Chair of Distributed Information Systems

Detection and Classification of Drone

Masterarbeit von

Syed Ayaz

1. Prüfer	2. Prüfer
Prof.Dr.Habil Mario Döller	Prof. Dr. Harald Kosch

October 24, 2021

Contents

Abstract

Acknowledgments

List of Acronyms

FAA Federal Aviation Administration

UAV Unmanned Aerial Vehicle

ANN Artificial Neural Network

CNN convolutional neural network

TPU Tensor Processing Units

CPU central processing unit

GPU graphics processing unit

ReLU Rectified Linear Unit

ASIC application-specific integrated circuits

SIFT Scale-invariant feature transform

HOG Histogram of oriented Gradients

YOLO You Only Look Once

CUDA Compute Unified Device Architecture

API application programming interface

NLP natural language processing

OpenCV Open Source Computer Vision

VGG Visual Geometry Group

GUI graphical user interface

CLI command line interface

RBDNA Rotor-Based Drone-net Advance

APdI application programming interface

List of Figures

List of Tables

1 Introduction

For many years, drones were primarily used by the military and had yet to come into the hands of the common public. In the year 2016, it was reported that many companies had been exempted by the FAA(Federal Aviation Administration) to operate drones in the United States. This was a major breakthrough for drone enthusiasts and helped to commercialize the drone industry, which in turn led to the extensive manufacturing and availability of drones for public and personal use. In the last few years, with the advent of technologies and the increasing investment in drone technology, the cost of production of the basic building blocks like motors, batteries, microcontrollers, and sensors have been driven down by a large factor. This has given rise to the production of cheaper drones that can be used in many capacities, including security, survey, agriculture, photography, transportation, recreation, etc. These factors have led to a massive surge in the number of drones that have been manufactured and registered

Unmanned Aerial Vehicle (UAV)s and drones are in common use these days. Drones are used in a variety of applications, we have domestic, industrial, commercial, and even defense use cases for these UAV. We might need to detect these drones for various purposes; for example, there is a risk that drones might collide with airplanes, sensitive infrastructure such as Oil and Gas plants also need some sort of drone detection as a preventive measure, we need to be aware and safe from domestic drones to avoid them interfering with our privacy. Detection of drones is a complex task because of various reasons. We need to detect drones in real-time which requires complex and accurate detection methods, the atmosphere is not always clear or suitable for the detection of drones, and detection methods might confuse birds, airplanes, clouds with drones and UAV as well.

1.1 Problem Statement

According to the Federal Aviation Administration (FAA)'s prediction, the total estimated number of drones that will have been registered in the U.S. by 2023 is between 1.3 million and 1.7 million drones.[Adm] Similarly, Insider Intelligence expects the total amount of global drone shipments to reach about 2.4 million in 2023. [1]

However, due to the technology being very economical and thus, its widespread use, it has uncovered some of its more negative sides. There have been reports of potential misuse of the technology in regards to invasion of privacy, terrorism, unauthorized surveillance and exploration, smuggling, and other similar incidents. For example, an incident was reported in Japan, where a drone landed on the roof of the residence of the Prime Minister. The alarming part was that the drone was carrying a radioactive substance.[15] There have also been incidents of drones that crashed, a big event which has led to the legislation and restriction of using drones in certain areas that may be tourist spots or sensitive government facilities. As the number of drones increases, the possibility of the incidents, such as those mentioned previously, continues to increase in the same manner.

Now, to prevent these situations, systems are being developed to detect and classify drones using the image processing and recognition capabilities of deep learning and neural network technology. In this study, we aim to detect and classify the drones from a video with the help of deep learning technologies. We well develop a system that can detect and classify drones within the frames of a video. However, the detection will be limited to the multi-rotor drones. Our aim will be to detect the drones using an annotated dataset, where the data will be classified into two types; rotor and drone; to increase the accuracy of detection. We will use the YOLO model for detection. During the process, we will be able to analyze the accuracy of our results and work towards increasing that accuracy to the maximum possible value.

1.2 Research Questions

1. Are there any dataset available with rotor image annotation?

[1] Check `https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts` Business Insider

According to our research, a dataset with drone images containing rotor annotation could not be found. Due to the unavailability of such a dataset, we are compelled to annotate an existing drone dataset with two types of classifications, one for the drone and another for the rotor. This will be achieved with the help of an annotation software called "LabelImg".

2. What effect does the annotation of drone images based on rotor have on the accuracy of detection and classification of drones in time video feed?

   Annotation of the drone images is a major, and sometimes, the most time consuming part of dataset preparation. It can help the model to properly learn about the features and characteristics of the object. The success of the model is very much, dependent on correct annotations of the data. It is also very important because a slight error in the annotation can cause the model to learn incorrect information about the object which can affect the accuracy of the detection to a large degree. Therefore, annotation will help the model to detect a drone with greater accuracy in the presence of other objects such as a bird, plane, etc. The classification of the drone on the basis of the rotor and a drone will surely improve the accuracy of detection.

2 State of the Art

2.1 Machine Learning: A Brief Overview

The advent of machine learning algorithms, in many fields of research, has made it a driving force in the technological advancements of the present era. It has been utilized for modern applications like the self-driving cars, speech recognition, natural language processing, traffic predictions, and personalized recommendations about products as well as the popular field of computer vision. It is a continuously developing field and is being adopted all around the world. We will be going through the background and fundamentals of machine learning.

One of the most important concepts of machine learning is that it is loosely based on the model of the interactions of the human brain. Machine learning has been widely used for the formulation of mathematical models. These models are very difficult to formulate otherwise, and thus machine learning provides a way to solve this problem. We know that conventionally, computer programs are basically programmed by a human, who defines a set of rules for performing a certain task when given a certain input. This concept has been abolished by machine learning by providing algorithms that can learn to make the rules from a given set of data. This replaces the need for a human to manually program the rules in order to perform a task correctly.

2.1.1 Feature extraction

In recent years, we have seen a massive and widespread increase in the use of technology, which has resulted in an enormous amount of data that has to be processed in the best possible manner. Processing such large amounts of data, manually, is impossible which is

why machine learning algorithms are becoming more popular and relevant every passing day.

Modern-day machine learning problems depend on large amounts of data that cannot be immediately processed and learned by algorithms. This can be solved by preprocessing that data to extract and ascertain representations that are simple and useful for the algorithm. This step is called feature extraction. The criterion for detecting such representations are usually programmed manually, however, we are actively moving towards the automation of the feature extraction process.

2.1.2 Problem Classification

The two main types of learning algorithms that are used in the implementation of machine learning are the supervised and the unsupervised learning algorithms. Supervised learning takes place when the algorithm is provided with data or images that have been classified and labeled manually. This type of data is called training data, which is specifically used for training the model. Using these images, the algorithm 'learns' to predict those labels from data or images that are similar in nature but different from those used in the training of the algorithm; this type of data is called testing data. In the case of unsupervised learning, the aim is to make the algorithm learn valuable characteristics of the data when no information is provided about the correct outcomes.

In machine learning, we mainly encounter two types of tasks namely 'classification' and 'regression'. Classification tasks are performed when we want the algorithm to accurately predict and classify data based on the training data. On the other hand, in a regression task, the algorithm is used for predicting an output that is continuous, as opposed to predicting the class of the data.

2.1.3 Generalization and Loss Function

We should also know that the datasets that are used for training the model cannot possibly contain every single occurrence of the input. Thus, the algorithm should be able to provide accurate results when tested on similar but unseen data, by generalization. When a model has been trained too much on the training data, it will result in erroneous predictions due to the model being unable to generalize. This is known as overfitting.

It also occurs when the training data is not sufficient for the algorithm and it can cause the algorithm to model those parts and details that are not essential or can be regarded as noise.

The effectiveness of the algorithm can be measured by analyzing the errors through a function called the "loss function". This function is used to measure the distance of a predicted value from its actual value. The output of a loss function will be higher when the predictions of the algorithm are inaccurate and in the same way, lower output for an accurately predicting algorithm. This output will be used to optimize the algorithm in a way that the algorithm becomes more accurate in its prediction [Bro20].

2.2 Foundation of Neural Networks

One of the most popular models of machine learning is Artificial Neural Network (ANN) or simply Neural Networks. This model is loosely based on the model of the human brain and is a collection of connected nodes which are called artificial neurons [RM07]. We will be discussing the basics of neural networks before getting into the more complex case of neural networks that is the convolutional neural network (CNN).

In simple terms, we can say that a neural network, is a network consisting of a very large number of nodes or neurons that are interconnected and capable of communicating with each other utilizing the connections between them, called edges. It is not correct to state that the working of neural networks resembles that of our brains, but it is true that these networks were inspired by the networks of neurons in our brains. Neural networks aim to identify the fundamental relationships and representations that exist in a collection of data with the help of neurons []. The neurons adjust themselves as the input to the network changes, in an effort to provide improved results. The signals that are used to communicate between neurons comprise real numbers. A non-linear function of the sum of inputs to a neuron is used to compute its output. An important part of the neural network is the weights that exist between the neurons and the edges. These weights are adjusted throughout the learning phase of the network and the significance of a signal on the output is also influenced by them [21a].

