/dataset-new/dataset/test/drone/OIP (32).jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))
```

```
[[0. 1.]]
[1]
```

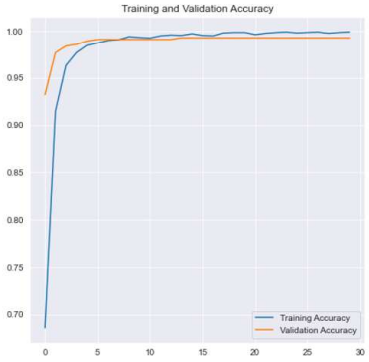
- CNN-Classification-dataset-new

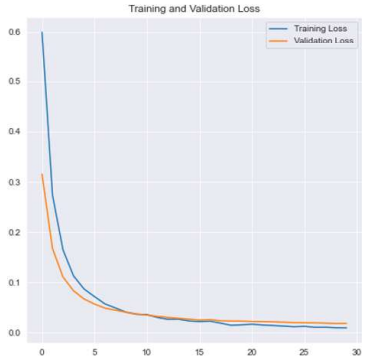
```
[ ] img = image.load_img('E:/FYP/Classification/dataset-new/dataset/test/bird/Brandt_Cormorant_0018_23090.jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))
```

```
[[1. 0.]]
[0]
```

Results and Inference ▸ Classification-MobilenetV2-dataset-new

```
Epoch 27/30
69/69 [==============================] - 78s 1s/step - loss: 0.0085 - accuracy: 0.9995 - val_loss: 0.0189 - val_accuracy: 0.9929
Epoch 28/30
69/69 [==============================] - 77s 1s/step - loss: 0.0089 - accuracy: 0.9984 - val_loss: 0.0183 - val_accuracy: 0.9929
Epoch 29/30
69/69 [==============================] - 76s 1s/step - loss: 0.0105 - accuracy: 0.9978 - val_loss: 0.0176 - val_accuracy: 0.9929
Epoch 30/30
69/69 [==============================] - 78s 1s/step - loss: 0.0087 - accuracy: 0.9994 - val_loss: 0.0180 - val_accuracy: 0.9929
```





	precision	recall	f1-score	support
Bird (Class 0)	1.00	0.99	0.99	280
Drone (Class 1)	0.99	1.00	0.99	280
accuracy			0.99	560
macro avg	0.99	0.99	0.99	560
weighted avg	0.99	0.99	0.99	560

Results and Inference ▸ Classification-MobilenetV2-dataset-new

Bird -> Class 0 Drone -> Class 1

```
[ ] img = image.load_img('E:/FYP/Classification/dataset-new/dataset/test/drone/OIP (32).jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))
```

```
[[0.00190556 0.9980945 ]]
[1]
```

```
[ ] img = image.load_img('E:/FYP/Classification/dataset-new/dataset/test/bird/Bronzed_Cowbird_0039_24026.jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))
```

```
[[0.9760527  0.02394728]]
[0]
```

Hardware Implementation

Jetson Nano:

- It's NVIDIA's smallest and lowest powered ES.
- Integrated GPU
- Powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing

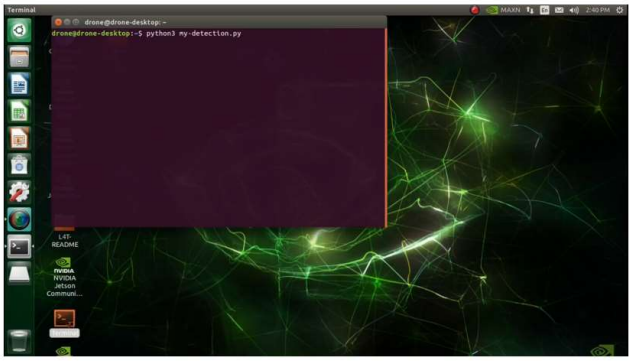
Real-time object detection:

- Jetson Nano interfaced with Rpi camera module v2
- Using SSD-mobilenet-v2 model
- Trained for 91-classes using MS-COCO dataset
- Uses Tensor-RT for real-time performance
- Function implemented in DetectNet network that return list of detections
- Saving the image with bounding boxes, label and confidence values

```
# load an image (into shared CPU/GPU memory)
img, width, height = jetson.utils.loadImageRGBA(opt.file_in)

# load the object detection network
net = jetson.inference.detectNet(opt.network, sys.argv, opt.threshold)

# detect objects in the image (with overlay)
detections = net.Detect(img, width, height, opt.overlay)

# print the detections
print("detected {:d} objects in image".format(len(detections)))

for detection in detections:
    print(detection)

# print out timing info
net.PrintProfilerTimes()

# save the output image with the bounding box overlays
if opt.file_out is not None:
    jetson.utils.saveImageRGBA(opt.file_out, img, width, height)
```

- TensorRT library for accelerating to real-time rates
- To create DetectNet object, we have loaded the SSD-mobilnet-v2 model
- Creating object for Camera module, by specifying the width, height and device file for video
- Display object for window to show the results
- Application loop to show the detection until video display is closed
- This function will return the image capture, this function will block until the next frame is available
- Detecting the object in the image
- Display each overlaid image

```
Open
my-detection.py
import jetson.inference
import jetson.utils

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5)
camera = jetson.utils.gstCamera(1280, 720, "/dev/video1")
display = jetson.utils.glDisplay()

while display.IsOpen():
    img, width, height = camera.CaptureRGBA()
    detections = net.Detect(img, width, height)
    display.RenderOnce(img, width, height)
    display.SetTitle("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
```



Conclusion

- Classification of Bird and Drone distinctly on PC (Accuracy as presented)
- Classification of Bird and Drone distinctly for each input image on Jetson
- Real time detection on Jetson

Future Scope

- Increased range
- Practical deployment of the prototype
- Differentiating different types of drones (Such as weaponized or not)
- Faster detection
- Counter-drone technologies

References:

[1] C. Aker and S. Kalkan, "Using deep networks for drone detection," *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Lecce, 2017, pp. 1-6, DOI:10.1109/AVSS.2017.8078539. https://ieeexplore.ieee.org/document/8078539

[2] W. Dai, T. Chang, and L. Guo, "Video object detection based on the spatial-temporal convolution feature memory model," *2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, Shenyang, China, 2020, pp. 312-317, DOI: 10.1109/ICPICS50287.2020.9202168. https://ieeexplore.ieee.org/document/9202168