CHAPTER 1

1.INTRODUCTION

A drone, in a technological context, is an unmanned aircraft. Drones are more formally known as unmanned aerial vehicles (UAVs) or unmanned aircraft systems (UASes). Essentially, a drone is a flying robot. The aircraft may be remotely controlled or can fly autonomously through software-controlled flight plans in their embedded systems working in conjunction with onboard sensors and GPS. In the recent past, UAVs were most often associated with the military, where they were used initially for anti-aircraft target practice, intelligence gathering and then, more controversially, as weapons platforms. Drones are now also used in a wide range of civilian roles ranging from search and rescue, surveillance, traffic monitoring, weather monitoring and firefighting to personal drones and business drone-based photography, as well as videography, agriculture and even delivery services.

Origin of a drone can be traced with special techniques to provide relevant information to the military. The very first aircraft with reusable type radio control mechanism was designed in the 30s and it worked like a base model for all new advancements of today's world.Later, the military drones were developed with classic sensors and camera units and now they have been fixed inside missiles too. With so much advancement in technology, now you can easily find so many variants of drones. Few are used for military applications but others are finding the potential role in many big companies.As per a recent new update, Google and Amazon are developing their drones so that parcels can be delivered by air with ease. One more interesting concept is presented by Facebook as they are thinking to develop some giant drones that can carry the signal to remote locations for direct internet access.Drones in today's world have also been an important part of the film industry and news reporters are also using them to carry information from inaccessible locations.A typical unmanned aircraft is made of light composite materials to reduce weight and increase manoeuvrability. This composite material strength allows military drones to cruise at extremely high altitudes.UAV drones are equipped with a different state of the art technology such as infrared cameras, GPS and laser (consumer, commercial and military UAV). Drones are controlled by remote ground control systems (GSC) and also referred to as a ground cockpit.An unmanned aerial vehicle system has two parts, the drone itself and the control system.The nose of the unmanned aerial vehicle is where all the sensors and navigational systems are present. The rest of the body is full of drone technology systems since there is no space required to

accommodate humans.The engineering materials used to build the drone are highly complex composites designed to absorb vibration, which decrease the sound produced. These materials are very lightweight.

1.1 Drone Types and Sizes

UAV drones come in a wide variety of sizes, with the largest being mostly used for military purposes such as the Predator drone. The next in size is unmanned aircraft, which have fixed wings and require short runways. These are generally used to cover large sections of land, working in areas such as geographical surveying or to combat wildlife poaching.

VTOL Drones

Next in size for drones are what is known as VTOL drones. These are generally quadcopters but not all. VTOL drones can take off, fly, hover and land vertically. The exact meaning of VTOL is "Vertical Take-Off and Landing".

Quite a few of the latest small UAV drones such as the DJI Mavic Air 2 take VTOL to the next level and can be launched from the palm of your hand.

1.2 Radar Positioning & Return Home

The latest drones have dual Global Navigational Satellite Systems (GNSS) such as GPS and GLONASS.Drones can fly in both GNSS and non-satellite modes. For example, DJI drones can fly in P-Mode (GPS & GLONASS) or ATTI mode, which doesn't use GPS. Highly accurate drone navigation is very important when flying, especially in drone applications such as creating 3D maps, surveying landscape and SAR (Search & Rescue) missions.When the quadcopter is first switched on, it searches and detects GNSS satellites. High-end GNSS systems use Satellite Constellation technology. A satellite constellation is a group of satellites working together giving coordinated coverage and are synchronized so that they overlap well in coverage. Pass or coverage is the period in which a satellite is visible above the local horizon.

UAV Drone GNSS On Ground Station Remote Controller. The radar technology will signal the following on the remote controller display;

- The signal that is enough drone GNSS satellites have been detected and the drone is ready to fly
- Display the current position and location of the drone in relation to the pilot
- Record the home point for 'Return to Home' safety feature

Most of the latest UAVs have 3 types of Return to Home drone technology as follows;

- The pilot initiated a return to home by pressing a button on Remote Controller or in an app
- A low battery level, where the UAV will fly automatically back to the home point
- Loss of contact between the UAV and Remote Controller, with the UAV flying back automatically to its home point

The latest high-tech drones are now equipped with collision avoidance systems. These use obstacle detection sensors to scan the surroundings, while software algorithms and SLAM technology produce the images into 3D maps allowing the drone to sense and avoid. These systems fuse one or more of the following sensors to sense and avoid;

- Vision Sensor
- Ultrasonic
- Infrared
- Lidar
- Time of Flight (ToF)
- Monocular Vision

Gyroscope Stabilization, IMU and Flight Controllers

Gyro stabilization technology gives the UAV drone its smooth flight capabilities. The gyroscope works almost instantly to the forces moving against the drone, keeping it flying or hovering very smoothly. The gyroscope provides essential navigational information to the central flight controller.The inertial measurement unit (IMU) works by detecting the current rate of acceleration using one or more accelerometers. The IMU detects changes in rotational attributes like pitch, roll and yaw using one or more gyroscopes. Some IMU includes a magnetometer to assist with calibration against orientation drift.The Gyroscope is a component of the IMU and the IMU is an essential component of the drone's flight controller. The flight controller is the central brain of the drone.Here is a terrific article, which covers gyro stabilization and IMU technology in drones.

UAV Drone Propulsion Technology

The propulsion system (motors, electronic speed controllers and propellers) are the drone technology, which moves the UAV into the air and to fly in any direction or hover. On a quadcopter, the motors and propellers work in pairs with 2 motors/propellers rotating clockwise (CW Propellers) and 2 motors rotating Counter Clockwise (CCW Propellers).

They receive data from the flight controller and the electronic speed controllers (ESC) on the drone motor direction to either fly or hover.

Top UAV drone motors and propulsion systems are highly advanced and include the following components;

- Motor Stator
- Motor Bell (rotor)
- Windings
- Bearings
- Cooling System
- Electronic Speed Controllers
- ESC Updater
- Propellers
- Wiring
- Arm

keeping it flying or hovering very smoothly. The gyroscope provides essential navigational information to the central flight controller.The inertial measurement unit (IMU) works by detecting the current rate of acceleration using one or more accelerometers. The IMU detects changes in rotational attributes like pitch, roll and yaw using one or more gyroscopes. Some IMU includes a magnetometer to assist with calibration against orientation drift.The Gyroscope is a component of the IMU and the IMU is an essential component of the drone's flight controller. The flight controller is the central brain of the drone.Here is a terrific article, which covers gyro stabilization and IMU technology in drones.The propulsion system (motors, electronic speed controllers and propellers) are the drone technology, which moves the UAV into the air and to fly in any direction or hover. On a quadcopter, the motors and propellers work in pairs with 2 motors/propellers rotating clockwise (CW Propellers) and 2 motors rotating Counter Clockwise (CCW Propellers). They receive data from the flight controller and the electronic speed controllers (ESC) on the drone motor direction to either fly or hover. The latest drones have dual Global Navigational Satellite Systems (GNSS) such as GPS and GLONASS.Drones can fly in both GNSS and non-satellite modes. For example, DJI drones can fly in P-Mode (GPS & GLONASS) or ATTI mode, which doesn't use GPS.

CHAPTER 2

LITERATURE SURVEY

[1] Y. Yu, D. Barthaud, B. A. Price, A. K. Bandara, A. Zisman and B. Nuseibeh: Unmanned Aerial Vehicles (UAVs), or drones, are increasingly expected to operate in spaces populated by humans while avoiding injury to people or damaging property. However, incidents and accidents can, and increasingly do, happen. Traditional investigations of aircraft incidents require on-board flight data recorders (FDRs); however, these physical FDRs only work if the drone can be recovered. A further complication is that physical FDRs are too heavy to mount on light drones, hence not suitable for forensic digital investigations of drone flights. In this paper, we propose a self-adaptive software architecture, LiveBox, to make drones both forensic-ready and regulation compliant. We studied the feasibility of using distributed technologies for implementing the LiveBox reference architecture. In particular, we found that updates and queries of drone flight data and constraints can be treated as transactions using decentralised ledger technology (DLT), rather than a generic time-series database, to satisfy forensic tamper-proof requirements. However, DLTs such as Ethereum, have limits on throughput (i.e. transactions-per-second), making it harder to achieve regulation-compliance at runtime. To overcome this limitation, we present a self-adaptive reporting algorithm to dynamically reduce the precision of flight data without sacrificing the accuracy of runtime verification.