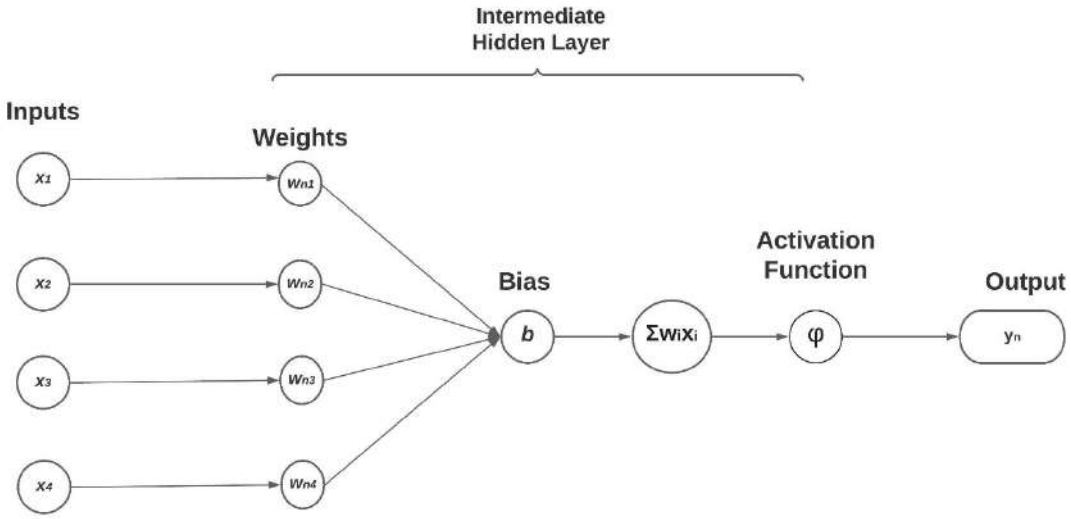


Figure 2.1: Neural Network

2.2.1 Artificial Neuron Model

The above figure 2.1 exhibits a simplified version of the artificial neuron that is based on the model proposed by the neuroscientist Warren S. McCulloch and, a logician, Walter Pitts in their published paper titled "A logical calculus of the ideas immanent in nervous activity", back in 1943. [21a] In this figure, 4 parameters $x_1, x_2, x_3, and x_4$ are being received as input by the neuron n. In the same way, 4 weight parameters are corresponding to the 4 input parameters, labeled as $w_{n1}, w_{n2}, w_{n3}, and w_{n4}$. Another term that is included alongside the weights, is a term called bias. It is similar to the intercept that is added in a linear equation. The output can be adjusted using this parameter, as per the weighted sum of inputs that are received by the neuron. A linear combination and summation of the inputs and the weights is performed, which is used as input to an activation function. This activation function ϕ provides the output of the neuron for the given input and it is denoted by y_n in the figure above.

$$y_n = \varphi(s_n) = \varphi(\sum_{j=0}^{m} W_{n_j} X_j) \tag{2.1}$$

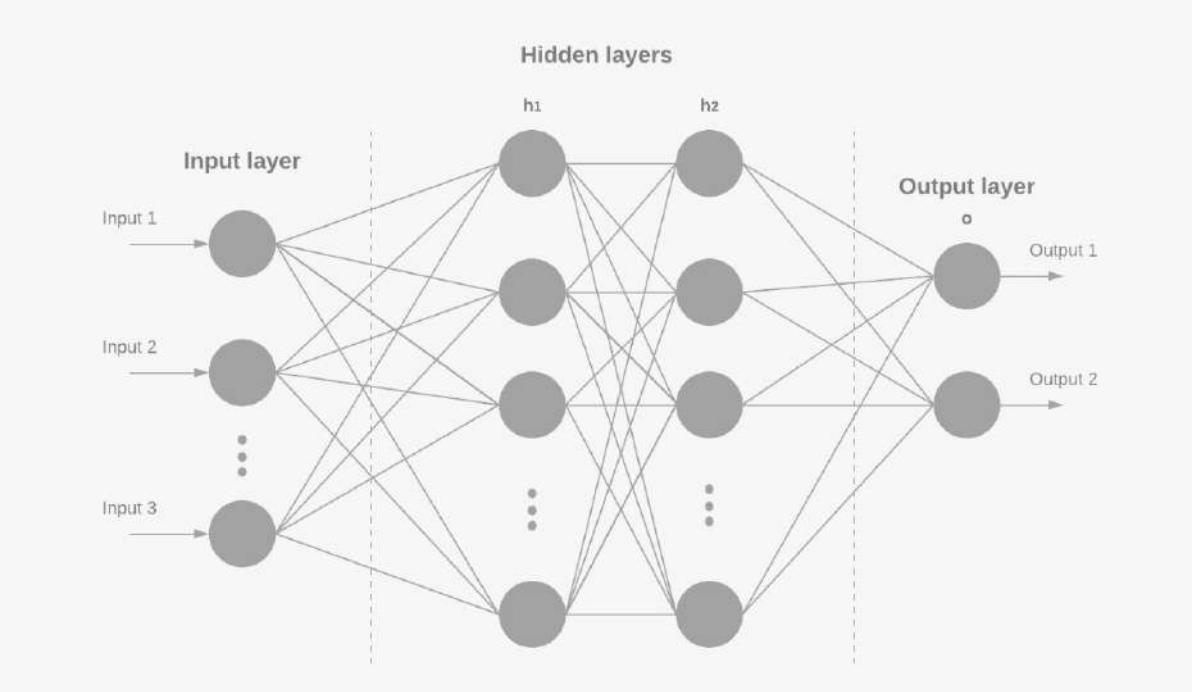


Figure 2.2: Multi-layer Neural Network

2.2.2 Multi-layer Neural Network

We have established that a neural network consists of many connected neurons. These neurons are assembled to form layers that are well connected. The layers are connected in a way that the output of each layer is provided as input to the neuron at the next layer. In this way, we see that the original data that was given as input is only processed by the input layers and the layers following them receive an input that has been processed by the previous layers. The weights of the neurons have values that are equal to the number of neurons present in the previous layer. One such multi-layer network can be seen in the figure 2.2 .

Based on our discussion earlier, we can divide a multilayer network into three main types of layers. These layers are the input, hidden, and output layers. The data at the input of the input layers are passed on to the hidden layers next to them, without much alteration. Thus, a large portion of the computation on the data takes place in the hidden layers. The main function of the output layers is to perform calculations on the data it from the hidden layers and compute the output variables. We will be discussing networks that are fully connected as well as the convolutional neural networks in this thesis. The other few types of networks will not be discussed due to the limited scope

of this thesis.

2.2.3 Back-Propagation

The research on neural networks was advanced by the idea of using more than one layer for learning algorithms and paved the way for multilayer neural networks such as the fully connected, feedforward networks as well as the concept of backpropagation. This concept took form in the early 1970s, it provided a way for neural networks to learn from their mistakes and adjust their layers by propagating the errors at the output, back to the layers. Deep neural networks are being trained by the method of backpropagation. The reason behind its popularity is that the weights of the neurons in the layers of a network are very tough to calculate and adjust manually, by using this algorithm we can compute the weights by using iterations. The gradient descent algorithm is one such first-order iterative optimization algorithm. However, using this algorithm takes up a lot of time and in many cases, does not provide the global minimum of the error without suitable configurations [].

2.2.4 Activation Function

The basic function of an activation function is that is used for computing the output of a neuron. This output is usually computed as the weighted sum of inputs and biases. The effectiveness of a network depends greatly on the selection of the activation function [15]. When an activation function is not selected, the model can be called a linear regression model. In a linear model, the input and output is mapped linearly within the hidden layers. This happens before the labels are finally predicted.

The input vector x transformation is given by $f(x) = w^T.x + b$ where, x = input, w = weight, and b = bias.

Thus, from the above equation, we can see that the mapping produces results that are linear in nature and require an activation function to change them into non-linear outputs that can be computed to learn useful representations in the data.

The output of these models are given by

$$y = (w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) \tag{2.2}$$

The selection criteria for the activation function depends on the output that is expected. Then after applying the activation function, we will get the non-linear output. It is given by:

$$y = \alpha(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) \tag{2.3}$$

where α is the activation function.

2.2.5 Deep Learning

History and Advancements

Deep learning or deep structured learning is a broad class of machine learning. Deep learning has gained extreme popularity in almost all the major fields and extensive research is going on in it, however, deep learning has existed since the 1980s[Goo+16] . Due to several reasons, especially so-called AI Winters during 1974 – 1980 and 1987 – 1993 decelerated progress of research in deep learning.

The major difference in the advancement is made by the provision of specialized hardware and large amounts of data in recent years. Neural networks were found to be trainable on graphics processing units in the early 2000s. graphics processing unit (GPU)s are better suited for processing in these networks compared to the regular central processing unit (CPU)s [16]. High-end graphics cards are in use for training deep learning models both locally and in the cloud, and graphics cards dedicated to deep learning are being manufactured. Apple M1 chip is another breakthrough; packing 16 billion transistors on a small area, executing 11 trillion operations per second is the state of the art Machine Learning implementation on the edge [Bus20]. Today, researchers are working on Tensor Processing Units (TPU)s and application-specific integrated circuits (ASIC)s that will enhance the performance of the neural networks, which are more efficient and much cheaper [21b].

Pros of Deep Learning:

Another breakthrough is in the architecture of neural networks. Modern activation functions and cost functions have resulted in better performances. Replacing the traditional mean-squared error with cross-entropy for example and the sigmoid activation function

Table 2.1: Choosing the right last-layer activation and loss function for your model[Cho17]

Problem	**Activationfunction**	**Lossfunction**
Regression	None	mse
Classification (single label)	softmax	categorical-crossentropy
Classification (multi label)	sigmoid	binary-crossentropy
Binary Classification	sigmoid	binary-crossentropy

with Adam, rmsprop or Rectified Linear Unit (ReLU) results in models achieving higher accuracy. Machine learning problems, broadly classified in regression and classification problems, have activation functions in deep learning suited to each problem type.

