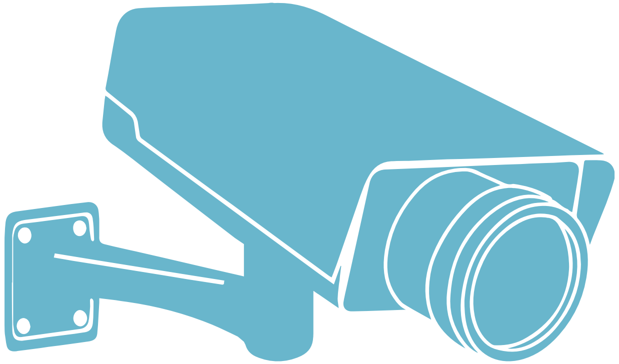
RAMAIAH
Institute of Technology

Final year Project (ECP)

Department of Electronics and Communication Engineering

Drone Detection Using Deep Learning

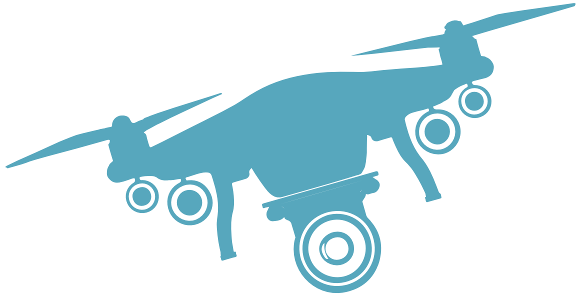


By:

Sinchana S R 1MS17EC106
Soma Rohith 1MS17EC109
Sushmith Thuluva 1MS17EC119
Chiranthan K 1MS17EC135

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

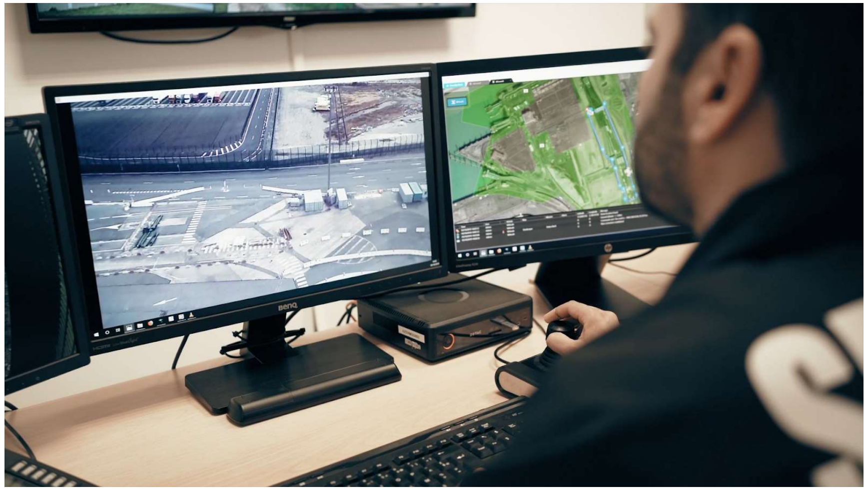
Introduction



- UAVs(drones) has seen a sudden usage increase in last few years, with high accessibility to public.
- Raised security concerns due to fact that these devices can intentionally or unintentionally cause serious hazards.
- Computer vision is extensively used autonomously compared to other available solutions such as RADAR, acoustics and RF signal analysis.
- Among these computer vision-based approaches, deep learning algorithms thanks are much effective.
- This project deals in with such implementation to obtain effective results in detection of drones.

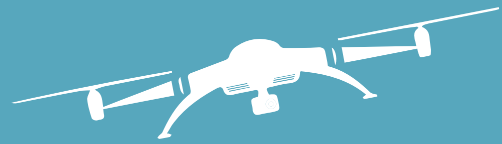
Problem Statement

- Surveillance on drones
- Protection of privacy
- Portability
- Operation feasibility
- Installation and Maintenance
- Affordability
- Range and Accuracy



Proposed Design

1. Video Acquisition
2. Video Processing
3. Frame Separation
4. Background Subtraction
5. Image Processing
6. Deep Learning model
7. Hardware Equipment
8. Drone Detection