Summary: About drones/ Unmanned Aerial Vehicles

[2] D. Lee, W. Gyu La and H. Kim: As drone attracts much interest, the drone industry has opened their market to ordinary people, making drones to be used in daily lives. However, as it got easier for drone to be used by more people, safety and security issues have raised as accidents are much more likely to happen: colliding into people by losing control or invading secured properties. For safety purposes, it is essential for observers and drone to be aware of an approaching drone. In this paper, we introduce a comprehensive drone detection system based on machine learning. This system is designed to be operable on drones with camera. Based on the camera images, the system deduces location on image and vendor model of drone based on machine classification. The system is actually built with OpenCV library. We collected drone imagery and information for learning process.

Summary: Introduction to a drone detection.

[3] C. Liu, Y. Tao, J. Liang, K. Li and Y. Chen: Object detection based on the deep learning has achieved very good performances. However, there are many problems with images in real-world shooting such as noise, blurring and rotating jitter, etc. These problems have an important impact on object detection. Using traffic signs as an example, we established image degradation models which are based on YOLO network and combined traditional image processing methods to simulate the problems existing in real-world shooting. After establishing the different degradation models, we compared the effects of different degradation models on object detection. We used the YOLO network to train a robust model to improve the average precision (AP) of traffic signs detection in real scenes.

Summary: Object detection using YOLO

[4] N. Jmour, S. Zayen and A. Abdelkrim: This paper describes a learning approach based on training convolutional neural networks (CNN) for a traffic sign classification system. In addition, it presents the preliminary classification results of applying this CNN to learn features and classify RGB-D images task. To determine the appropriate architecture, we explore the transfer learning technique called “fine tuning technique”, of reusing layers trained on the ImageNet dataset in order to provide a solution for a four-class classification task of a new set of data.

Summary: Image classification using CNN

[5] Xiaowu Sun, Lizhen Liu, Hanshi Wang, Wei Song and Jingli Lu: With the rapid growth of images information, how to classify the images has been a main problem, and most of researchers are concerning on the neural networks to realize the images classification. However, the neural networks cannot escape from its own limitations including the local optimum or the dependence on the input sample data. In this paper, another new algorithm named support vector machine, whose main idea is to build a hyperplane as the decision surface, is introduced to solve the problems. In the theory part, in order to solve the optimal hyperplane for the separable patterns problem, the method of Lagrange multiplier is transformed into its dual problem. In the application section, where it proves that the support vector machine can solve the problem of classification perfectly, with regard to the input data, the eigenvalues of the images' gray information which are treated by the method of Principal Component Analysis are abstracted as input sample.

Summary: Image classification using SVM

CHAPTER 3

EXISTING METHOD

3.1 ANN:

An artificial neural network is an adaptive system that learns by using interconnected nodes or neurons in a layered structure that resembles a human brain. A neural network can learn from data so it can be trained to recognize patterns, classify data, and forecast future events.A neural network breaks down the input into layers of abstraction. It can be trained using many examples to recognize patterns in speech or images, for example, just as the human brain does. Its behavior is defined by the way its individual elements are connected and by the strength, or weights, of those connections. These weights are automatically adjusted during training according to a specified learning rule until the artificial neural network performs the desired task correctly.

Structure of ANN:

The idea of ANNs is based on the belief that working of human brain by making the right connections can be imitated using silicon and wires as living neurons and dendrites. The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward. ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value. Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple ANN. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value. Each link is associated with weight

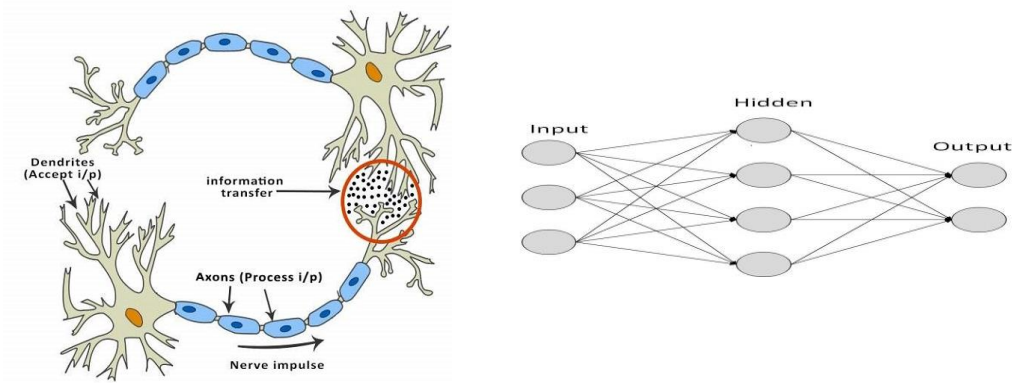

Fig1: Structure of ANN

This artificial neuron can be considered as a node and interconnection of nodes is called as network. Each node is connected by a link with numerical weights and these weights are stored in the neural network and updated through the process called learning. An ANN essentially has three layers of neurons namely, input layer, output layer and hidden layer.

3.2 Types of Artificial Neural Networks:

There are two Artificial Neural Network Topologies Feed Forward and Feedback.

Feed Forward ANN:

In this ANN, the information flow is unidirectional. A unit sends information to other unit from which it does not receive any information.They have fixed inputs and outputs. It is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer are connected with the nodes of the previous layers. The connection has different weights upon them. It may be divided into the following two types

- **Single layer feed forward network:** The concept is of feed forward ANN having only one weighted layer. In other words, we can say the input layer is fully connected

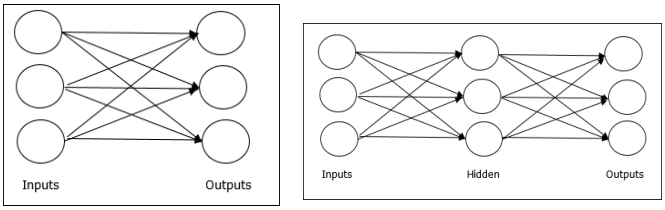

Fig2: Types of Artificial Neural Networks

- **Multilayer feed forward network:** The concept is of feed forward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.

Feedback Network:

As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types −

- **Recurrent networks:** They are feedback networks with closed loops. Following are the two types of recurrent networks.
- **Fully recurrent network:** It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.

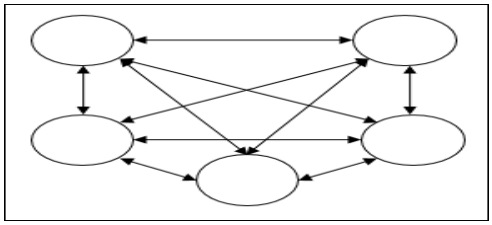

Fig3: Fully recurrent network

- **Jordan network:** It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.

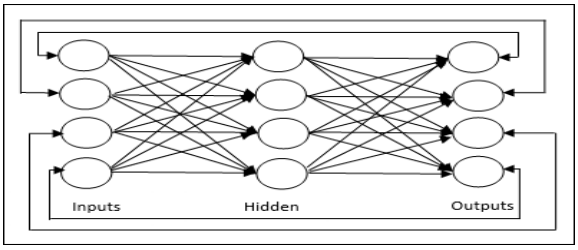

Fig4: Jordan network

3.3 Machine Learning in ANN:

The procedure that consists in estimating the parameters of neurons (setting up the weights) so that the whole network can perform a specific task is called learning. ANNs are capable of learning and they need to be trained. There are several learning strategies −

- **Supervised Learning:** It involves a teacher that is scholar than the ANN itself. For example, the teacher feeds some example data about which the teacher already knows the answers. For example, pattern recognizing. The ANN comes up with guesses while recognizing. Then the teacher provides the ANN with the answers. The network then compares it guesses with the teacher's "correct" answers and makes adjustments according to errors.
- **Unsupervised Learning:** It is required when there is no example data set with known answers. For example, searching for a hidden pattern. In this case, clustering i.e. dividing a set of elements into groups according to some unknown pattern is carried out based on the existing data sets present.
- **Reinforcement Learning:** This strategy built on observation. The ANN makes a decision by observing its environment. If the observation is negative, the network adjusts its weights to be able to make a different required decision the next time.