In deep learning, the model has one sole purpose: to find suitable representations and patterns from the data. The network or model is intelligent enough to proceed in a direction that maximizes its performance and rejects the features that have no or very little effect on the overall prediction target, most of this is due to backpropagation. There is much less need for human supervised operations in Deep Learning compared to Machine Learning. Due to this reason, feature engineering which is an essential part of the traditional Machine Learning pipeline, is not required in Deep Learning.

Cons of Deep Learning:

One of the main problems in Deep Learning is the curse of dimensionality [Goo+16] . As the number of features increases, the model gets more complex having a greater number of weights associated with these features. This phenomenon slows down the training process of the network. Also, a large number of trainable hyper parameters require that the training dataset is of a large size as well, which is not always the case. The curse of dimensionality introduces slow progress and higher costs thus making the network not scalable. Despite these issues, Deep Learning has emerged to be the main choice for specialized problems such as object detection, image processing, time series analysis, natural language processing etc. because fortunately enough, the real world data is mostly correlated and non-linear. The non-uniform behavior in real world data means that the model can infer patterns and find suitable representations in the data without going into hardcore complex computations, as is the case with uniform data. The non-linearity in real world data can be easily modeled by introducing Activation Function in the neural networks.

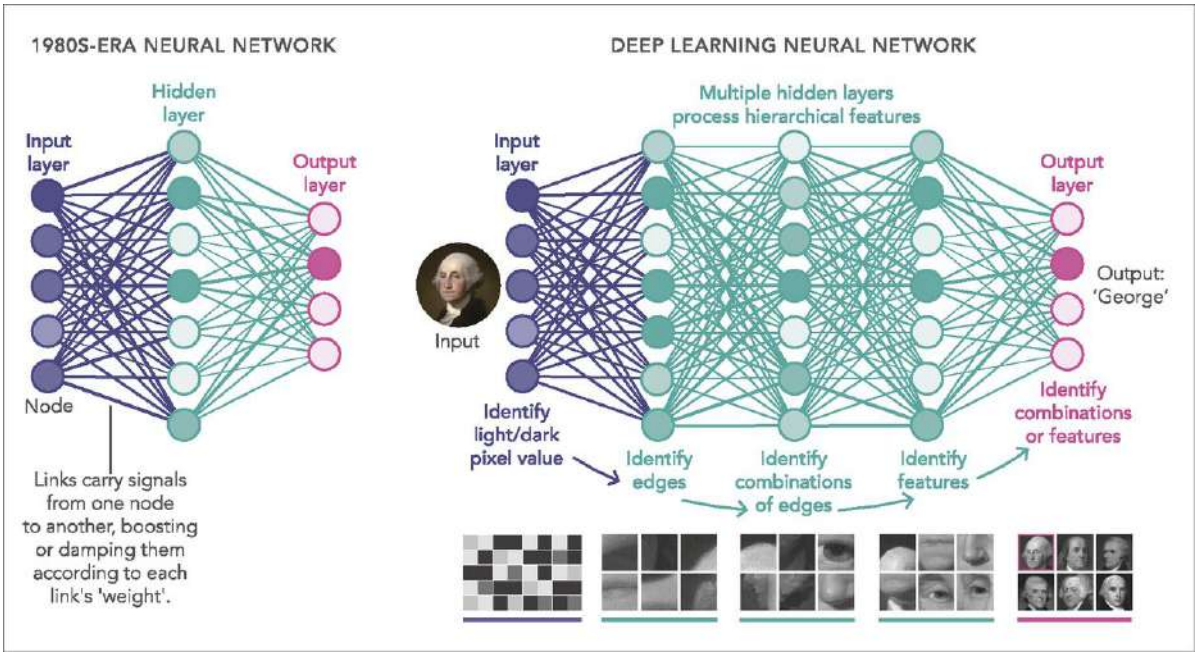


Figure 2.3: Deep Learning Network vs. Traditional Network

2.3 Computer Vision

In this part, we will discuss our main topic of interest which is, Computer vision and a problem related to computer vision that is object detection.

2.3.1 Overview

Computer vision is a branch of artificial intelligence which focuses on the primary aim of training computer to understand and recognize the world using visuals such as digital images and videos. It is being used in various capacities to get useful information from these images and videos. Another field called image processing, is sometimes confused with computer vision however, it is very different from computer vision. Image processing is a kind of signal processing where the input is an image and the output might be an image or characteristic that is related to that image.

Computer vision has many applications such as in image recognition and classification, defect detection, intruder detection, assembly verification, and also in the field of Metrology [19]. Most of the algorithms used in computer vision essentially comprise machine

Figure 2.4: Multi-Object Detection and Classification

learning as well as image processing. Algorithms that can successfully provide a solution to computer vision problems should have the ability to handle enormous amounts of image and video data and extract information from it, often in real-time.

2.3.2 Object detection

Object detection is a technique used in computer vision that can be used for locating, recognizing, and classifying objects images, and videos. It can also be effectively used to count the number of a specific object in addition to tracking their locations accurately as well as labeling them. It is considered a difficult problem to solve in computer vision because parameters of data such as lighting and viewpoint, when changed, should not affect the solution. It is slightly different from whole image classification problems because we need to locate the object before it is classified. This means that the knowledge about the position of an object in the image or video is very important for detecting an object correctly. It is also important to know about the shape of the object that is meant to be detected.

A bounding box is used to define the size and location of the object. The coordinates of the corners of this box are stored and various operations including convolution can be performed on this rectangular box. A machine learning algorithm is used to classify the part of an image bounded by the rectangular bounding box. The rectangular shape of the box can be developed and shaped further, after an initial estimation of the object.

2.3.3 Multi-Object Detection and Classification

In addition to the location of an object, we need to know the segmentation of the image. Image segmentation is used to understand and differentiate between the elements of an image. This helps in counting and tracking the objects within an image. Thus, the image can be divided into parts and then only those parts can be used that contain useful information. This is a simple explanation of the working of image segmentation. The pixels of images having identical features can be grouped together using image segmentation to create a pixel-wise mask for objects in the image. This provides us with in-depth understanding of those objects [Mst20].

Types of Image Segmentation

[47] There are two types of image segmentation techniques that are mentioned below:

Semantic Segmentation:

In this type of segmentation, each pixel fits into a certain class. And the pixels of each class are represented by the same color.

Instance segmentation:

In this type of segmentation, classes are assigned to the pixels of an image but different objects of a class are represented by different colors.

Feature descriptors were being used for object detection back in the 2000s as a solution. These descriptors include the Histogram of oriented Gradients (HOG) and Scale-invariant feature transform (SIFT), however after the year 2010, convolutional neural networks were becoming more popular for object detection problems. The workings of the above mentioned algorithms are provided below.

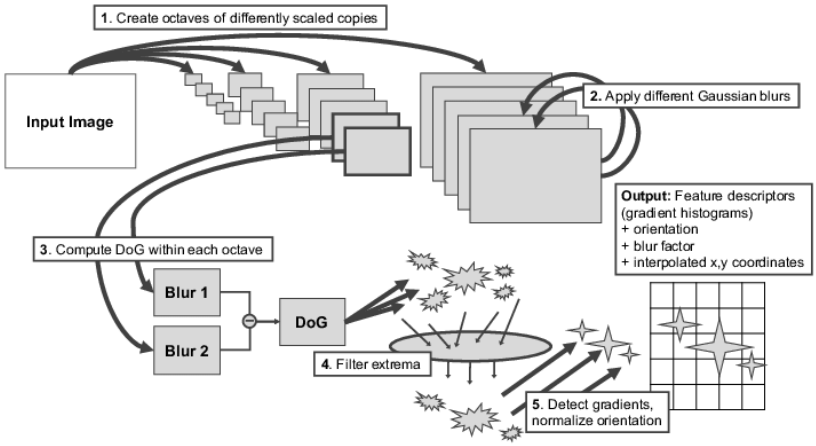


Figure 2.5: SIFT Algorithm Working

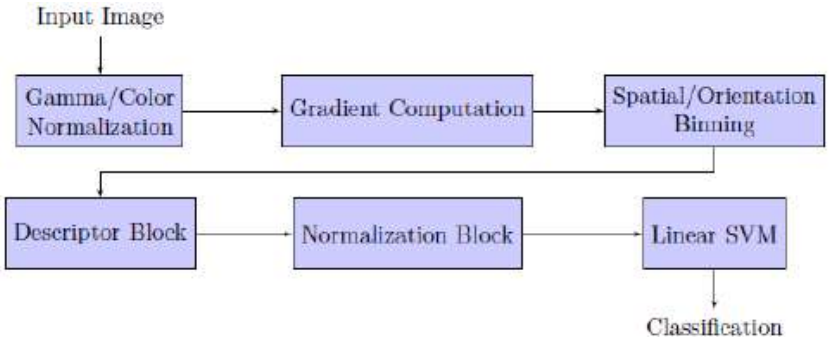


Figure 2.6: HOG Algorithm Working

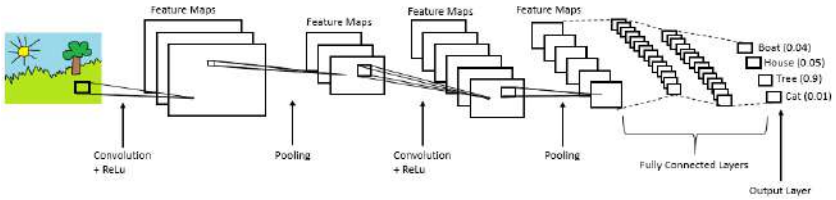


Figure 2.7: CNN Algorithm Working

2.3.4 YOLO Model Description

You Only Look Once (YOLO) is a state-of-the-art model for real-time multi-class object detection. It was introduced in 2015 and outperformed all the previously used object detention methods namely R-CNN, Fast R-CNN, and Faster R-CNN with excellent performance particularly in image annotation [60], activity recognition [61], and video object co-segmentation. YOLO stands for You Only Look Once which essentially is the way how YOLO object detection works. It divides the whole image into multiple grids and locates the centers of objects.