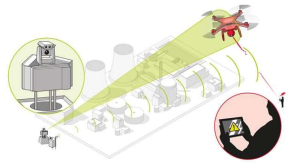
Methodology

- Drones detection and tracking from the real time video footages obtained from the camera
- Involves two processes:
  - **Image classification**: Assigning labels
  - **Object localization** :Coordinates mapping
- These two processes are together known as **Object recognition**
- Implementing it using deep learning model

- The real time implementation on single board computer:
  - Raspberry Pi 3/ NVIDIA Jetson Nano: processing the video captured
  - Raspberry Pi Camera module: capturing video

- Software includes:
  - Operating System: Windows / Ubuntu 16 (Rpi) / Ubuntu 18.04 (Jetson)
  - IDE: Anaconda, Google Colab
  - Frameworks: PyTorch, Keras, OpenCV, TensorFlow

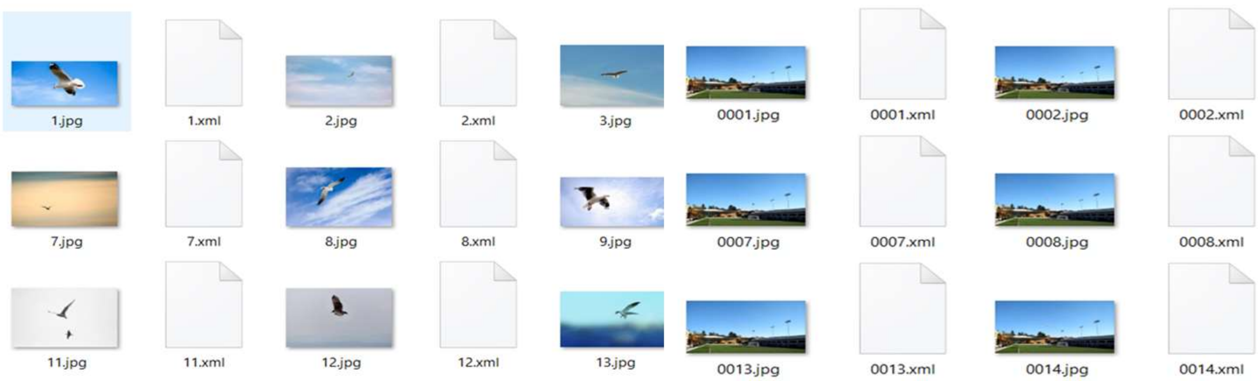
- Hardware includes:
  - NVIDIA Jetson Nano
  - Raspberry Pi 3 B+
  - Raspberry Pi CSI camera



Dataset

Dataset

- Images of drones and birds along with annotations (.xml files)
  - 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

- CNN-Drone (Classification)

Lenet-5 Architecture

Inference:

Training-accuracy: 93.55%

Validation-accuracy: 86.87%

```
[ ] # Building the CNN
    # Initialising the CNN
    classifier = Sequential()

[ ] # Convolution
    classifier.add(Conv2D(32, (3, 3), input_shape = (32, 32, 3), activation = 'relu'))

[ ] # Pooling
    classifier.add(MaxPooling2D(pool_size = (2, 2)))

[ ] # Adding a second convolutional layer
    classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
    classifier.add(MaxPooling2D(pool_size = (2, 2)))

[ ] # Flattening
    classifier.add(Flatten())

[ ] # Full connection
    classifier.add(Dense(units = 128, activation = 'relu'))
    classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