Activation Functions:

It may be defined as the extra force or effort applied over the input to obtain an exact output. In ANN, we can also apply activation functions over the input to get the exact output. Followings are some activation functions of interest −

- **Linear Activation Function:**

It is also called the identity function as it performs no input editing. It can be defined as $F(x) = x$

Sigmoid Activation Function:

It is of two types as follows −

- **Binary sigmoid function:** this activation function performs input editing between 0 and 1. It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1. It is also strictly increasing in nature, which means more the input higher would be the output. It can be defined as

$$F(x) = sigm(x) = 11 + exp(-x)$$

- **Bipolar sigmoid function:** this activation function performs input editing between -1 and 1. It can be positive or negative in nature. It is always bounded, which means

its output cannot be less than -1 and more than 1. It is also strictly increasing in nature like sigmoid function. It can be defined as

Perceptron:

Developed by Frank Rosenblatt by using McCulloch and Pitts model, perceptron is the basic operational unit of artificial neural networks. It employs supervised learning rule and is able to classify the data into two classes. Operational characteristics of the perceptron: It consists of a single neuron with an arbitrary number of inputs along with adjustable weights, but the output of the neuron is 1 or 0 depending upon the threshold. It also consists of a bias whose weight is always 1. Following figure gives a schematic representation of the perceptron.

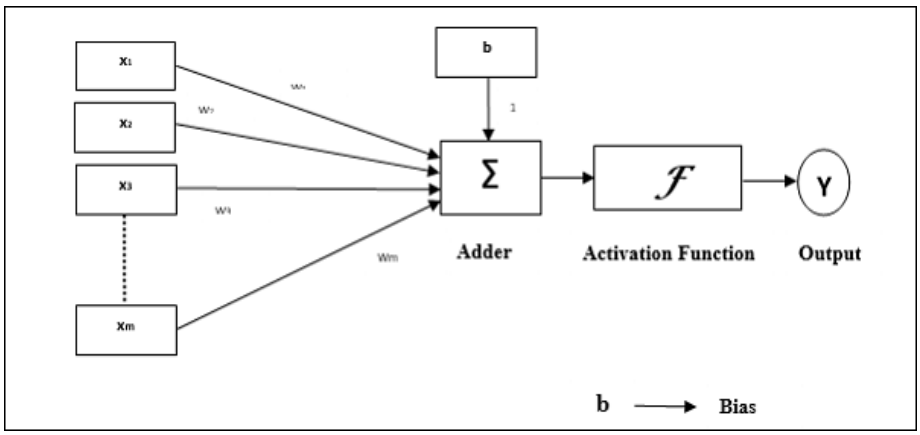

Fig5: Perceptron

Perceptron thus has the following three basic elements −

- **Links** − It would have a set of connection links, which carries a weight including a bias always having weight 1.
- **Adder** − It adds the input after they are multiplied with their respective weights.
- **Activation function** − It limits the output of neuron. The most basic activation function is a Heaviside step function that has two possible outputs. This function returns 1, if the input is positive, and 0 for any negative input.

Working of ANN:

In the topology diagrams shown, each arrow represents a connection between two neurons and indicates the pathway for the flow of information. Each connection has a weight, an integer number that controls the signal between the two neurons.

If the network generates a "good or desired" output, there is no need to adjust the weights. However, if the network generates a "poor or undesired" output or an error, then the system alters the weights in order to improve subsequent results.

3.4 ANN Based Classifiers:

ANN based classifiers are widely used for classification in pattern recognition applications. ANN will be trained to act as a classifier for a given problem. This includes the linear perceptron classifier, feed-forward networks, radial basis networks, recurrent networks and learning vector quantization (LVQ) networks structures. ANN based classification works in two phases namely training phase and testing phase. During training, the labeled data are used to train (learning) the network and to build the classifier model. During testing, the trained network (model) is given with the unlabeled data and it is classified based on the model built during training phase.

Why is it important?

Being a complex adaptive system, learning in ANN implies that a processing unit is capable of changing its input/output behavior due to the change in environment. The importance of learning in ANN increases because of the fixed activation function as well as the input/output vector, when a particular network is constructed. Now to change the input/output behavior, we need to adjust the weights.

Advantages of Artificial Neural Networks (ANN):

- Storing information on the entire network: Information such as in traditional programming is stored on the entire network, not on a database. The disappearance of a few pieces of information in one place does not restrict the network from functioning.
- The ability to work with inadequate knowledge: After ANN training, the data may produce output even with incomplete information. The lack of performance here depends on the importance of the missing information.
- It has fault tolerance: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault-tolerant.
- Having a distributed memory: For ANN to be able to learn, it is necessary to determine the examples and to teach the network according to the desired output by showing these examples to the network. The network's progress is directly proportional to the selected

instances, and if the event cannot be shown to the network in all its aspects, the network can produce incorrect output

- Gradual corruption: A network slows over time and undergoes relative degradation. The network problem does not immediately corrode.
- Ability to train machine: Artificial neural networks learn events and make decisions by commenting on similar events.
- Parallel processing ability: Artificial neural networks have numerical strength that can perform more than one job at the same time.

Disadvantages of Artificial Neural Networks (ANN):

- Hardware dependence: Artificial neural networks require processors with parallel processing power, by their structure. For this reason, the realization of the equipment is dependent.
- Unexplained functioning of the network: This is the most important problem of ANN. When ANN gives a probing solution, it does not give a clue as to why and how. This reduces trust in the network.
- Assurance of proper network structure: There is no specific rule for determining the structure of artificial neural networks. The appropriate network structure is achieved through experience and trial and error.
- The difficulty of showing the problem to the network: ANNs can work with numerical information. Problems have to be translated into numerical values before being introduced to ANN. The display mechanism to be determined here will directly influence the performance of the network. This depends on the user's ability.
- The duration of the network is unknown: The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.

Applications of Neural Networks:

- **Aerospace** − Autopilot aircrafts, aircraft fault detection.

- **Automotive** − Automobile guidance systems.
- **Military** − Weapon orientation and steering, target tracking, object discrimination, facial recognition, signal/image identification.
- **Electronics** − Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.
- **Financial** − Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document readers, credit application evaluators.
- **Industrial** − Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modeling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.
- **Medical** − Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
- **Speech** − Speech recognition, speech classification, text to speech conversion.
- **Telecommunications** − Image and data compression, automated information services, real-time spoken language translation.
- **Transportation** − Truck Brake system diagnosis, vehicle scheduling, routing systems.
- **Software** − Pattern Recognition in facial recognition, optical character recognition, etc.
- **Time Series Prediction** − ANNs are used to make predictions on stocks and natural calamities.
- **Signal Processing** − Neural networks can be trained to process an audio signal and filter it appropriately in the hearing aids.
- **Control** − ANNs are often used to make steering decisions of physical vehicles.
- **Anomaly Detection** − as ANNs are expert at recognizing patterns, they can also be trained to generate an output when something unusual occurs that misfits the pattern.

3.5 Support vector machines

Introduction to SVM:

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

Working of SVM:

An SVM model is basically a representation of different classes in a hyper plane in multidimensional space. The hyper plane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyper plane (MMH).