A brief history

YOLOv1 was introduced in 2016 by Joseph Redmon and others [62]. A year later, YOLOv2 with faster speeds supporting up to 67 frames per second was proposed by Redmon and Farhadi [63] The subsequent versions of YOLO have slightly less frame rate per second support but great improvements in inaccuracy. In 2018, YOLOv3 was proposed [64] with an improvement in accuracy. Alex Redmon terminated his research on Computer Vision soon after he proposed YOLOv3, due to his conflict on ethical concerns of research in CV. YOLOv4 was proposed by Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao in 2020 [64] having a higher number of convolutional layers in the neural network architecture. YOLOv4 has a 10% improvement in accuracy compared to the previous version and 12% in frame rate per second (FPS).

2.4 Datasets

A dataset can be defined as a collection of data, commonly distributed into rows and columns in such a way that each column represents a certain variable and each row relates to a record of the set. For example, in a data set of patients, the columns could indicate the attributes such as the height, weight, and blood type of the patient whereas the rows could correspond to the individual records of patients identified by their names.

Datasets are the building blocks of machine learning models. They are used to train and test the models. Many different kinds of objects can be detected with the help of relevant data sets that have a sufficient amount of data to train the model completely.

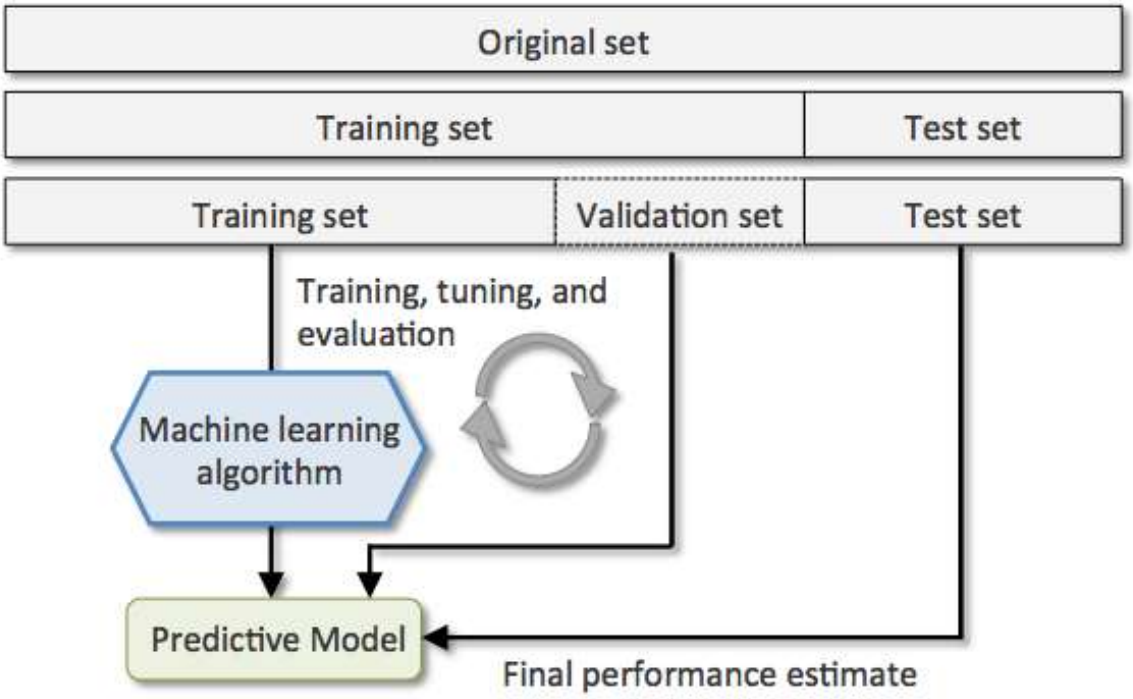


Figure 2.8: Dataset

We can say that it is impossible for an algorithm to learn without data. It should also be noted that although the size of the dataset is important, it is also equally important that the data set contains relevant and useful data otherwise the model can never be able to train correctly. Moreover, the classification and labeling of the dataset is also a very important task. This is because the labels of the dataset must be correct in identifying the data.

2.4.1 Types: Training and Testing

A dataset can be divided into two main types, in order to train a machine learning model. The first is the training data set. This set consists of data that will be used to train the model. It consists of the input data as well as the output that is expected from the model. A major of the total data is used as training data.

The second division is the test data. This data is used in the evaluation of our model that was trained on the training data set. The training data cannot be used for testing the model, because the output from that data will already be known to the model.

2.4.2 Dataset Preparation

There is a good chance that the dataset we have chosen, cannot be used for training the model right away. The dataset may need to be prepared, before feeding it as input to the model. This preparation is dependent on the kind of data, goals of the project as well as the models that are used. This process consumes a lot of time mainly because the data points must be manually analyzed and prepared for our model. It is important because the performance and quality of the output from our model are determined by the quality of the dataset. A large dataset does not mean that it can produce meaningful output, rather our aim should be to gather an assortment of data that can make the model learn about the target object. For example, a dataset of images should contain images with different backgrounds, angles, lighting conditions, etc. Data set preparation includes the pre-processing of data to select and build the most suitable data set for training the model. This process is called "feature transformation". This process includes tasks such as mentioned below:

- Data Cleaning: The process of recognizing mistakes in the data and correcting them accordingly.
- Feature Selection: Analyzing the data set for the input variables that are most appropriate for our task.
- Data Transforms: Changing the format, structure, or distribution of variables.
- Feature Engineering: Creating new variables from existing data.
- Dimensionality Reduction: Transforming the training data to reduce the number of input variables so that the model has to train on fewer parameters.

2.5 Frameworks

What is a framework A software framework is a level of abstraction which in essence is, a collection of libraries, support systems, a scripting language, and other software to build upon the previous components by integrating them in a software application designed with reusability in mind. Frameworks are commonly used as APIs with exposed endpoints.

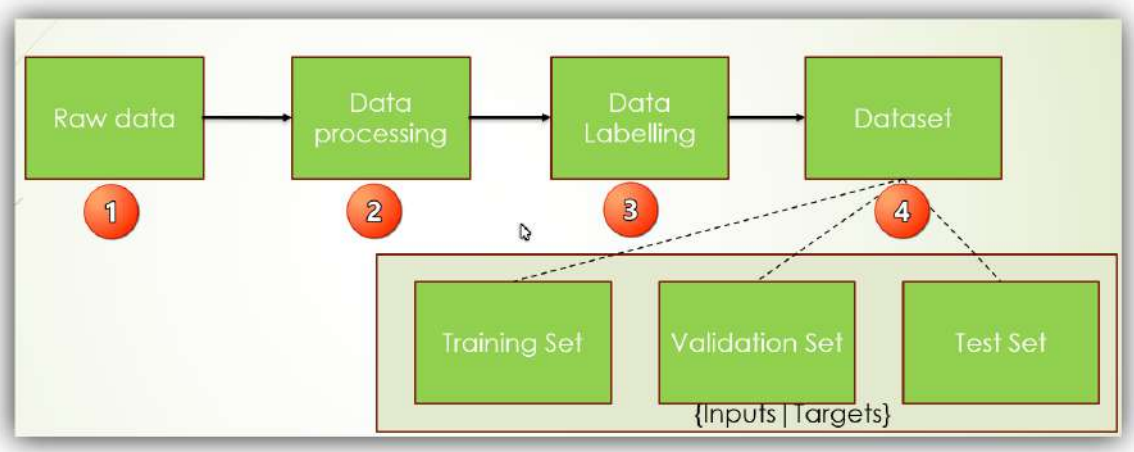


Figure 2.9: Dataset Preparation

Framework v/s library: Frameworks 'encapsulate' libraries along with other resources or assets such as images, data files, and other software objects. Libraries are a collection of functions, objects, and methods containing reusable code which can be possibly utilized in various applications. Frameworks, on the other hand, can be a collection of entities used in application development. It can be one single library, multiple libraries, scripts, software.

The Darknet Framework Darknet[50] is a neural network framework written in C and CUDA which supports CPU and GPU computation, provisioned by the Nvidia CUDA API. It is open-source [51] and was created by the YOLO author Joseph Redmon. Darknet depends on Open Source Computer Vision (OpenCV) for tasks involving image processing and Compute Unified Device Architecture (CUDA) for GPU-intensive tasks and is used in advanced implementations of Deep Artificial Neural Networks. These implementations include You Only Look Once (YOLO), ImageNet, Recurrent Neural Networks for real-time object classification, natural language processing (NLP), and time series analysis. Other implementations include Nightmare and DarkGo. Nightmare uses the convolutional layer of Visual Geometry Group (VGG)-16 model and runs the model 'backward' instead of forward and essentially shows what the model is learning in the form of distorted and malformed images. DarkGo is an implementation of game-playing neural networks that predict the most probable next moves in the game of Go.

