International Engineering Journal For Research & Development

Vol.5
Issue 5

E-ISSN NO:-2349-0721

Impact factor : 6.549

AUTONOMOUS SURVEILLANCE DRONE

[1]Dhavalshree Khedkar, [2]Akshata Kasar, [3]Vishwas Shelke

Information Technology Department, Datta Meghe College of Engineering,
Airoli, Navi Mumbai, Maharashtra 400708, India[1,2,3]
[1]dhavalshree.khedkar@gmail.com, [2]ahkasar111@gmail.com, [3]vishwas3g@gmail.com

---

ABSTRACT

We present Autonomous Surveillance Drone, a new approach to serve purpose of UAV surveillance. Instead, just sending video stream from drone to display device at user's side we have developed complete system to get insights from video stream. Our system architecture is extremely simple and biggest reason for this is platforms provided by Nvidia and Microsoft, throughout this paper we are demonstrating all results/outputs using services and hardware provided by Nvidia and Microsoft. We are using the Nvidia's Jetson Development board to compile our inference part(object detection using YOLO-V3) along with stream and metadata fetch at remote terminal, Jetson nano is our IoT edge device, to connect edge device and the remote terminal we are using Microsoft Azure IoT hub (Cloud Service).

Keywords— Drone, Nvidia Jetson Nano,Microsoft Azure IoT hub, NVIDIA DeepStream SDK, YOLO.

INTRODUCTION

Humans glance at an image and instantly know what objects are present in that image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection (like YOLO-You Only Look Once) would allow computers to drive drone without specialized sensors, enable assistive devices to convey real-time scene information to human, and unlock the potential for general purpose responsive robotic systems. So we have prepared this whole system to do such task. We have divided it in three major parts – Hardware, Deep Learning, Internet of Things.

HARDWARE

The power of modern AI is now available for makers, learners, and embedded developers everywhere. One of the powerful hardware for this type of system is Nvidia Jetson Nano Board.

A. Nvidia Jetson Nano

NVIDIA Jetson Nano Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts[1]. Features of board are as follows –

- RAM-4 GBLPDDR4 at frequency 1600MHz
- CPU -Quad-core ARM Cortex-A57processor 1.4 GHZ
- GPU-NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores

- POWER-5 to 10 watts
- I/O – 4USB 3.0 ports, 2 Display ports, separate header for camera
- STORAGE - (MIN 16GB – MAX 128 GB) UHS-1 MicroSD card

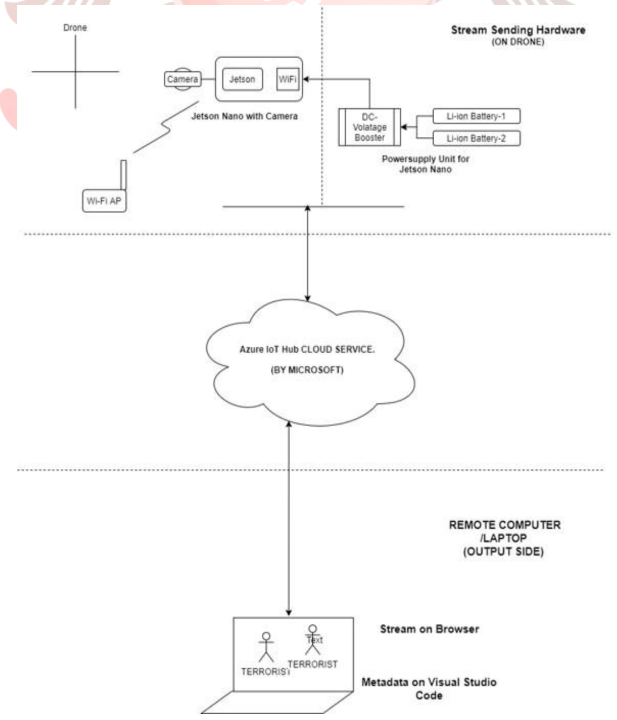
DEEP LEARNING

A. Prepare Own Dataset

As this system can be used for any specific purpose. If for that purpose you don't have any dataset than you can make your own dataset.Formation of dataset is major task while training a model, dataset for video object detection consists of various images. Now images can be formed using camera by extracting frames from video or can be directly downloaded from internet.