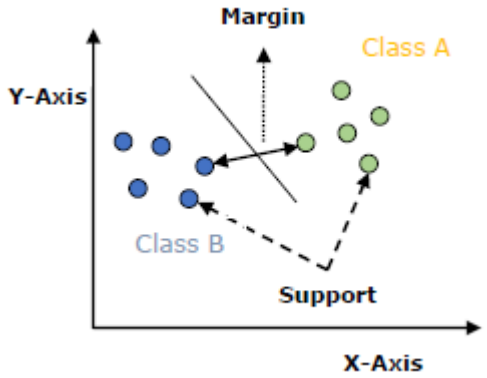

Fig6: Working of SVM

The followings are important concepts in SVM

- **Support Vectors:** Data points that are closest to the hyper plane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyper plane:** As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin:** It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to

the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyper plane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyper planes iteratively that segregates the classes in best way.
- Then, it will choose the hyper plane that separates the classes correctly.

A support vector machine (SVM) is a supervised learning algorithm that can be used for binary classification or regression. Support vector machines are popular in applications such as natural language processing, speech and image recognition, and computer vision. A support vector machine constructs an optimal hyper plane as a decision surface such that the margin of separation between the two classes in the data is maximized. Support vectors refer to a small subset of the training observations that are used as support for the optimal location of the decision surface. Support vector machines fall under a class of machine learning algorithms called kernel methods and are also referred to as kernel machines.

Training for a support vector machine has two phases:

1. Transform predictors (input data) to a high-dimensional feature space. It is sufficient to just specify the kernel for this step and the data is never explicitly transformed to the feature space. This process is commonly known as the kernel trick.
2. Solve a quadratic optimization problem to fit an optimal hyper plane to classify the transformed features into two classes. The number of transformed features is determined by the number of support vectors.

Disadvantages:

- Choosing a “good” kernel function is not easy.
- Long training time for large datasets.
- Difficult to understand and interpret the final model, variable weights and individual impact.
- Since the final model is not so easy to see, we cannot do small calibrations to the model hence it’s tough to incorporate our business logic.

CHAPTER 4

PROPOSED METHOD

Here, we are going to discuss the Yolo and CNN network. Below figures shows the model diagram of our proposed method.

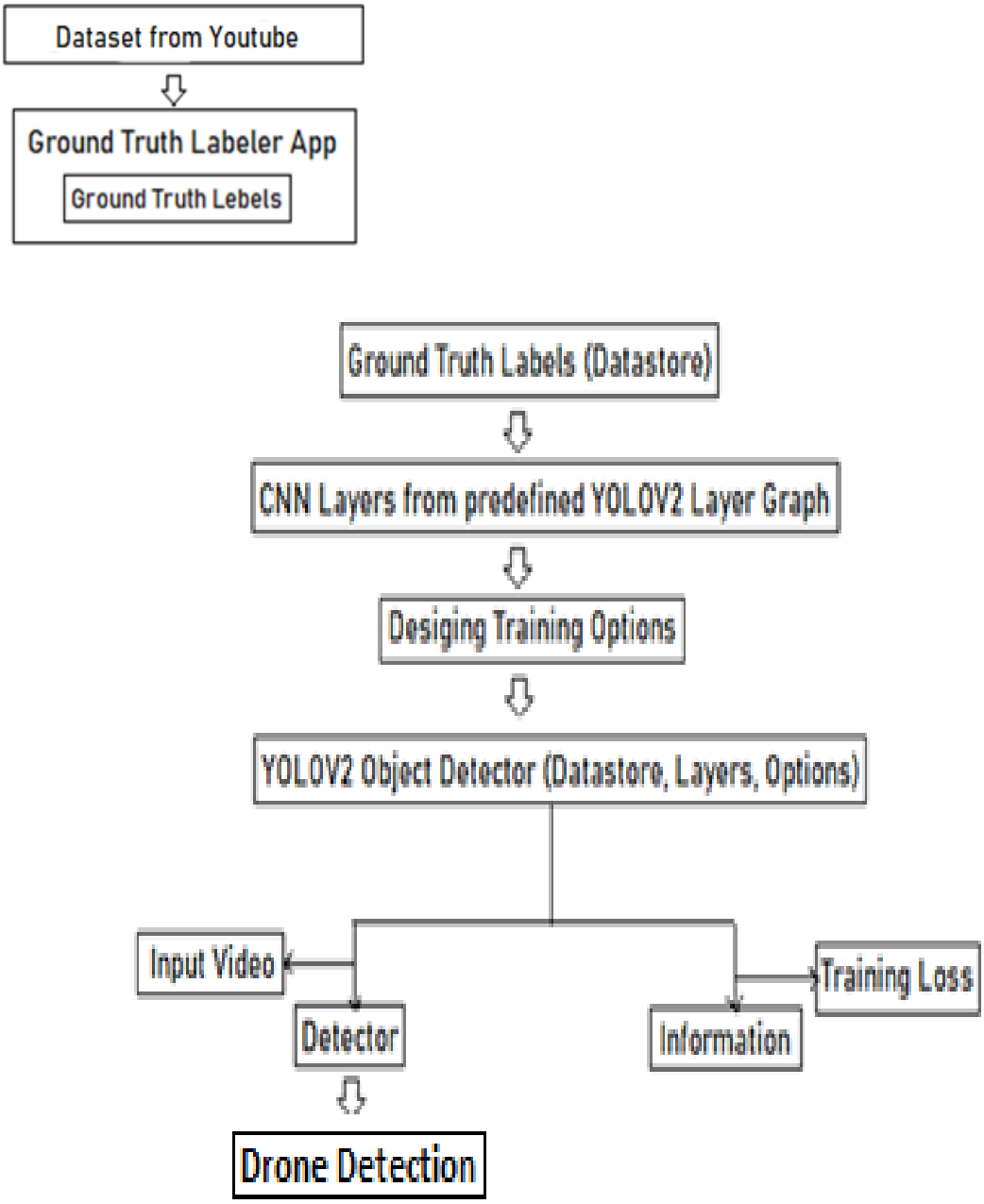

Fig7: Drone Detection using YOLO algorithm (Deep Learning) Block Diagram

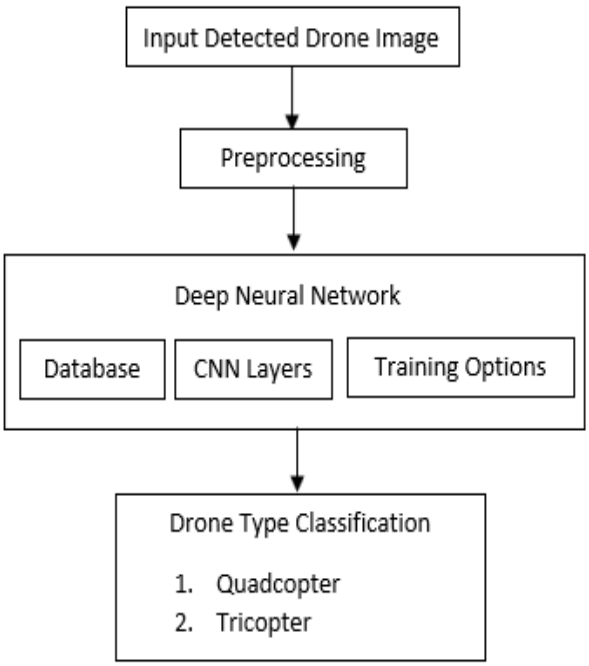

Fig8: **Drone Type Classification Using Convolutional Neural Network (Deep Learning) Block Diagram**

1	'input'	Image Input	128x128x3 images
2	'conv_1'	Convolution	16 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
3	'BN1'	Batch Normalization	Batch normalization
4	'relu_1'	ReLU	ReLU
5	'maxpool1'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv_2'	Convolution	32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
7	'BN2'	Batch Normalization	Batch normalization
8	'relu_2'	ReLU	ReLU
9	'maxpool2'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv_3'	Convolution	64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
11	'BN3'	Batch Normalization	Batch normalization
12	'relu_3'	ReLU	ReLU
13	'maxpool3'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
14	'conv_4'	Convolution	128 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
15	'BN4'	Batch Normalization	Batch normalization
16	'relu_4'	ReLU	ReLU
17	'yolov2Conv1'	Convolution	128 3x3 convolutions with stride [1 1] and padding 'same'
18	'yolov2Batch1'	Batch Normalization	Batch normalization
19	'yolov2Relu1'	ReLU	ReLU
20	'yolov2Conv2'	Convolution	128 3x3 convolutions with stride [1 1] and padding 'same'
21	'yolov2Batch2'	Batch Normalization	Batch normalization
22	'yolov2Relu2'	ReLU	ReLU
23	'yolov2ClassConv'	Convolution	24 1x1 convolutions with stride [1 1] and padding [0 0 0 0]
24	'yolov2Transform'	YOLO v2 Transform Layer.	YOLO v2 Transform Layer with 4 anchors.
25	'yolov2OutputLayer'	YOLO v2 Output	YOLO v2 Output with 4 anchors.