Installation and Usage: The base environment of Darknet can be installed from the GitHub repository via the command

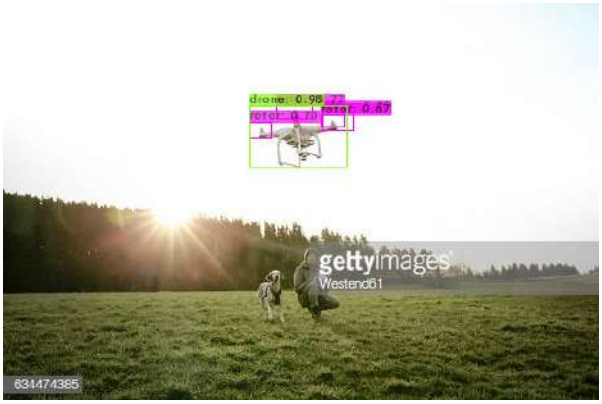


Figure 2.10: Object detection

```
git clone https://github.com/pjreddie/darknet.git
```

to clone the files followed by make to initiate the installation.

Using a pre-trained model for real-time object detection and classification: YOLO is a powerful method to perform such tasks with the integration of Convolutional Neural Networks (CNNs) with advanced image processing techniques. First, the data set with pre-trained weights are imported. The weights for YOLOv4 can be downloaded using the wget command wget `https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights`. The command detect performs the detection and generates output with labels of images on a bounding box. This works equally well for multiple objects in a frame.

2.6 Convolutional Neural Networks:

As discussed in section 2.3, Artificial Neural Networks have shortcomings when we try to solve image processing, classification, and segmentation problems with them. We are going to discuss these problems and how Convolutional Neural Networks or ConvNets (CNNs) address these problems.

2.6.1 Justification

When working with image or video data, the inputs are multi-dimensional. Image data is usually represented by four-dimensional tensors. These dimensions are height, width, and color depth along with the batch size. Image data can be reduced to three dimensions if we use grayscale images since grayscale images have a single-dimensional color channel. Video data is represented by 5-dimensional tensors. Videos are sequences of frames of color images [Cho17] If traditional neural networks (ANNs) are used to handle these computer vision problems, data should first have to be reshaped to one-dimensional tensors. This results in a huge number of trainable parameters (weights and biases). ANNs are not scalable.

ANNs cannot handle pixel shifting and correlated pixels. Optimized pattern detection requires algorithms that can take advantage of correlated pixels. Specialized filters in CNNs can detect and group pixels that have the same features, for instance, color. Similarly, ANNs are not translation invariant [Mee21]. Pixel shifting is not tolerated by regular neural networks and they essentially re-train on the same patterns they previously trained on.

2.6.2 Limitations of ANNs

The inputs for ANNs should be 1 dimensional and valid data points. ANNs rely heavily on weights. After each epoch, the loss function calculates the loss, and the weights and biases are changed via Backpropagation. A large number of these parameters affects the performance of the neural network, takes large amounts of storage, and is a slow process due to complex computations. This also results in overfitting.

In contrast, Convolutional Neural Networks do not have weights. CNN's use multiple layers and filters to extract suitable representations from data.

ANNs require valid data points as input. This means that distinct features have to be explicitly provided to the neural network as data. This task easily becomes non-viable on large datasets or even on small datasets with large possible features or representations. For classifying drones, we would have to manually think of all possible features that differentiate drones from other aerial bodies (birds, trees, airplanes).

Convolutional Neural Networks solves this problem by extracting useful representations from the input data on their own without any human supervision.

Pros of CNNs

Convolutional Neural Networks perform exceptionally well in image classification problems because:

1. CNN's have reduced the number of input nodes, and therefore, reduced the number of trainable parameters. A 6 x 6 image trained on an artificial neural network would have 36 weights and biases for each node in the first hidden layer. The same 6 x 6 image trained on a convolutional neural network having a 3 x 3 filter would result in only 4 weights and biases per node in the first hidden layer.
2. Pixel shifting can be tolerated by convolutional neural networks.
3. Convolutional neural networks are highly optimized and translation invariant and act as large feature detectors.[LeC+89]

Cons of CNNs

1. CNNs have specific hardware and storage constraints.
2. CNNs also require larger amounts of data for better accuracy; techniques such as data augmentation is used to address this.

2.6.3 Basic Structure

Convolutional Neural Networks are inspired by the receptors in the visual cortex of animals.[Fuk88]. These receptors are sensitive to edges and can be mimicked by image processing filters in computers. The thing which differentiates the CNNs from ANNs is the convolution layers. These convolution layers also process the input data and provide output, but differently than the regular neural networks. Inputs to convolution layers are called 'input channels', the outputs are called 'output channels'; there are filters present in these layers which play the fundamental role in CNNs. Different convolution filters present in the convolutional layers are used to detect a variety of edges and features in

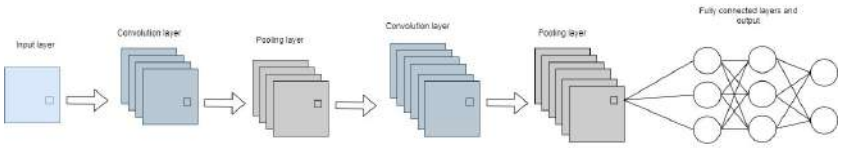


Figure 2.11: Basic Structure of CNN

image processing. Some filters are used to detect horizontal edges, while others for the detection of vertical edges. Highly specialized filters can also detect textures, shapes, boundaries, colors, curves, and objects.

In a convolution layer, filters become successively advanced, from detecting simple edges to detecting complex curves and objects and finally detecting whole entities. Essentially, the deeper the network gets, the more specialized the filters become.[Fuk88] The number of convolution layers and filters in each layer are under human supervision.

Convolutional network example

On a superficial level, CNNs are a group of filters applied on data (images) where each filter has a specialized purpose for detecting a specific representation in the data. These filters are placed in a cascade-like manner, much like the hidden layers in a regular neural network. A filter does its processing on the data after the previous filter finishes its processing.

Mathematically, these operations can be modeled by the discrete convolution operation on an image f and a filter matrix g.

$$h[x,y] = f[x,y] * g[x,y] = \sum_n \sum_m f[n,m]g[x-n, y-m] \tag{2.4}$$

2.6.4 Regularization and data augmentation

As discussed in the section on Neural Networks (insert section number here), the fundamental issue in machine learning is to find the sweet spot between optimization and generalization [Cho17]. Optimization is the process of tuning the model to get the best possible classification accuracy on the 'training data, while generalization refers to how well the model performs on never-seen-before data, that is, testing data or real-world

data after the model has been deployed to production. Unbalanced optimization and generalization lead to overfitting and under-fitting, we aim to find a suitable balance for the two.

Regularization and data augmentation are some techniques used to cope with overfitting. In regularization, we introduce some sort of penalty term to the cost or loss function of the network that affects certain types of weights. There are many approaches for regulation, for example reducing the network size, weight regularization (L1, L2 regularization), and adding dropout layers [Sri+14]. Dropout layer regularization is usually used in which we reduce the dependency of the network on any single neuron (thus reducing overfitting) by randomly dropping out neurons with a predefined probability by introducing dropout layers. In CNN's, dropout is typically introduced in the final dense layers [SZ15].

Another way to reduce overfitting is by increasing the amount of data. This is called data augmentation. Essentially what we do is, add different transformations of the already present training data. For example, transforming the input images by rotating, or flipping, or sampling with different crops and scale values. Noise can also be introduced [Goo+16].

2.6.5 Development

The history of CNNs goes back to the 1980s when Yann LeCun, Turing Award winner postdoctoral computer science researcher built on the previous work of Kunihiko Fukushima, The Neocognitron which was a very naive image recognition neural network [Fuk88]. LeCun combined The Neocognitron with the backpropagation algorithm [LeC+89] to form what we refer to as LeNets. LeNet is one of the earliest forms of convolutional neural networks finding its application mostly in recognition of handwritten digits. These networks were mostly used in postal services or banking where they were used to recognize zip codes and digits written on envelopes.

A major drawback of these networks was the lack of scalability, requiring large amounts of data and hardware resources. These issues, along with the era of so-called AI Winter, halted the research progress in CNNs. In 2012, AlexNet, a CNN architecture designed by Alex Krizhevsky trained on the ImageNet data set having millions of labeled images resulted in great success in the field of deep learning.

Similarly, in 2014, Simonyan and Zisserman [SZ15] improved the architecture of ImageNet by increasing the network depth-wise and achieved better accuracy on classification problems. The ImageNet challenge 2016 winner [Rus+15] is also CNN-based, using ensemble learning techniques, CRAFT region proposal generation [Yan+16], and gated bi-directional CNN [Zen+16]. ImageNet has proved to be the building block for the current fancier and robust object detection methods, some of them will be described in chapter 3.

3 Drone Detection and Classification with Rotor Labelling

In this chapter, we will discuss the experimental setup of the thesis. To start with, we will discuss the selection criteria for applied methods and datasets for drone detection. We will discuss the selected methods and datasets and evaluate them. results of these methods is discussed in greater detail in successive chapters, however, some implementation details are also discussed in this chapter.

3.1 Selection criteria of drone detection

The experimental part aimed to implement an object detection system specifically for the detection of drones, and evaluate it based on how well it generalizes after training on the selected dataset and tune the model to achieve improved performance. These goals, along with the practical environment such as available hardware and computation capacity proved to be the selection criterion for the methods and datasets used for experimentation.