1)Collecting Images for Training: For preparing the dataset, download multiple images from Google by randomly searching by name, e.g. Car, Person, Bag, Terrorist, etc. This is the one way to gather images. Second way is extracting frames from video which is random YouTube video, or video which is recorded by camera, this can be achieved using simple python program. Formula is simple, duration of video let's say 5 seconds of video which contains 30 or 60 frames (again depends on camera settings) 5 x 30 = 150 frames(150 images) similar to that 5 x 60 = 300 frames(300 images).

*2) Resize into Fixed Resolution:*When we feed input data (training images) to Neural Network it should be in uniformed resolution. This applies to image data as well as for text data.To simplify this task we found one great website (refer below image).



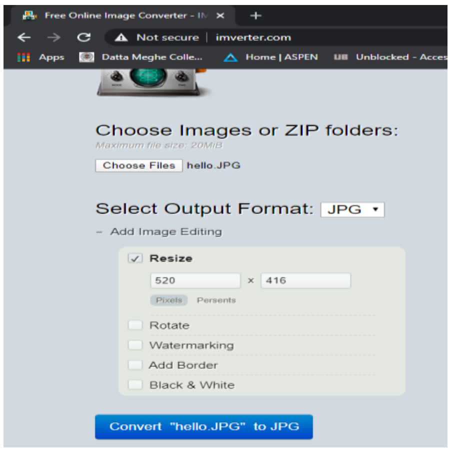


Fig. 1 Website

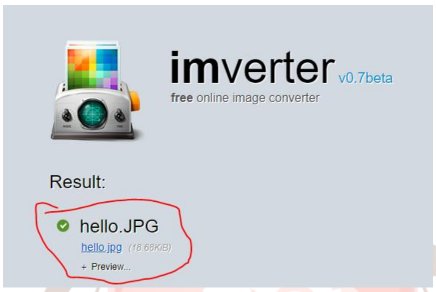
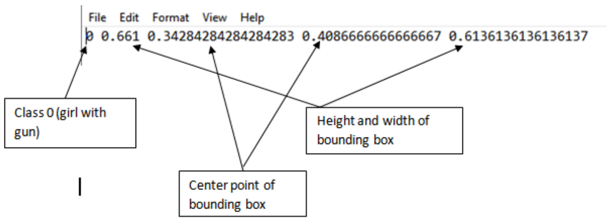


Fig. 2 Converted image ready to download

3)Make Bounding Boxes (Annotation of images) and Generating Labels: Annotation involves tagging of objects in an image, this basically generates a text file for each image with the proper co-ordinates and position of the object e.g. Person, Dog, Cat, Laptop etc. For tagging/annotating the images we have used https://github.com/drainingsun/ybat git-hub repository.



```
0 0.661 0.34284284284284283 0.40866666666666667 0.6136136136136137
```

File Edit Format View Help
Class 0 (girl with gun)
Height and width of bounding box
Center point of bounding box

Fig. 3 Format of label(.txt) file for each image

4) Separate Train and Test Data/Images: This step involves splitting of well formed dataset into train images and test images. Train images are directly fed into neural network with its labels. Test images are use to check whether model is predicting the correct objects or not, in simple words it is basically checking of accuracy of the model by making predictions. To separate train and test images we can use python script which is capable of splitting train and test data with some ratio which means 80% of training data and 20 % of testing data.

B. Training of Algorithm

Now we are ready with our dataset and training of ANN/DNN/YOLO is time consuming process, it will take lot of system resources especially GPU (Graphics Processing Unit).

1) Use of Darknet Framework to Train YOLO-V3 Model: Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. We have referred https://pjreddie.com/darknet/ website to implement the training. To install Darknet framework this website is having simple command listed onhttps://pjreddie.com/darknet/install/ website. You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Nvidia GTX 1660 Ti it processes images at 39 FPS. On a Jetson Nano platform it is giving us 15 FPS.

2) Steps of Training: https://github.com/dhaval18k/YOLO-V3-using-Darknet-Framework-with-NVIDIA-GPU we have made this repository on GitHub which is having all the required instructions to setup an environment for Darknet framework and also the steps to training our model. We also having our UDEMY Course for this[2].