```
Epoch 9/10
50/50 [==============================] - 17s 334ms/step - loss: 0.1777 - accuracy: 0.9279 - val_loss: 0.3359 - val_accuracy: 0.8250
Epoch 10/10
50/50 [==============================] - 16s 316ms/step - loss: 0.1588 - accuracy: 0.9355 - val_loss: 0.3849 - val_accuracy: 0.8687
```

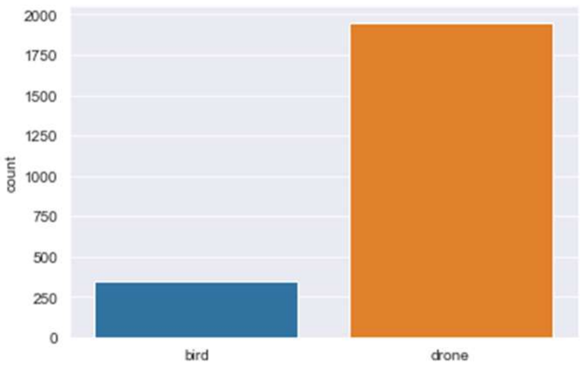
RAMAIAH Institute of Technology

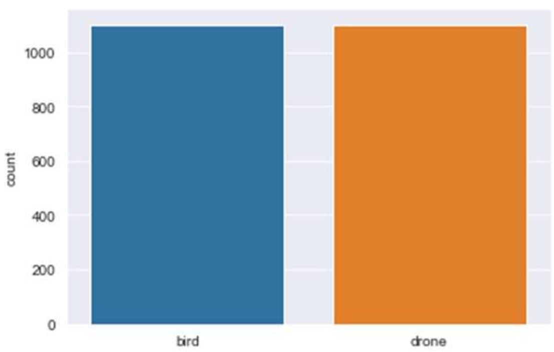
Department of Electronics and Communication Engineering

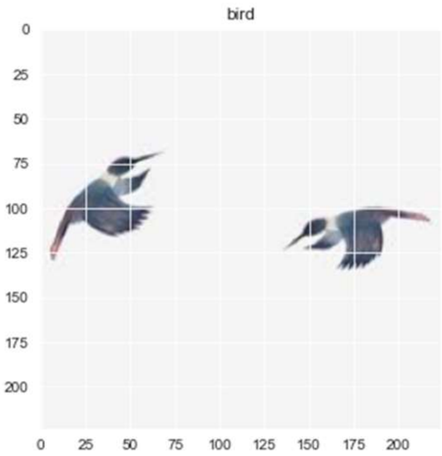
Drone Detection Using Deep Learning

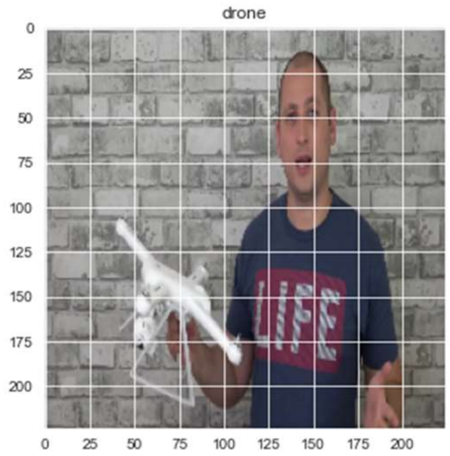
Results and Inference ▸ CNN-Classification

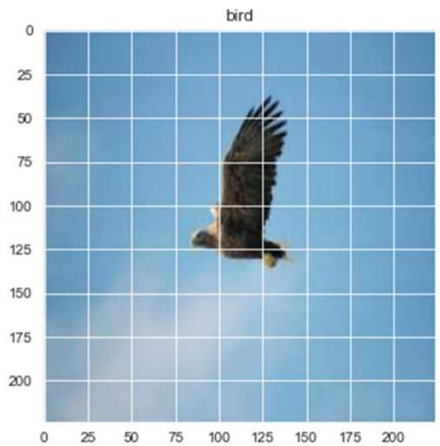
No. of images in train & test

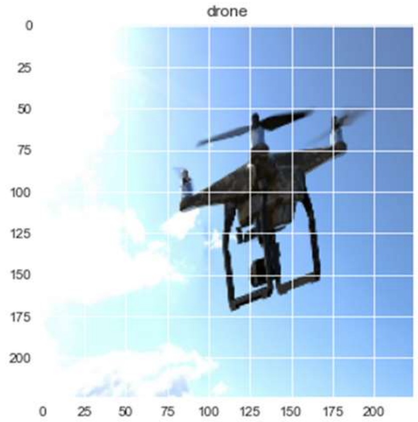













```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 32)      896
_________________________________________________________________
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