Hardware constraints and the availability of training data are the major limitations for the application of deep learning methods. We did not have access to a server farm or a high-performance GPU for research purposes. Initially, we tried to train the model locally on a regular consumer laptop, however, the model failed to train owing to the limited computation power. Google Colaboratory proved to be a good alternative for experimentation and research due to the provision of free GPUs and minimum configuration required, however, Google Colab has an idle timeout of 90 minutes and the maximum lifetime is 12 hours. This means after ninety minutes of human inactivity, the training process is stopped and has to be initiated again. One workaround was to

save the weights after each time the training stopped and then resume training after initializing the model with those weights. This process was not viable. This issue was resolved by initiating the training process on a server and using the GNU Screen [47] to prevent the server from going into sleep mode while training. The model takes around 16 hours on average for training.

In addition to computation power constraints, deep convolutional neural networks also require large amounts of data for training. Since collecting a labeled dataset of sufficient size was a tedious process, our research is based on the already available dataset called drone net [48].

This data set consists of 2664 images of DJI drones pre-trained on YOLO. One issue with drone-net is that it had only one class i.e. drone. Models trained on this dataset had low accuracy scores on testing data due to this reason. Fortunately, original images without annotations were also provided in the drone-net dataset and we annotated the data into two classes i.e. drone and rotor. Annotation was done according to 4 points annotation convention in YOLO using the labelImg open-source graphical image annotation tool [49], and thus the single class classification task was turned into a binary classification task resulting in greater accuracy on evaluation.

3.2 Types of Drones

Major risks are posed by the drones, because of their ability to fly quickly and swiftly while remaining undetected. This makes them ideal for tracking, disrupting safety, entering unauthorized areas as well as inflicting damage to property. These reasons have compelled regulatory authorities to devise regulations for these objects. The classification of drones that are available largely depends on the perspective from which one is looking at them. Therefore, we can classify them on the basis of their utilization whether it may be for commercial purposes or simply, recreation. Another way to look at them is by classifying them based on their design. We will focus on and discuss the latter type of classification in this chapter.

The detection and classification of these drones by the object recognition algorithms depend upon their visual features as well as their movements. The shape of a drone is one of the most important visual features for detection. We see that the multi-rotor

drones such as the tri-copter have a triangular shape whereas a quadcopter is usually designed in the shape of a square. In addition to the shape of the drone, we see that each type of drone has very well-defined spaces for storage of its control board as well as arms depending on the number of rotors. At the ends of these arms, the rotors are firmly installed. These visual features are shared among the various types of drones except the latest nano or mini drones. These drones have long battery lives, so they are able to function for long times in the air as well as fly higher than other types of drones. Drones can also be characterized based on individual components. It means that they can also be categorized, based on some components that may be present or absent from the design. Some drones have designs that are enclosed by a ring. We also come across drones having parts that are entirely uncovered. Cameras can be installed on drones, at locations that are not noticeable, inside its body whereas cameras that provide better quality output have to be uncovered and visible, usually attached at the bottom of the drone. The detection of drones can be more effective in the cases where lighting conditions are unfavorable for those drones that carry a light-emitting device on the lower part of their bodies. It has also been observed the movement of drones follows certain patterns, and can change their speed and direction instantly. These movements can be detected and differentiated from those of the birds.

Now, on the basis of design, we can broadly classify drones into three categories. The first category consists of sigle rotor, the second category consists multi-rotor drones and the third category consists of fixed-wing drones. This is the simplest type of classification of drones as the classes are very distinct thus allowing them to be classified easily and clearly. Generally, when we think about drones the first picture that comes to mind is the multi-rotor drone, usually a quad-copter which is a multi-rotor drone having four rotors. However, multi-rotor drones are not limited to quad-copters, rather they are available in several distinct shapes, sizes, and the count of rotors they have installed on them. Generally, the cost of drones rises with the increase in the number of rotors as the drone becomes larger to accommodate more rotors. Hence, popular multi-rotor drones include hexacopters as well as octa-copters. The former is used in industries for inspections and to gather aerial data for maps whereas the latter is used in the film industry for high-end cinematography. The fixed-wing drones resemble airplanes that have a fixed-wing in the middle of their bodies. They have a firm structure that assists in the generation of the lift under the fixed wings. This lift is caused by airspeed in the forward direction which is generated by the internal engine or the propeller that is controlled by an electric motor. They are often used in the survey of huge pieces of land. They have very long

flight times and payload capacity as compared to the multi-rotor drones. These types of drones are also being used for delivering medical supplies and facilities to remote areas that cannot be acceded by land routes. Zipline, an American medical products delivery company, is designing, manufacturing, and operating delivery drones in remote areas such as in Ghana, for delivering blood, vaccines, and other medical supplies [51].

The main purpose of our research is to detect these drones through reliable methods that can provide the best accuracy and are economical. The approach used in this thesis will be able to detect multi-rotor drones based on their appearance. The multi-rotor drones are our primary focus because of their usefulness for many applications.

We can generally, divide the detection of drones into two main approaches, which are visual and acoustic. The visual approach utilizes cameras that are capable of recording 2-D images, additionally, cameras that can record in the dark can be beneficial for effective detection. It can combine the appearance and movement of the drone to detect it with precision.

Due to the small sizes of drones, the visual approach may become inefficient, due to lighting, obstruction of views, or the field of view being very narrow. This is why the acoustic approach is used in drone detection because it can detect even when the drone is not in range of sight [53]. This approach may also be erroneous at times, due to the noise in the audio signals, however, it has also proved to be accurate up to 98.3%. The detection can be done using the help of minimal devices such as microphones and computational devices that are essentially inexpensive.

It is also possible to combine both of these approaches, into a hybrid system for the detection of drones. One such example is the anti-drone system described in [52]. This system uses a combination of audio, video, radio-frequency sensors, and a radio-frequency jamming unit.

3.3 Darknet Framework with YOLO version 4

Now, we will discuss about the YOLO version that are built upon the darknet framework.

3.3.1 YOLOv4

YOLO v4 was not developed by the original authors of the YOLO model. He stopped the research on YOLO v4 specifically due to ethical concerns. He was concerned about the misuse of his technology and referred to "military applications and data protection issues" as his primary concerns [70]. Thus, YOLO v4 was developed by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao [71]. CSPDarknet53 serves as the backbone of this model's architecture. The improvements in this model include spatial pyramid pooling additional module, PANet path-aggregation neck, and YOLOv3 head.

CSPDarknet53

CSPDarknet53 uses the DarkNet-53 framework [69]. It uses a CSPNet strategy to split and merge the feature maps generated in the base layer. First, it splits them into two parts, and then these parts are merged through a cross-stage hierarchy. The most important context features of the image are extracted using the block called "spatial pyramid pooling block". This block is added over the CSPDarknet53 to improve the receptive field as well. The technique used for aggregation of parameters is PANet, whereas the FPN technique was being used in the previous version, for different levels of detectors.

Advantages

The improvements that were applied in YOLOv4 have greatly improved its performance as compared to its competitors. According to an analysis, it performs two times faster than its competitor, the "EfficientDet" model [72]. The Average Precision, as well as the Frames Per Second, has been increased by 10% and 12% compared to its previous version. Due to these advantages of YOLOv4, we have chosen it for our research. YOLO v4 provides faster FPS and accuracy than detectors that are currently available.

3.4 Datasets of Flying Objects

These are some of the famous freely available datasets related to flying objects. Many machine learning projects and research topics are based on training models on visual data. Medical Imaging, Facial recognition, motion and object detection, drones are some examples where computer vision finds application. Image and video datasets can be used for a wide range of tasks related to computer vision. However, for building a robust deep learning computer vision model, high quality training data is needed. The amount of data required is subjective to the problem at hand, but as a general rule of thumb, a dataset should have enough samples for the neural network to find all local and translation invariant features and representations. Some datasets for flying objects are listed below:

3.4.1 Flying-planes

This is an extensive dataset[cite1] comprising a variety of flying vehicles including drones, helicopters, passenger planes, fighter jets, missiles, and rockets. The source of data is Google Image Search and the dataset can be used for multiclass classification. The data is 1.85 GB in size having 6 classes of labeled images. There are 8530 image files in various formats including jpg, jpeg, png, and others. The dataset is almost two years old and has not been actively maintained. It is also not cleaned and preprocessed properly and contains many irrelevant images and files which are not images. Image classes are slightly imbalanced too i.e. the number of images in each class is not the same.

3.4.2 Drone Dataset UAV

This dataset[cite2] contains 1359 labeled images of unmanned aerial vehicles (UAVs). The dataset is limited to the rotary-wing UAVs and does not include unmanned aerial vehicles with fixed wings. The labels are provided in both .xml and .txt file formats for training on YOLO, TensorFlow, and PyTorch. The data collected in this dataset generally comprises drones in the sky with a variety of background conditions and plain drones images with no background. The size of the dataset is almost 725 MB and the data is scraped from Google and Yandex image searches. Some of the data is also prepared by cropping images from YouTube videos. Images are present in .jpg, .jpeg

Figure 3.1: Flying Planes Dataset

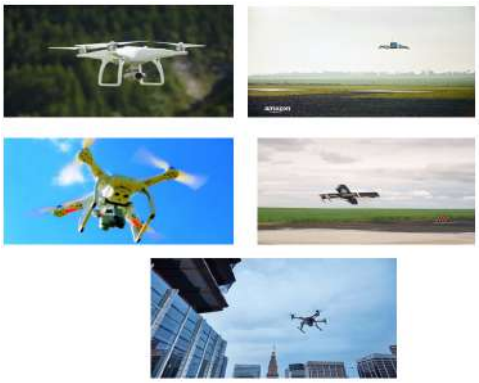


Figure 3.2: Drone Dataset UAV

and .png formats, however .jpg format is recommended for achieving better results. On training the dataset YOLO-v3, the model gets to around 86%mAP and loss converges to almost 0.2 after 2000 epochs.