3) Final weight Generation: At this stage our Darknet framework generates (.weight) file. Weights are used to connect the each neurons in one layer to the every neurons in the next layer, weight determines the strength of the connection of the neurons.

4) Iteration Concept: An iteration is a term used in ML and Deep Learning, and indicates the number of times the algorithm's parameters are updated. We have trained our model with 2000 iterations.

5) Batch training Concept: It consist of taking previously trained weights and train it further with more iterations to gain more accuracy than pervious. Major advantage of deep learning is, it does not overfits the model. The more you train it ,more the accuracy you get.

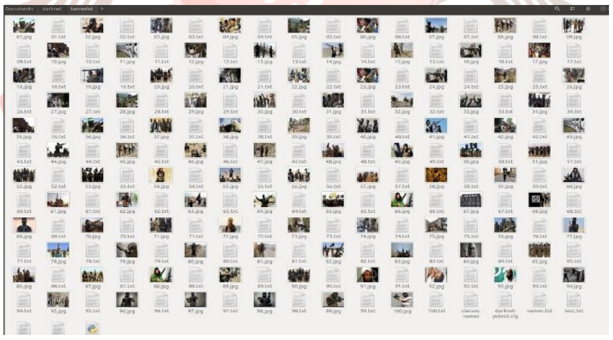


Fig. 4 Terrorists Dataset

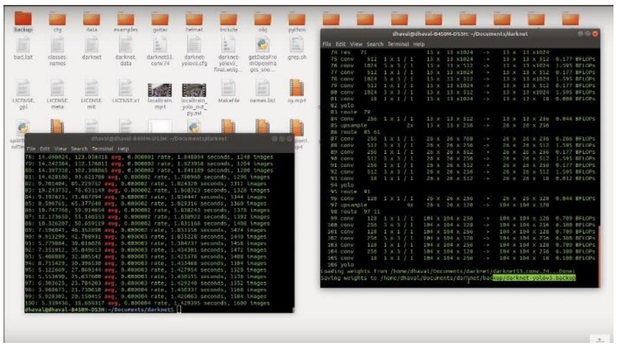


Fig. 5 Iterations of training

3) Final weight Generation: At this stage our Darknet framework generates (.weight) file. Weights are used to connect the each neurons in one layer to the every neurons in the next layer, weight determines the strength of the connection of the neurons.

4) Iteration Concept: An iteration is a term used in ML and Deep Learning, and indicates the number of times the algorithm's parameters are updated. We have trained our model with 2000 iterations.

5) Batch training Concept: It consist of taking previously trained weights and train it further with more iterations to gain more accuracy than pervious. Major advantage of deep learning is, it does not overfits the model. The more you train it ,more the accuracy you get.

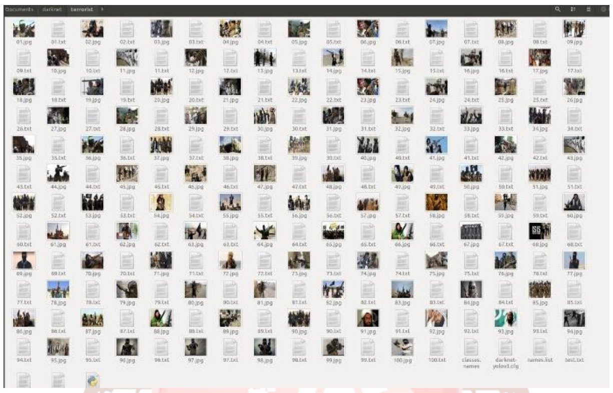
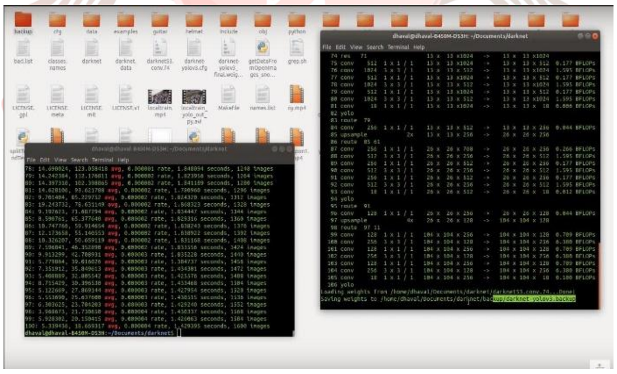