Fig9: **Layers of YOLO Model**

In YOLOv2 the details of each block in the visualization can be seen by hovering over the block. Each Convolution block has the BatchNorm normalization and then Leaky Relu activation except for the last Convolution block. YOLO divides the input image into

an **S×S** grid. Each grid cell predicts only **one** object. For example, the yellow grid cell below tries to predict the "person" object whose center (the blue dot) falls inside the grid cell. Each grid cell predicts a fixed number of boundary boxes. In this example, the yellow grid cell makes two boundary box predictions (blue boxes) to locate where the person is. However, the one-object rule limits how close detected objects can be.

For each grid cell,

- it predicts **B** boundary boxes and each box has one **box confidence score**,
- it detects **one** object only regardless of the number of boxes B,
- it predicts **C conditional class probabilities** (one per class for the likeliness of the object class).

The boundary boxes contain box confidence score. The confidence score reflects how likely the box contains an object (**objectless**) and how accurate is the boundary box. We normalize the bounding box width *w* and height *h* by the image width and height. *x* and *y* are offsets to the corresponding cell. Hence, *x*, *y*, *w* and *h* are all between 0 and 1. Each cell has 20 conditional class probabilities. The **conditional class probability** is the probability that the detected object belongs to a particular class (one probability per category for each cell). The class confidence score for each prediction box is computed as:class confidence score = box confidence score * conditional class probability.It measures the confidence on both the classification and the localization (where an object is located). We may mix up those scoring and probability terms easily. Here are the mathematical definitions for your future reference.

$$\begin{aligned}\text{box confidence score} &\equiv P_r(object) \cdot IoU \\ \text{conditional class probability} &\equiv P_r(class_i|object) \\ \text{class confidence score} &\equiv P_r(class_i) \cdot IoU \\ &= \text{box confidence score} \times \text{conditional class probability}\end{aligned}$$

where

$P_r(object)$ is the probability the box contains an object.
IoU is the IoU (intersection over union) between the predicted box and the ground truth.
$P_r(class_i|object)$ is the probability the object belongs to $class_i$ given an object is presence.
$P_r(class_i)$ is the probability the object belongs to $class_i$

YOLO predicts multiple bounding boxes per grid cell. To compute the loss for the true positive, we only want one of them to be **responsible** for the object. For this purpose, we select the one with the highest IoU (intersection over union) with the ground truth. This strategy leads to specialization among the bounding box predictions. Each prediction gets better at predicting certain sizes and aspect ratios.

YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of:

- the **classification loss**.
- the **localization loss** (errors between the predicted boundary box and the ground truth).
- the **confidence loss** (the objectness of the box).

Classification loss

If *an object is detected,* the classification loss at each cell is the squared error of the class conditional probabilities for each class:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

where

$\mathbb{1}_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

We do not want to weight absolute errors in large boxes and small boxes equally. i.e. a 2-pixel error in a large box is the same for a small box. To partially address this, YOLO predicts the square root of the bounding box width and height instead of the width and height. In addition, to put more emphasis on the boundary box accuracy, we multiply the loss by λ*coord* (default:5).

4.1 CNN:

When it comes to Machine Learning, artificial neural network performs really well. Artificial Neural Networks are used in various classification task like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural Network. In this we are going to build basic building block for CNN. A convolutional neural network can consist of one or multiple convolutional layers. The number of convolutional layers depends on the amount and complexity of the data.

Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network, there are three types of layers:

1. Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels in case of an image).

2. **Hidden Layer:** The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer

with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.

3**. Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we back propagate into the model by calculating the derivatives. This step is called back propagation which basically is used to minimize the loss.A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. A convolution is essentially sliding a filter over the input. One helpful way to think about convolutions is this quote from Dr Prasad Samarakoon: "A convolution can be thought as "looking at a function's surroundings to make better/accurate predictions of its outcome." Rather than looking at an entire image at once to find certain features it can be more effective to look at smaller portions of the image. The most common use for CNNs is image classification, for example identifying satellite images that contain roads or classifying hand written letters and digits. There are other quite mainstream tasks such as image segmentation and signal processing, for which CNNs perform well at. CNNs have been used for understanding in Natural Language Processing (NLP) and speech recognition, although often for NLP Recurrent Neural Nets (RNNs) are used.

A CNN can also be implemented as a U-Net architecture, which are essentially two almost mirrored CNNs resulting in a CNN whose architecture can be presented in a U shape. U-nets are used where the output needs to be of similar size to the input such as segmentation and image improvement. Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map. These are linear transformations; each convolution is a type of affine function. In computer vision the input is often a 3 channel RGB image. For simplicity, if we take a greyscale image that has one channel (a two-dimensional matrix) and a 3x3 convolutional kernel (a two-dimensional matrix). The kernel

strides over the input matrix of numbers moving horizontally column by column, sliding/scanning over the first rows in the matrix containing the images pixel values. Then the kernel strides down vertically to subsequent rows.

Padding:

To handle the edge pixels there are several approaches:

- Losing the edge pixels
- Padding with zero value pixels
- Reflection padding

Reflection padding is by far the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image. For a 3x3 kernel, one pixel needs to be added around the outside, for a 7x7 kernel then three pixels would be reflected around the outside. The pixels added around each side is the dimension, halved and rounded down.Traditionally in many research papers, the edge pixels are just ignored, which loses a small proportion of the data and this gets increasing worse if there are many deep convolutional layers. For this reason, I could not find existing diagrams to easily convey some of the points here without being misleading and confusing stride 1 convolutions with stride 2 convolutions.With padding, the output from a input of width w and height h would be width w and height h (the same as the input with a single input channel), assuming the kernel takes a stride of one pixel at a time.

Strides:

It is common to use a stride two convolution rather than a stride one convolution, where the convolutional kernel strides over 2 pixels at a time, for example our 3x3 kernel would start at position (1, 1), then stride to (1, 3), then to (1, 5) and so on, halving the size of the output channel/feature map, compared to the convolutional kernel taking strides of one. With padding, the output from an input of width w, height h and depth 3 would be the ceiling of width w/2, height h/2 and depth 1, as the kernel outputs a single summed output from each stride.

For example, with an input of 3x64x64 (say a 64x64 RGB three channel image), one kernel taking strides of two with padding the edge pixels, would produce a channel/feature map of

32x32.The first step of creating and training a new convolutional neural network (Convnet) is to define the network architecture. This topic explains the details of Convent layers, and the order they appear in a ConvNet. For a complete list of deep learning layers and how to create them, see List of Deep Learning Layers. To learn about LSTM networks for sequence classification and regression, see Long Short-Term Memory Networks. To learn how to create your own custom layers, see Define Custom Deep Learning Layers. The network architecture can vary depending on the types and numbers of layers included.

Image Input Layer:

Create an image input layer using image input layer. An image input layer inputs images to a network and applies data normalization. Specify the image size using the `input Size` argument. The size of an image corresponds to the height, width, and the number of color channels of that image. For example, for a grayscale image, the number of channels is 1, and for a color image it is 3.