3.4.3 Amateur Unmanned Air Vehicle Detection

This dataset[cite3] consists of more than 4000 amateur drone pictures in a variety of different background conditions including background where it is very hard to distinguish between the drone and its background. The dataset is just 168 MB in size and also has "negative" labeled pictures which are essentially drone-like but non-drone objects in order for the model to be more robust and generalized. The data is suitably structured and prepared to work well with YOLO architectures and the Dark-net framework and

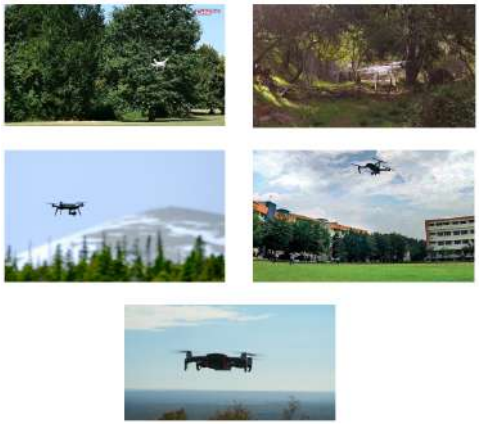


Figure 3.3: Amateur Unmanned Air Vehicle Dataset

has been tested on YOLO v2-tiny and YOLO v3-voc. The dataset was made publicly available in 2019 and was updated in 2020 so it is fairly active.

3.4.4 Flying Object Detection from a Single Moving Camera

This dataset[cite4] contains videos of drones and planes in various backgrounds and scenes. The videos of drones amount to almost 4 GB of data and the plane videos make up to almost 2 GBs. We can see in the videos that the drone appears at various parts in the video and the illumination and noise coming from the background is different in most videos. Similar to previous datasets, this dataset also annotates only the drones in the frames of the videos. As the size of the dataset is very large, it contains a lot of resources for the training of a deep learning model. The videos provide many different learning opportunities for the model as there are both darker and less illuminated videos as well as those where the background is brighter and some videos also contain interesting and complex backgrounds. These different qualities of the videos make it difficult to detect the drones within the frame and thus the annotation will help the model to achieve the desired results.

3.4.5 A Deep Learning Approach to Drone Monitoring

This dataset[cite6] contains a large collection of images and videos of drones that have been labeled. The drone images in the dataset have been annotated to the position of the drone in the images. Due to a limited amount of training images, the deep

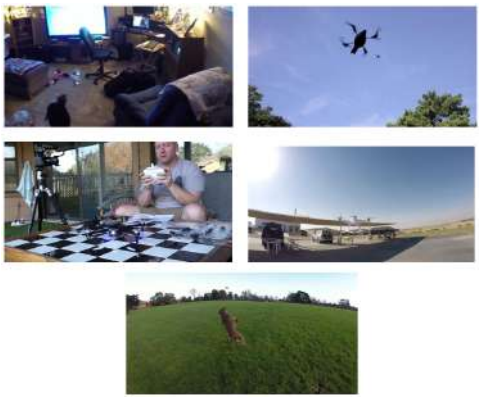


Figure 3.4: A Deep Learning Approach to Drone Monitoring Dataset

learning methods are difficult to adopt. This problem was overcome by the creators of this dataset with the help of an augmentation technique that was used to automatically generate the drone images with bounding boxes on the location of drones. The images in this dataset have not been annotated based on the rotors. This emphasizes the usefulness of our Rotor-Drone Dataset. This drone detection and tracking dataset with user-labeled bounding boxes are available to the public.

3.5 Annotation of Datasets

3.5.1 How to Perform Dataset Annotation

Annotation:

A labelled or annotated image dataset is a prerequisite for training an object detection and classification model, whether using either simple convolutional neural networks or the YOLO framework. One of the commonly used techniques for image annotation is the 'bounding box technique'. In essence, it is the process of drawing bounding boxes enclosing different objects in a frame and then labelling the bounding box.

LabelImg[52] is an easy to use and open source tool written in Python that can be used for image annotation. Images can be directly saved in the format specified by the YOLO framework.

Usage

LabelImg can be used with the command python3 labelImg.py, this will open the LabelImg tool in a graphical user interface (GUI) where users can interact with it. It can also be used as a command line interface (CLI) tool by executing it with Python and passing the image path and predefined classes or labels as the command line arguments. python3 labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]

Steps for generating annotated images for YOLO:

1. A text file is created which defines the list of all classes used in training. This file should be placed in the **data** directory and named **predefined_classes.txt**.
2. Launch the LabelImg tool using the command **python3 labelImg.py**.
3. Switch to YOLO format by clicking on the **PascalVOC** button below the **Save** button in the toolbar.
4. LabelImg allows processing single or multiple images simultaneously.
5. Click on **Save** after annotating a single image. This will generate a text file of YOLO format for the current image and another text file named classes.txt which has the collection of all classes the YOLO label is referring to.

3.5.2 Drone-net Dataset: A Drone Dataset : Single Class

Drone-net [Red17a] is a publicly available dataset for drone detection. It has only one class which called drone. The dataset has 2664 images of drones that are pre-trained with YOLO. Both, Original and labeled images are available. The size of the dataset is 92.1 MB. Normalized labels are also provisioned for compatibility with different variants of the YOLO algorithm. Any sort of preprocessing was not done before training. YOLOv3 and darknet53 [Red16] (pre-trained on ImageNet) were used in training to generate the drone weights. This is advantageous since these pre-trained weights can be used to train farther away drone images without any re-training.

Labeling is according to YOLO conventions i.e. the class ID is the initial digit of each label. In YOLO, each class has a unique ID that starts from 0 and increment for each successive class; in this case, however, it is only 0 since there is only one class (drone).

Figure 3.5: Labelled image of the drone

Figure 3.6: Labelled image of the rotor and the drone

The labels are in the format: [class_id] [x_center] [y_center] [x_width] [y_width] which is the standard format for YOLO.[Red17b]

3.5.3 Rotor-Based Drone-net Advance Dataset:(RBDNA) Two Classes

The images in the Drone-net dataset were annotated based on two classes; rotor and drone; using LabelImg software. Some of the annotated images are included in this section.

In this figure 3.6, we can see that the multiple rotors of the drone are annotated in addition to the body of the drone. This close-up image of the multi-rotor drone will help the model to recognize the drone more efficiently in images where the drone occupies most of the pixels.

(a) (b)

Figure 3.7: Labelled image of the rotor and the drone

(a) (b)

Figure 3.8: Labelled image of the rotor and the drone

In the first image 3.7a, the visible rotors and body of the drone have been annotated at a distant view. The scenic background will help the model to differentiate and learn about a distant perspective of the drone and its rotors. It will also improve the detection accuracy in an illuminated setting as well as when the drone is flying on the horizon. The second image 3.7b also exhibits somewhat similar features of the drone and its rotors in the sky. The annotation is a bit unclear but the model will recognize the drone from this distance with the help of separate annotations in the image.

These images 3.8a and 3.8b provide a close and distant view of the drone and the rotors. It should be noted that the close-up image shows a slightly larger and different design of the drone body than some of the previous drones. This helps the model to learn about the different sizes and designs of the drones. The mountains in the background will train the model to detect drones in mountainous regions such as in the picture.

These images 3.9, 3.10 and 3.11 contain annotated drones in different backgrounds and lighting conditions. The first image will benefit the model by training it to detect

Figure 3.9: Labelled image of the rotor and the drone

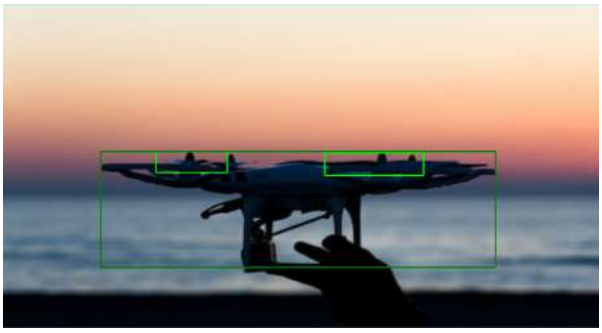


Figure 3.10: Labelled image of the rotor and the drone

Figure 3.11: Labelled image of the rotor and the drone

Figure 3.12: Labelled image of the rotor and the drone

Figure 3.13: Labelled image of the rotor and the drone

drones in a slightly different shade of the sky and the ocean. In the second and third image, the background consists of a mixture of the sky and the ocean in addition to the sun in the third image. These images have lower levels of lighting which will help the model to detect the drones in minimal lighting conditions.

These close-up images 3.12, 3.13 and 3.14 provide a clear view of the drone and the size and shape of its rotors. This will help the model to learn about the various sizes and shapes of the rotors. This will be important in improving the accuracy of the model.

Figure 3.14: Labelled image of the rotor and the drone

Figure 3.15: Labelled image of the rotor and the drone

Figure 3.16: Labelled image of the rotor and the drone

In this 3.15 image, the six rotors of the drone are clearly identified by the bounding boxes. As the drone is present near the camera, its features are visible with great detail. The drone is also slightly different in design as well as size in comparison to the other images. The background is a wooden texture with several shades of light and dark colors. This background can help the model to differentiate a six-rotor drone from a complex wooden background. The drone which is bounded by the box having a dark green outline can also be detected with the help of the arms that hold together the body and upon which the rotors have been installed.