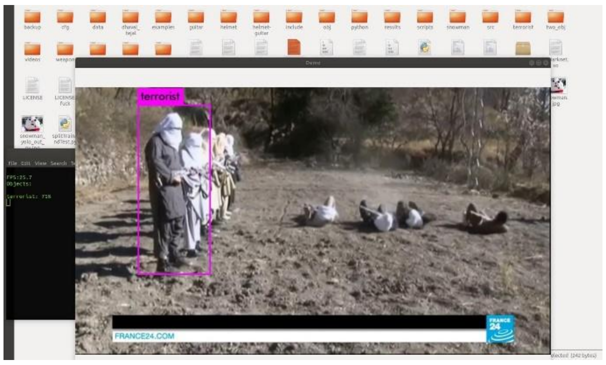
Fig. 6 Terrorists Dataset



```
dhaval@dhaval-B450M-DS3H: ~/Documents/darknet
78: 14.090024, 123.058418 avg, 0.000001 rate, 1.848094 seconds, 1248 images
79: 14.242384, 112.176011 avg, 0.000002 rate, 1.823958 seconds, 1264 images
80: 14.397318, 102.398865 avg, 0.000002 rate, 1.841189 seconds, 1280 images
81: 14.028106, 93.621788 avg, 0.000002 rate, 1.780960 seconds, 1296 images
82: 9.701404, 85.229752 avg, 0.000002 rate, 1.824320 seconds, 1312 images
83: 19.243732, 78.631149 avg, 0.000002 rate, 1.868323 seconds, 1328 images
84: 9.192671, 71.687294 avg, 0.000002 rate, 1.834447 seconds, 1344 images
85: 8.590761, 65.377648 avg, 0.000002 rate, 1.829316 seconds, 1360 images
86: 10.747768, 59.914654 avg, 0.000002 rate, 1.838243 seconds, 1376 images
87: 12.173658, 55.140553 avg, 0.000002 rate, 1.838922 seconds, 1392 images
88: 10.326207, 50.659119 avg, 0.000002 rate, 1.831168 seconds, 1408 images
89: 7.196841, 46.352898 avg, 0.000002 rate, 1.833558 seconds, 1424 images
90: 9.913299, 42.708931 avg, 0.000003 rate, 1.835228 seconds, 1440 images
91: 5.779884, 39.616026 avg, 0.000003 rate, 1.384737 seconds, 1456 images
92: 7.351912, 35.849613 avg, 0.000003 rate, 1.434381 seconds, 1472 images
93: 5.408889, 32.805542 avg, 0.000003 rate, 1.425576 seconds, 1488 images
94: 8.715429, 30.396538 avg, 0.000003 rate, 1.433468 seconds, 1504 images
95: 5.122669, 27.869144 avg, 0.000003 rate, 1.427954 seconds, 1520 images
96: 5.553690, 25.637608 avg, 0.000003 rate, 1.430515 seconds, 1536 images
97: 6.303625, 23.704203 avg, 0.000003 rate, 1.429240 seconds, 1552 images
98: 3.960673, 21.730610 avg, 0.000004 rate, 1.430337 seconds, 1568 images
99: 5.928302, 20.150415 avg, 0.000004 rate, 1.420063 seconds, 1584 images
100: 5.339436, 18.669317 avg, 0.000004 rate, 1.429395 seconds, 1600 images
dhaval@dhaval-B450M-DS3H:~/Documents/darknet$
```

```
dhaval@dhaval-B450M-DS3H: ~/Documents/darknet
   74 res   71                  13 x  13 x1024   ->    13 x  13 x1024
   75 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   76 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   77 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   78 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   79 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   80 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   81 conv     18  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x  18  0.006 BFLOPs
   82 yolo
   83 route  79
   84 conv    256  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x 256  0.044 BFLOPs
   85 upsample            2x    13 x  13 x 256   ->    26 x  26 x 256
   86 route  85 61
   87 conv    256  1 x 1 / 1    26 x  26 x 768   ->    26 x  26 x 256  0.266 BFLOPs
   88 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   89 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   90 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   91 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   92 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   93 conv     18  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x  18  0.012 BFLOPs
   94 yolo
   95 route  91
   96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128  0.044 BFLOPs
   97 upsample            4x    26 x  26 x 128   ->   104 x 104 x 128
   98 route  97 11
   99 conv    128  1 x 1 / 1   104 x 104 x 256   ->   104 x 104 x 128  0.709 BFLOPs
  100 conv    256  3 x 3 / 1   104 x 104 x 128   ->   104 x 104 x 256  6.380 BFLOPs
  101 conv    128  1 x 1 / 1   104 x 104 x 256   ->   104 x 104 x 128  0.709 BFLOPs
  102 conv    256  3 x 3 / 1   104 x 104 x 128   ->   104 x 104 x 256  6.380 BFLOPs
  103 conv    128  1 x 1 / 1   104 x 104 x 256   ->   104 x 104 x 128  0.709 BFLOPs
  104 conv    256  3 x 3 / 1   104 x 104 x 128   ->   104 x 104 x 256  6.380 BFLOPs
  105 conv     18  1 x 1 / 1   104 x 104 x 256   ->   104 x 104 x  18  0.100 BFLOPs
  106 yolo
Loading weights from /home/dhaval/Documents/darknet/darknet53.conv.74...Done!
Saving weights to /home/dhaval/Documents/darknet/backup/darknet-yolov3.backup
```