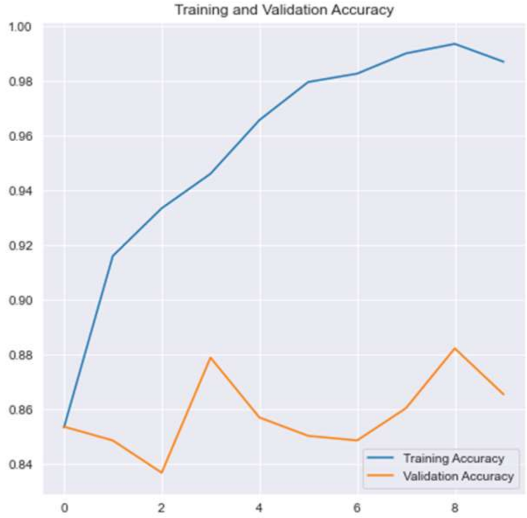
```
Epoch 22/25
69/69 [==============================] - 102s 1s/step - loss: 0.0380 - accuracy: 0.9900 - val_loss: 0.4154 - val_accuracy: 0.8875
Epoch 23/25
69/69 [==============================] - 100s 1s/step - loss: 0.0286 - accuracy: 0.9950 - val_loss: 0.4099 - val_accuracy: 0.8839
Epoch 24/25
69/69 [==============================] - 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
```

RAMAIAH Institute of Technology

Department of Electronics and Communication Engineering

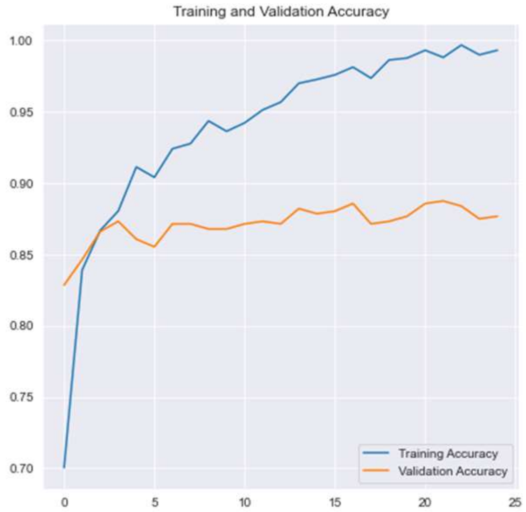
Drone Detection Using Deep Learning

▸ 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

▸ 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

```
[ ] 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]
```

- CNN-Classification-dataset

```
[ ] 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]
```

```
[ ] 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

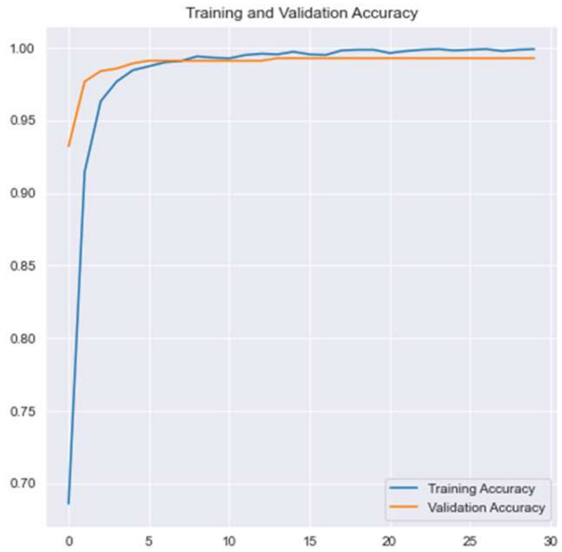
```
[ ] img = image.load_img('E:/FYP/Classification/dataset-new/dataset/test/bird/Brandt_Cormorant_0018_23090.jpg', target_size = (img_width, img_height))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis = 0)
    print(new_model.predict(img))
    print(new_model.predict_classes(img))

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

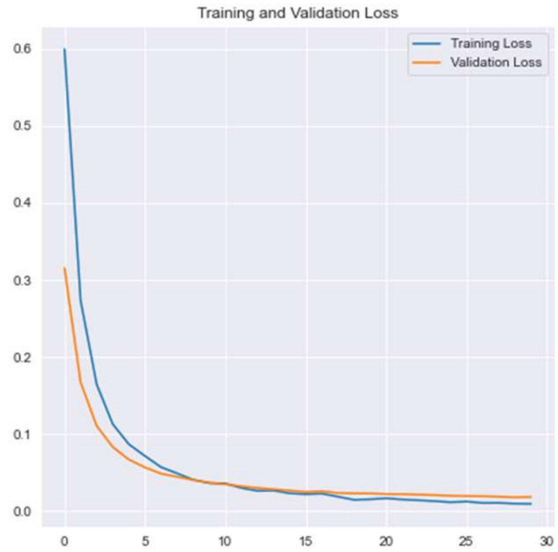
Department of Electronics and Communication Engineering