Convolutional layer:

A 2-D convolutional layer applies sliding convolutional filters to the input. Create a 2-D convolutional layer using convolution2dLayer. The convolutional layer consists of various components. Filters and Stride a convolutional layer consist of neurons that connect to sub regions of the input images or the outputs of the previous layer. The layer learns the features localized by these regions while scanning through an image. When creating a layer using the convolution2dLayer function, you can specify the size of these regions using the filter Size input argument.

Dilated Convolution:

A dilated convolution is a convolution in which the filters are expanded by spaces inserted between the elements of the filter. Specify the dilation factor using the 'Dilation Factor' property. Use dilated convolutions to increase the receptive field (the area of the input which the layer can see) of the layer without increasing the number of parameters or computation. The layer expands the filters by inserting zeros between each filter element. The dilation factor determines the step size for sampling the input or equivalently the up-sampling factor of the filter. It corresponds to an effective filter size of (Filter Size – 1). * Dilation Factor + 1. For example, a 3-by-3 filter with the dilation factor [2 2] is equivalent to a 5-by-5 filter with

zeros between the elements. This image shows a 3-by-3 filter dilated by a factor of two scanning through the input. The lower map represents the input and the upper map represents the output.

Feature Maps

As a filter moves along the input, it uses the same set of weights and the same bias for the convolution, forming a *feature map*. Each feature map is the result of a convolution using a different set of weights and a different bias. Hence, the number of feature maps is equal to the number of filters. The total number of parameters in a convolutional layer is ((*h***w***c* + 1)**Number of Filters*), where 1 is the bias.

Zero Padding

You can also apply zero padding to input image borders vertically and horizontally using the 'Padding' name-value pair argument. Padding is rows or columns of zeros added to the borders of an image input. By adjusting the padding, you can control the output size of the layer. This image shows a 3-by-3 filter scanning through the input with padding of size 1. The lower map represents the input and the upper map represents the output.

Rectified Linear Unit (ReLU):

A Rectified Linear Unit is used as a non-linear activation function. A ReLU says if the value is less than zero, round it up to zero. Create a ReLU layer using reluLayer. A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero. Convolutional and batch normalization layers are usually followed by a nonlinear activation function such as a rectified linear unit (ReLU), specified by a ReLU layer. A ReLU layer performs a threshold operation to each element, where any input value less than zero is set to zero, that is, The ReLU layer does not change the size of its input. There are other nonlinear activation layers that perform different operations and can improve the network accuracy for some applications. For a list of activation layers, see Activation Layers.

Batch normalization layer:

Batch normalization has the benefits of helping to make a network output more stable predictions, reduce over fitting through regularization and speeds up training by an order of

magnitude. Batch normalization is the process of carrying normalization within the scope activation layer of the current batch, subtracting the mean of the batch's activations and dividing by the standard deviation of the batch's activations. Create a batch normalization layer using batch Normalization Layer. A batch normalization layer normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers.The layer first normalizes the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. Then, the layer shifts the input by a learnable offset β and scales it by a learnable scale factor γ. β and γ are themselves learnable parameters that are updated during network training. Batch normalization layers normalize the activations and gradients propagating through a neural network, making network training an easier optimization problem. To take full advantage of this fact, you can try increasing the learning rate. Since the optimization problem is easier, the parameter updates can be larger and the network can learn faster. You can also try reducing the L2 and dropout regularization. With batch normalization layers, the activations of a specific image during training depend on which images happen to appear in the same mini-batch. To take full advantage of this regularizing effect, try shuffling the training data before every training epoch. To specify how often to shuffle the data during training, use the 'Shuffle' name-value pair argument of training Options.

4.2 Max and Average Pooling Layers:

A max pooling layer performs down-sampling by dividing the input into rectangular pooling regions, and computing the maximum of each region. Create a max pooling layer using maxPooling2dLayer. An average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region. Create an average pooling layer using average Pooling2dLayer. Pooling layers follow the convolutional layers for down-sampling, hence, reducing the number of connections to the following layers. They do not perform any learning themselves, but reduce the number of parameters to be learned in the following layers. They also help reduce over fitting.A max pooling layer returns the maximum values of rectangular regions of its input. The size of the rectangular regions is determined by the pool Size argument of maxPoolingLayer. For example, if pool Size equals [2, 3], then the layer returns the maximum value in regions of

height 2 and width 3. An average pooling layer outputs the average values of rectangular regions of its input. The size of the rectangular regions is determined by the pool Size argument of averagePoolingLayer. For example, if pool Size is [2, 3], then the layer returns the average value of regions of height 2 and width 3.Pooling layers scan through the input horizontally and vertically in step sizes you can specify using the 'Stride' name-value pair argument. If the pool size is smaller than or equal to the stride, then the pooling regions do not overlap. For non-overlapping regions (Pool Size and Stride are equal), if the input to the pooling layer is n-by-n, and the pooling region size is h-by-h, then the pooling layer down-samples the regions by h [6]. That is, the output of a max or average pooling layer for one channel of a convolutional layer is n/h-by-n/h. For overlapping regions, the output of a pooling layer is (Input Size – Pool Size + 2*Padding)/Stride + 1.

Dropout Layer

Create a dropout layer using dropout layer. A dropout layer randomly sets input elements to zero with a given probability. At training time, the layer randomly sets input elements to zero given by the dropout mask rand (size(X)) <Probability, where X is the layer input and then scales the remaining elements by 1/ (1-Probability). This operation effectively changes the underlying network architecture between iterations and helps prevent the network from over fitting. A higher number results in more elements being dropped during training. At prediction time, the output of the layer is equal to its input. Similar to max or average pooling layers, no learning takes place in this layer.

Fully Connected Layer:

Create a fully connected layer using fully connected layer. A fully connected layer multiplies the input by a weight matrix and then adds a bias vector. The convolutional (and down-sampling) layers are followed by one or more fully connected layers. As the name suggests, all neurons in a fully connected layer connect to all the neurons in the previous layer. This layer combines all of the features (local information) learned by the previous layers across the image to identify the larger patterns. For classification problems, the last fully connected layer combines the features to classify the images. This is the reason that the output Size argument of the last fully connected layer of the network is equal to the number of classes of the data set. For regression problems, the output size must be equal to the number of response variables.

You can also adjust the learning rate and the regularization parameters for this layer using the related name-value pair arguments when creating the fully connected layer. If you choose not to adjust them, then train Network uses the global training parameters defined by the training Options function. For details on global and layer training options, and train the neural network. A fully connected layer multiplies the input by a weight matrix *W* and then adds a bias vector *b*. If the input to the layer is a sequence (for example, in an LSTM network), then the fully connected layer acts independently on each time step. For example, if the layer before the fully connected layer outputs an array *X* of size *D*-by-*N*-by-*S*, then the fully connected layer outputs an array *Z* of size output Size-by-*N*-by-*S*. At time step *t*, the corresponding entry of *Z* is, where denotes time step *t* of *X*.

Output Layers:

Softmax and Classification Layers:

A softmax layer applies a softmax function to the input. Create a softmax layer using softmax layer. A classification layer computes the cross-entropy loss for multi-class classification problems with mutually exclusive classes. Create a classification layer using classification Layer. For classification problems, a softmax layer and then a classification layer must follow the final fully connected layer. The softmax function is also known as the normalized exponential and can be considered the multi-class generalization of the logistic sigmoid function. For typical classification networks, the classification layer must follow the softmax layer. In the classification layer, train Network takes the values from the softmax function.

Regression Layer:

Create a regression layer using the regression layer. A regression layer computes the half-mean-squared-error loss for regression problems. For typical regression problems, a regression layer must follow the final fully connected layer. For a single observation, the mean-squared-error is given by: where *R* is the number of responses, t_i is the target output, and y_i is the network's prediction for response *i*. For image and sequence-to-one regression networks, the loss function of the regression layer is the half-mean-squared-error of the predicted responses, not normalized by *R*: For image-to-image regression networks, the loss function of the regression layer is the half-mean-squared-error of the predicted responses for each pixel, not normalized by *R*.