Fig. 7 Iterations of training



```
FPS:15.7
Objects:

terrorist: 71%
```

Fig. 8 Object Detected

C. VerifyAccuracy of Neural Network

1) Use of Python Script to Generate Plot of Accuracy:

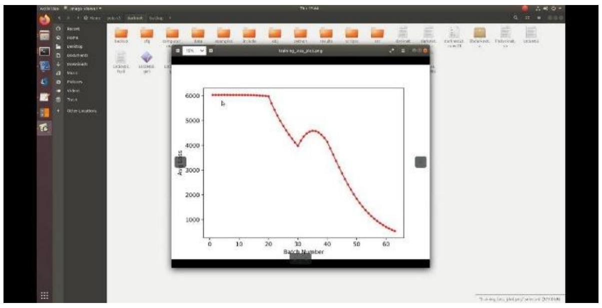


Fig. 9 60 iterations of training

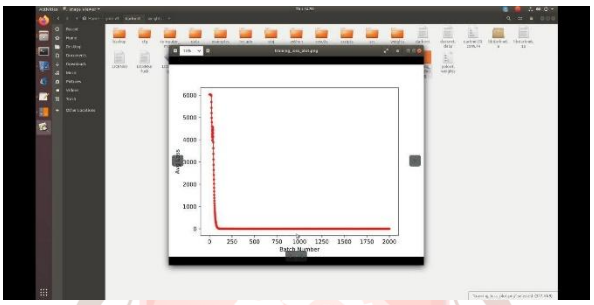
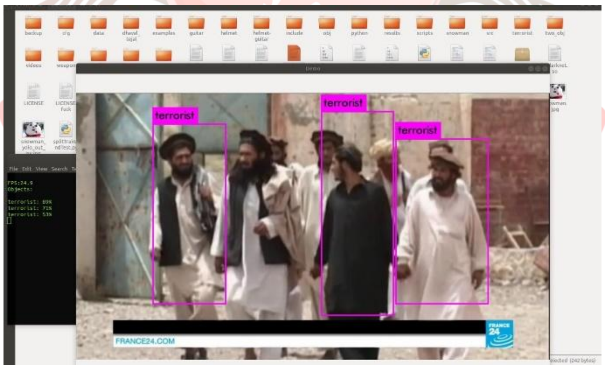


Fig. 10 2000 iterations of training

2)Test model on Test images and Video Sample:



```
FPS:24.9
Objects:

terrorist: 89%
terrorist: 71%
terrorist: 53%
```

Fig. 11 Detected Objects

INTERNET OF THINGS

NVIDIA Jetson Nano platform and NVIDIA Deepstream SDK, combining these both results in stream analysis/inference.NVIDIA's DeepStream SDK delivers a complete streaming analytics toolkit for AI-based video and image understanding, as well as multi-sensor processing. DeepStream is an integral part of NVIDIA Metropolis, the platform for building end-to-end services and solutions for transforming pixels and sensor data to actionable insights.