Drone Detection Using Deep Learning

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]
```

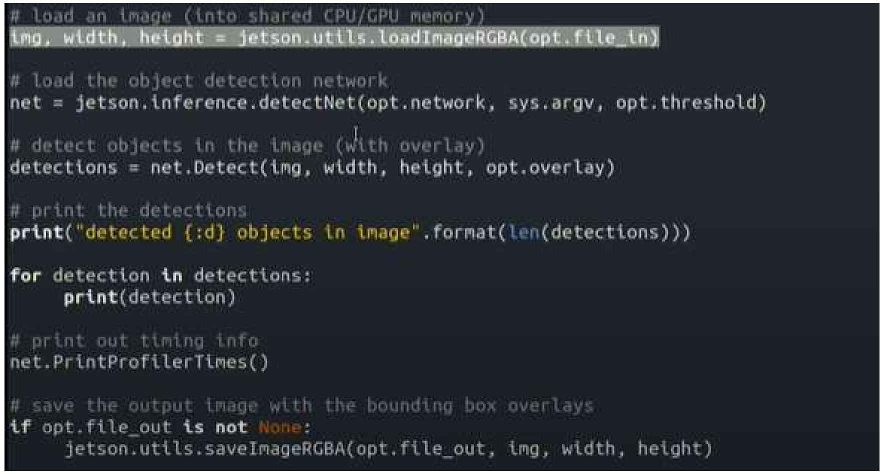
Hardware Implementation

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

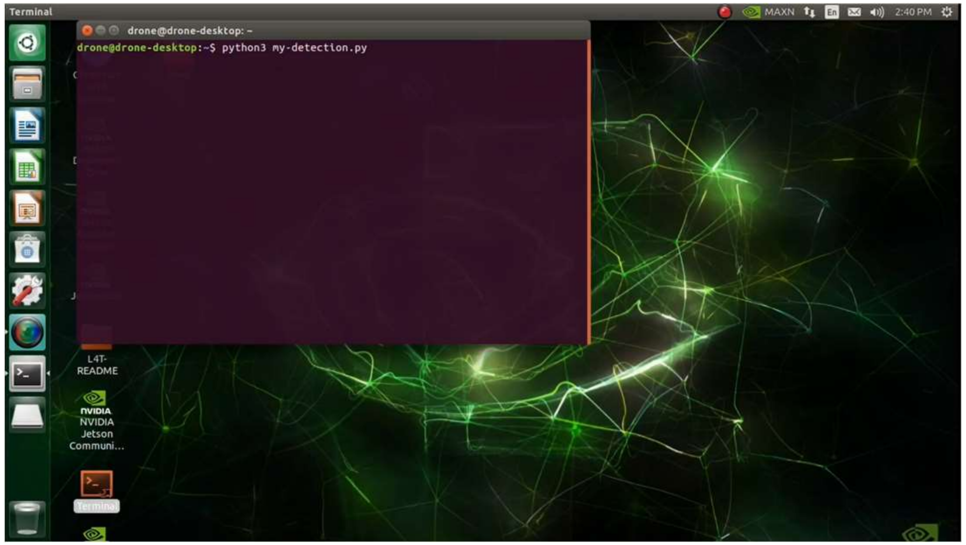
Real-time object detection:

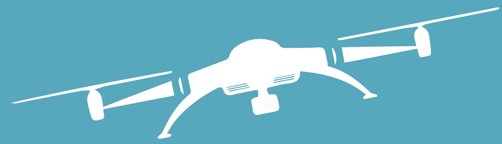
- Jetson Nano interfaced with Rpi camera module v2
- Using SSD-mobilenet-v2 model
- Trained for 91-classes using MS-COCO dataset
- Uses Tensor-RT for real-time performance
- Function implemented in DetectNet network that return list of detections
- Saving the image with bounding boxes, label and confidence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



Drone Detection Using Deep Learning

- 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





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

THANK YOU