Do you know what's the term "Training" means in Artificial Intelligence (AI)...?

Thoughts about AI...

Self-driving cars and robots always comes to mind as most people talk of artificial intelligence (AI). Yet others don't stop talking about how AI operates to bring those conveniences to life. AI is developed from machine learning (ML) and deep learning by being fed vast volumes of data to gain information from data and automate tasks at a scale. The computers are learning how to interpret and predict — to "think" as much as possible like humans.

And how long does it take to get trained with AI? It might take hours, weeks or even longer. The answer is really due to factors such as hardware, optimization, number of neural network layers, size of your dataset and more.

Let's discuss how AI training functions, and what it takes to get a better understanding of this.

Let's see something about Machine Learning & Deep learning…

Machine learning is a branch of AI that helps computer systems to learn and develop automatically, without being controlled by a person. Machine learning uses algorithms which discover patterns, then modify itself as it is transferred to more data — it adjusts according to the data to which it was attached, much like a child who learns from his experience/observations.

Deep learning is progressively a more advanced approach to machine learning that uses artificial neural networks to simulate the human brain while processing data. The computers learn, making decisions based on continuous processing and feedback, through positive and negative reinforcement. Examples of deep learning applications include image recognition technology and voice recognition assistants.

Deep learning depends on its "deep" neural networks and on multiple layers. Every neuron on the network consists of a mathematical function for transforming and analyzing data as an output. To make accurate predictions, the machine learns how to calculate the value of each relation between the neurons. Deep learning is extremely effective in overcoming difficult problems with different quantities.

In machine learning, relevant features are extracted from the images, whereas deep learning is an end-to-end process where features are automatically extracted. Deep learning also has the ability to scale with data as networks improve and when the amount of data increases. However, with this increased input of data, comes an increased computing power and training time.

4.3 AI Training means…?

The actual AI training phase includes three steps: training, validation, and testing. It is being trained by feeding data into the computer system to produce a specific prediction with each cycle. The parameters can be adjusted each time to make sure the predictions become more accurate with each training step.

It then verifies the algorithm by running validation data against the trained model. At this stage, new variables might need to be modified to improve the algorithm. When the validation stage has finished, the device can be checked using real-world data which do not have tags or labels. This is the time to see if the algorithm is ready for its intended use.To train a deep network from scratch, you gather a very large labeled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have a large number of output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train,Most deep learning applications use the transfer learning approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as Alexnet or GoogLeNet, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours.Feature extraction is another method of deep learning. It involves extracting a layer from the neural network that is tasked with learning a certain feature from the images and using that feature as an addition to another machine learning model.

Requirements that can influence the AI training process…

Some of the requirements that can influence the AI training process are data, hardware, software, knowledge.

- **Data:**

Because data is an integral part of the algorithm design, it is important to have a clean and correctly labelled dataset. It follows that by inputting accurate data into the algorithm you will have accurate outputs, resulting in a more effective and timely training process. For example, a dataset may include millions of images and thousands of hours of video for driverless car design.

- **Training data** is the part of your data which you use to help your machine learning model make predictions. Your model will be run on this set of data exhaustively, churning out results which your data scientists can use to develop your algorithm. It's the largest part of your overall dataset, comprising around 70-80% of your total data used in the project.
- **Validation data** is a second set of data, also containing input and target information, which the machine learning model has never seen before. By running the model on validation data, it's possible to see whether it can correctly identify relevant new examples. This is where it's possible to discover new values that are impacting the process. Another common problem often identified during validation is overfitting, where the AI has been wrongly trained to identify examples that are too specific to the training data. As you can imagine, after validation, data scientists will often go back to the training data and run through it again, tweaking values and hyperparameters to make the model more accurate.

  Note: Some of the issues like overfitting, misclassifications, increase in execution time, decrease in accuracy etc., are mainly depends on size of the dataset, system's processor, designed network (layers, options and dataset). The trained network is only capable to produce dataset according to the data that had trained i.e., if we trained with class of animals then it can only able to detection that class. No, other classes should not be used for testing that network.

- **Testing data** comes into play after a lot of improvement and validation. While validation data has tags and target information left on as training wheels, testing data provides no help to the model at all. Asking the model to make predictions based on this data is meant to test whether it will work in the real world, where it won't have helpful tags scattered around. The final test is the moment of truth for the model, to see if all the hard work has paid off.

- **Hardware:**

Deep learning demands massive amounts of computing power. This means combining high-performance Graphics Processing Units (GPUs) with clusters or cloud computing to reduce weeks-to-hours of deep learning time. Since AI training can be performed in parallel, setting up a network where you can train on several GPUs, or helping to accelerate training in a cluster.

- **Software:**

Each cloud provider has its own automated machine-learning software when renting computing infrastructure, such as Microsoft's Machine Learning Studio, Google's Cloud AutoML and AWS SageMaker. Some may also use deep learning tools to design training models, such as MathWorks's MATLAB, Google's TensorFlow and Pytorch.

- **Knowledge:**

- There is a worldwide shortage of skilled AI developers, with an estimate of less than 10,000 individuals who have the necessary AI research skills. AI experts are so widely sought after they are paid wherever they are. Developers need not only a Ph.D. in computer science but also experience in various fields such as MATLAB programming, C++ programming, STL and knowledge in physics or biological sciences. While validation data has tags and target information left on as training wheels, testing data provides no help to the model at all. Asking the model to make predictions based on this data is meant to test whether it will work in the real world, where it won't have helpful tags scattered around. The final test is the moment of truth for the model, to see if all the hard work has paid off. such as Microsoft's Machine Learning Studio, Google's Cloud AutoML and AWS SageMaker. Some may also use deep learning tools to design training models, such as MathWorks's MATLAB, Google's TensorFlow and Pytorch. By running the model on validation data, it's possible to see whether it can correctly identify relevant new examples. This is where it's possible to discover new values that are impacting the process. Another common problem often identified during validation is overfitting, where the AI has been wrongly trained to identify examples that are too specific to the training data. As you can imagine, after validation, data scientists will often go back to the training data and run through it again, tweaking values and hyperparameters to make the model more accurate.

CHAPTER 5
ADVANTAGES AND APPLICATIONS

Advantages:

- YOLO takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes.
- The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation.
- The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.
- CNN is also computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universally attractive.

Applications:

- Military. Probably the oldest, most well-known and controversial use of drones is in the military.
- Delivery.
- Emergency Rescue.
- Agriculture.
- Outer Space.
- Wildlife and Historical Conservation.
- Medicine.
- Photography.