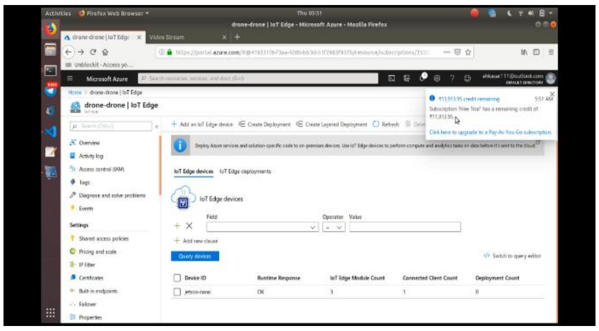
Microsoft Azure cloud provides the service to transfer stream and metadata to the remote local machine. Azure IoT hub is a managed IoT service which is hosted in the cloud. It allows bi-directional communication between IoT applications and the devices it manages. This cloud-to-device connectivity means that you can receive data from your devices, but you can also send commands and policies back to the devices. Steps for, from preparing Jetson nano setup to sending results on remote PC, are as follows.

A. Configuring the Jetson Nano Developer Kit[3]

1) The Jetson Nano Developer Kit uses a microSD card as a boot device and for main storage, the minimum recommended is a 16GB UHS-1 card. You'll need to power the developer kit with a good quality power supply that can deliver 5V⎓2A at the developer kit's Micro-USB port.
2) Download the Jetson Nano Developer Kit SD Card Image from https://developer.nvidia.com/jetson-nano-sd-card-image.
3) Write the image to your microSD card using a graphical program like Etcher or via command line.
4) Insert the microSD card (with system image already written to it) into the slot on the underside of the Jetson Nano module. Power on your computer display and connect it. Connect the USB keyboard and mouse. Connect your Micro-USB power supply (5V⎓2A). The Jetson Nano Developer Kit will power on and boot automatically. When you boot the first time, the Jetson Nano Developer Kit will take you through some initial setup.
5) Connect Jetson Nano to the internet. Then connect Jetson Nano to an SSH client.

B. Install IoT Edge

1) If not then first create Microsoft account. Create an IoT hub.
2) Register and then configure an IoT Edge device. Set the connection string on the IoT Edge device[4].
3) Now Install Azure IoT Edge for Ubuntu server 18.04 [5]. Connect your device to your IoT Hub using the manual provisioning option.



Device ID	Runtime Response	IoT Edge Module Count	Connected Client Count	Deployment Count
jetson-nano	OK	3	1	0

Fig. 12 Azure Dashboard

C. Install VS Code and its IoT Edge Extension on your Developer's Machine

1) On your developer's machine, get VS Code [6] and its IoT Edge extension[7].
2) Configure this extension with your IoT Hub[8].

D. Deploy Deepstream from the Azure Marketplace

In VS Code, from your development machine:

1) *Start by Creating a New IoT Edge Solution:*

i. Open the command palette (Ctrl+Shift+P)
ii. Select “Azure IoT Edge: New IoT Edge Solution”
iii. Select a parent folder
iv. Give it a name.
v. Select “Empty Solution” (if prompted, accept to install iotedgehubdev)

2) *Add the Deepstream Module to your Solution:*

i. Open the command palette (Ctrl+Shift+P).
ii. Select “Azure IoT Edge: Add IoT Edge module”.
iii. Select the default deployment manifest (deployment.template.json).
iv. Select “Module from Azure Marketplace”.
v. It opens a new tab with all IoT Edge module offers from the Azure Marketplace. Select the “Nvidia Deepstream SDK” one, select the NVIDIA DeapStream SDK 4.0.2 for Jetson plan and select the “latest” tag.

3) *Deploy the Solution to your Device:*

i. “Generate IoT Edge Deployment Manifest” by right clicking on the deployment.nano.template.json file.
ii. “Create Deployment for Single Device” by right clicking on the generated file in the /config folder.
iii. Select your IoT Edge device.

4) *Start Monitoring the Messages Sent from the Device to the Cloud:*

i. Right-click on your device (bottom left corner).
ii. Select “Start Monitoring Built-In Event Endpoint”.