CHAPTER 6

RESULTS

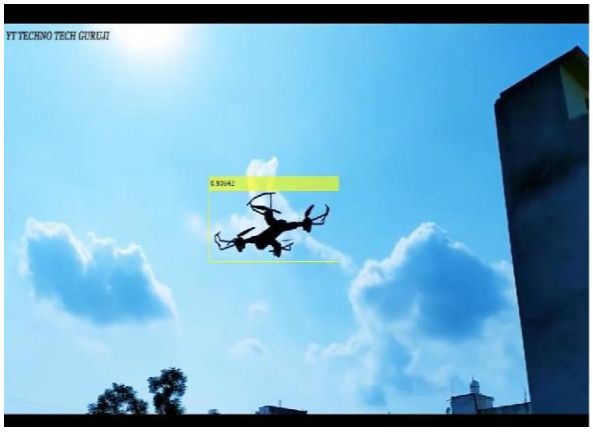

***Fig10*: Input Video**

***Fig11:* Drone Detection Video**

***Fig12:* Detected Drone**

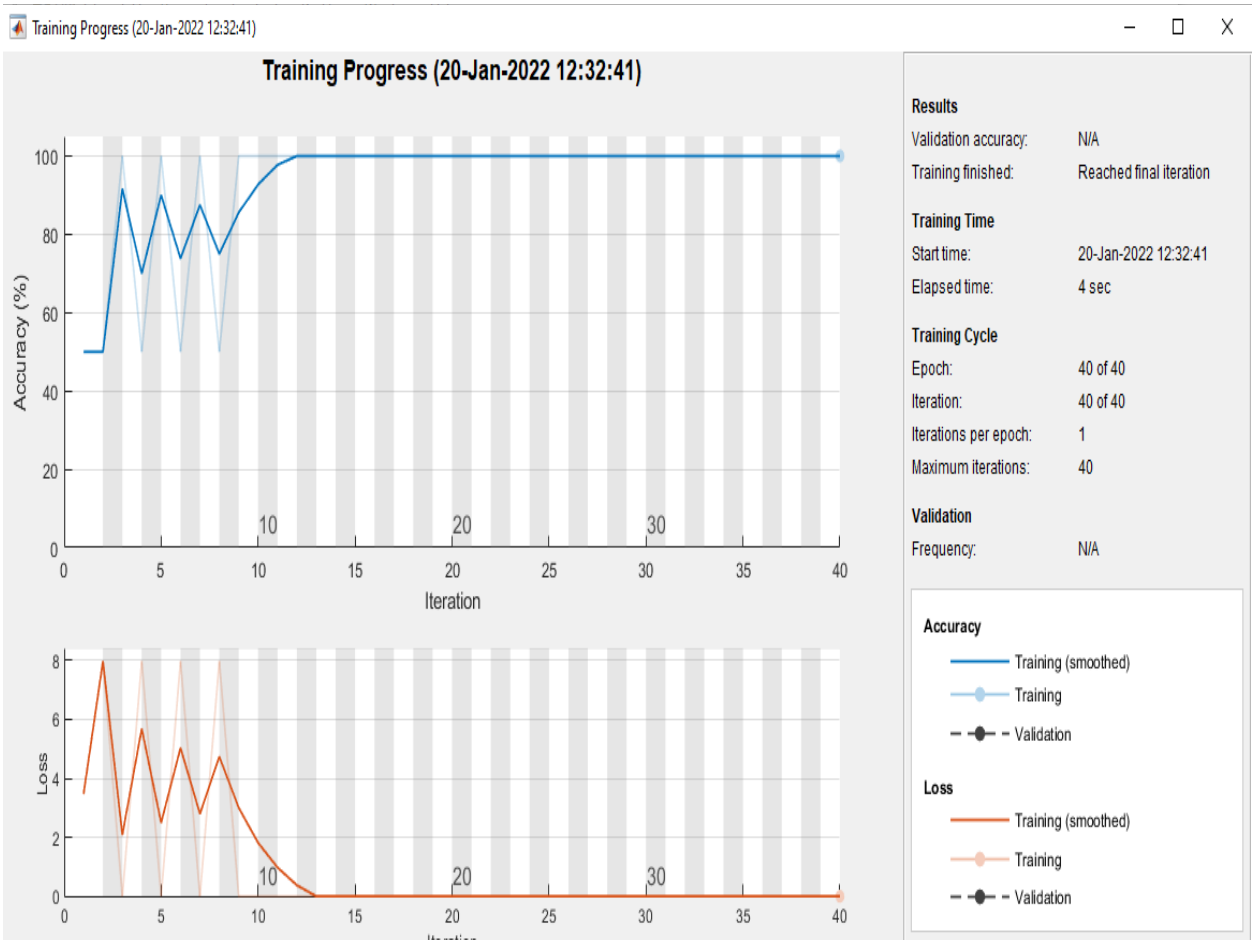

Fig13: **Training Progress**

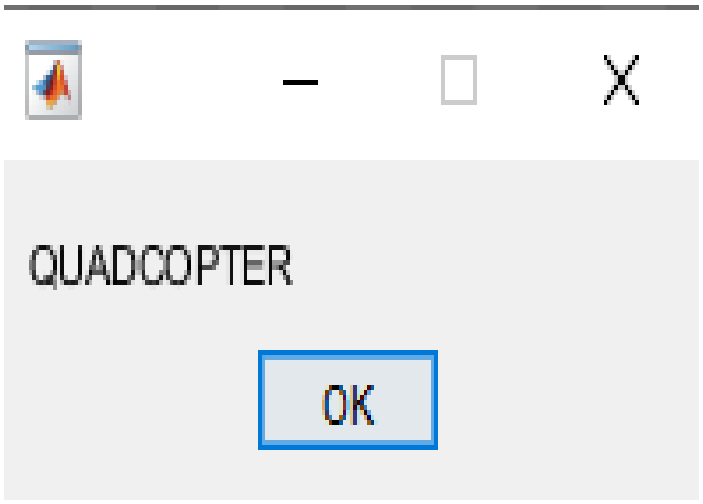

Fig14: **Classified Drone Type**

CONCLUSION

In this study, we proposed an approach to detect drones and classify the drone types using deep learning. The algorithm was depicted in two steps. A YOLO v2 model was trained to detect drones and convolutional neural networks (CNN) was trained to classify drone types (Tricopter or quadcopter). The trained YOLO v2 achieved a very low in training loss and accuracy with nearly 80% for both detection and classification tasks. In this article, we present an autonomous drone detection, tracking and identification system based on optics and deep learning. The experimental results show that the accuracy of the drone detection based on deep learning method is higher than the machine learning method. Object detection using deep learning provides a fast and accurate means to predict the location of an object in an image. Deep learning is a powerful machine learning technique in which the object detector automatically learns image features required for detection tasks. A highly integrated transceiver chip as RF end of radar receiver directly mix downlink signal to baseband while filtering uplink signal out. Then, radar receiver transforms baseband signal-to-digital data for PC processing. The signal processing pattern of our radar is similar to that of CP-OFDM and pulse-based radar, owing to CP-OFDM defined frame structure and idle state of eNB. Experimental results validate the feasibility and practicability of our radar system for low-altitude drone monitoring. In the near future, we will focus on promoting system performance by adding more channels and achieving estimation of arrival angle and track of UAVs.

FUTURE SCOPE

In this work, we are only willing to detect and classify the drones only. In future it can be extended to predict the location of the drone using/interfacing it to embedded systems/hardware devices.

REFERENCES

[1] Y. Yu, D. Barthaud, B. A. Price, A. K. Bandara, A. Zisman and B. Nuseibeh, "LiveBox: A Self-Adaptive Forensic-Ready Service for Drones," in IEEE Access, vol. 7, pp. 148401-148412, 2019, doi: 10.1109/ACCESS.2019.2942033.

[2] D. Lee, W. Gyu La and H. Kim, "Drone Detection and Identification System using Artificial Intelligence," 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018, pp. 1131-1133, doi: 10.1109/ICTC.2018.8539442.

[3] C. Liu, Y. Tao, J. Liang, K. Li and Y. Chen, "Object Detection Based on YOLO Network," 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), 2018, pp. 799-803, doi: 10.1109/ITOEC.2018.8740604.

[4] N. Jmour, S. Zayen and A. Abdelkrim, "Convolutional neural networks for image classification," 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), 2018, pp. 397-402, doi: 10.1109/ASET.2018.8379889.

[5] Xiaowu Sun, Lizhen Liu, Hanshi Wang, Wei Song and Jingli Lu, "Image classification via support vector machine," 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), 2015, pp. 485-489, doi: 10.1109/ICCSNT.2015.7490795.

[6] B. Choi and D. Oh, "Classification of Drone Type Using Deep Convolutional Neural Networks Based on Micro- Doppler Simulation," 2018 International Symposium on Antennas and Propagation (ISAP), 2018, pp. 1-2.

[7] Gunnarsson F., Johansson M., Furuskär A. *et al.*: 'Downtilted base station antennas - a simulation model proposal and impact on HSPA and LTE performance'. Vehicular Tech. Conf., Calgary, BC, Canada, September 2008, pp. 1 – 5

[8] Qualcomm: ' LTE TDD- the global solution for unpaired spectrum', 2014, p. 6