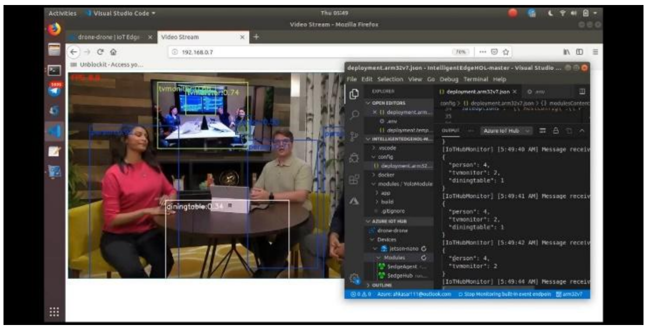
After a little while, you should be able to see messages sent by the Deepstream module to the cloud via the IoT Edge runtime in VS Code. These messages are the results of Deepstream processing a video and analysing it which detects terrorists, guns from the video and sends a message for each object found [9].

E. Run YOLO-V3 on Jetson Nano

The IntelligentEdgeHOL walks through the process of deploying an Azure IoT Edge module to an Nvidia Jetson Nano device to allow for detection of objects in YouTube videos, RTSP streams, or an attached web cam [10].

F. View the Processed Video:

Type the IP address of Jetson Nano on remote PC browser and detected objects with the bounding boxes will be displayed. Also metadata of the results can be seen on VS Code output terminal.


FPS
tvmonitor:0.74
diningtable:0.54
Video Stream - Mozilla Firefox
deployment.arm32v7.json - IntelligentEdgeHOL-master - Visual Studio ...
File Edit Selection View Go Debug Terminal Help
Azure IoT Hub

```
}
[IoTHubMonitor] [5:49:40 AM] Message receiv
{
  "person": 4,
  "tvmonitor": 2,
  "diningtable": 1
}
[IoTHubMonitor] [5:49:41 AM] Message receiv
{
  "person": 4,
  "tvmonitor": 2,
  "diningtable": 1
}
[IoTHubMonitor] [5:49:42 AM] Message receiv
{
  "person": 4,
  "tvmonitor": 2
}
[IoTHubMonitor] [5:49:44 AM] Message receiv
```

Fig. 13 Detected objects displayed in browser and metadata displayed in VS Code

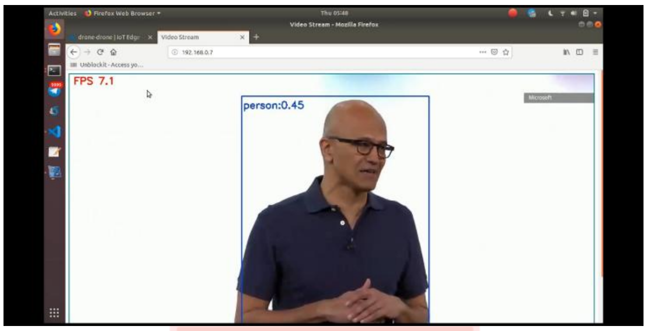


Fig. 14 Detected objects displayed in browser

CONCLUSIONS

The main aim of this system was to gain the insights from real-time video stream. As this system is a generalized solution for these types of purposes. If one wants to detect the objects from a particular area, then one can drove the drone to that area and can capture the stream to view the objects from that stream.

REFERENCES

[1] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[2] https://www.udemy.com/course/yolo-v3-using-darknet-framework-on-nvidia-gpu/

[3] https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit

[4] https://docs.microsoft.com/en-us/azure/iot-edge/quickstart-linux

[5] https://docs.microsoft.com/en-us/azure/iot-edge/how-to-install-iot-edge-linux

[6] https://code.visualstudio.com/

[7] https://marketplace.visualstudio.com/items?itemName=vsciot-vscode.azure-iot-tools#overview

[8] https://docs.microsoft.com/en-us/azure/iot-edge/how-to-deploy-modules-vscode#sign-in-to-access-your-iot-hub

[9] https://github.com/Azure-Samples/NVIDIA-Deepstream-Azure-IoT-Edge-on-a-NVIDIA-Jetson-Nano
https://github.com/Azure/IntelligentEdgeH