In this 3.16 image, the drone is seen at distance and is bounded by the dark green outlined box in addition to the light green boxes that bound the rotors of the drone. As the drone is far away from the camera, only three of its four rotors can be detected and are identified by the bounding boxes. The body of the drone as well as the equipment attached to it can also be clearly seen in the picture. The part of the background which contains the drone is a clear sky and the drone stands out completely in that area. The buildings that are visible in the background will allow the model to detect the drone in similar scenarios with a clear sky.

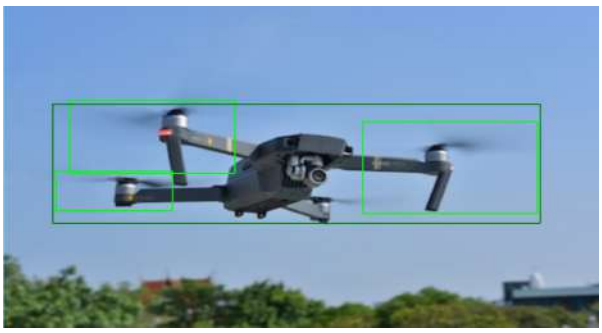


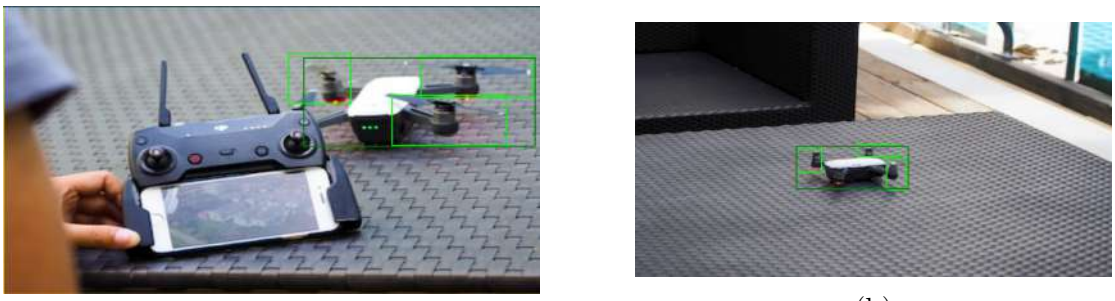
Figure 3.17: Labelled image of the rotor and the drone

(a) (b)

Figure 3.18: Labelled image of the rotor and the drone

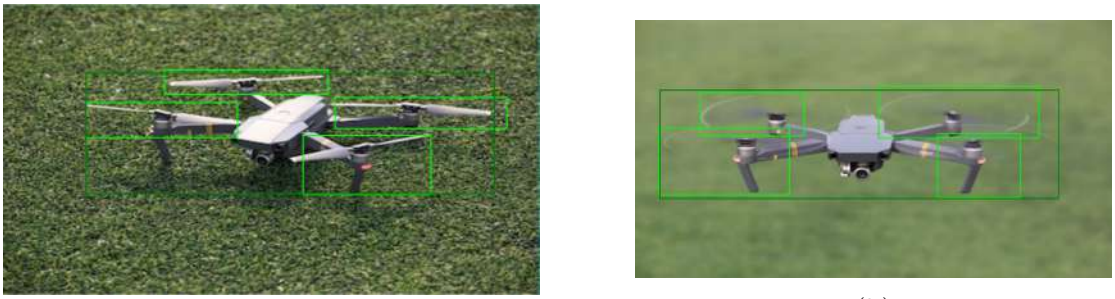
This 3.17 image provides a close-up view of the dark-colored drone that has four rotors attached with its body through the arms. However, only three rotors could be identified and bounded in this image. The rotors can be seen to be in a rotating motion that makes its detection slightly difficult due to the rotor blades appearing almost invisible next to the arms. The drone body also contains a camera attached to its front side which provides a different look to the drone. The drone is present in a clear area of the background and can easily be differentiated in similar backgrounds. The sky at backside of the picture also present the model with variations in the background of the drones.

In this 3.18b image, the color of the drone body is not in clear contrast with its immediate surroundings. The drone and its three clearly visible rotors have been annotated in the picture. The body of the drone as well as the camera attached to it at the bottom is more visible and noticeable than the upper part of the drone body. This image will greatly help the model to detect drones in such lighting conditions. The rotor blades are however not plainly visible but have been annotated so that the model recognizes them in such images with little disparity in color schemes between the drone body and the background. It can be seen that only the left rotor of the drone is easily identified



(a) (b)

Figure 3.19: Labelled image of the rotor and the drone



(a) (b)

Figure 3.20: Labelled image of the rotor and the drone

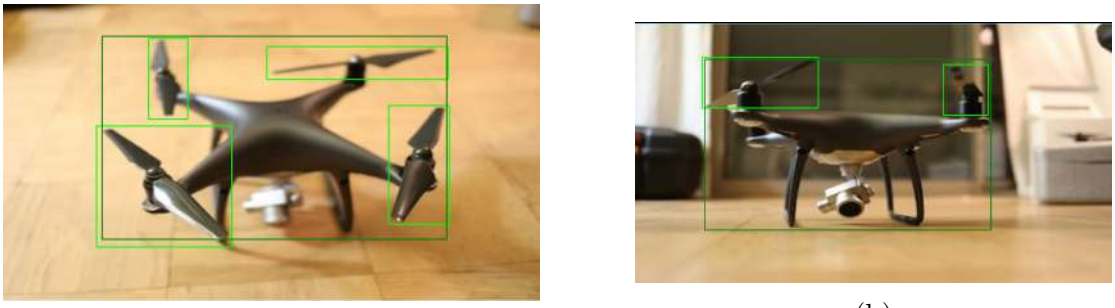
due to the different colors of the vase present in the background.

In these images, we can observe that the drones and their visible rotors have been annotated. The drones are placed on some kind of a platform and we can see the rotors blades are at rest. The rotors can be easily detected from the images. In the first image, the drone is partially covered by the controller and one of the rotors is also not visible due to it. This image can help the model to detect drones in similar positions where some parts are concealed behind other objects. In the second image, a distant view of the drone on the platform is provided. In this picture, we can observe that one of the rotors could not be annotated due to its position with respect to the drone's body in this view. Therefore, this image will enhance the model's detection capabilities in similar scenes.

In these 3.20 images, we can clearly see the drone in the rest position as well as during its flight. The rotors in the 3.20a first picture have been annotated easily because their blades are more recognizable than in the second picture. The body of the drone is also more noticeable in the background. The grassy background will help the model to detect

(a) (b)

Figure 3.21: Labelled image of the rotor and the drone

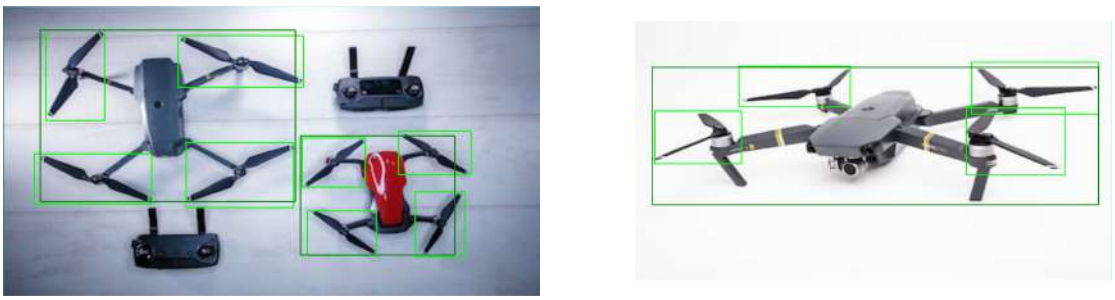


(a) (b)

Figure 3.22: Labelled image of the rotor and the drone

drones in similar scenes with better accuracy. In the 3.20b second image, the close-up view of the drone provides an opportunity for the model to learn about the features of the drone. The rotors are also visible and can be detected without much effort. The drone is easily recognized in the image due to its blurry background. In these 3.21 images, we can see two dissimilar versions of the same image. The grey scale and colored version of the same image are beneficial for the model's learning. The illumination and colors in the first image as well as the background of that image will allow the model to detect drones from such diverse scenes. In the second image, due to deprivation of color and illumination, the model can learn to detect drones with greater precision in scenes that lack color and adequate illumination. The drone and all four visible rotors have been annotated carefully so that the model can learn to localize the object correctly.

In these 3.22 images, the drones are placed on a wooden floor and the close-up view provides details about the drone's features. A camera can also be seen attached to the drone at the bottom. As the bounding box of the drone includes the body as well as the bottom part of the drone including the attachment, the model can learn about these features efficiently. The annotation of rotors will improve the chances that a drone is



(a) (b)

Figure 3.23: Labelled image of the rotor and the drone

(a) (b)

Figure 3.24: Labelled image of the rotor and the drone

detected correctly in similar images. In the 3.22a first image, we can see that although the attachment is not clearly visible, and it is bounded by the dark green outlined box. The rotor blades are also distinct above the drone's body. In the 3.22b second image, the dark object to the left of the drone could also be mistaken as a drone by the model if the rotor annotation was not present in the images.

We can observe two different designs of drones in these 3.18b images. The first image contains top views of two drones that have been annotated cautiously. The rotors on both of these drones are recognizable from a distance as they are not in motion. The controllers near the drones can also be mistaken for our desired targets due to their shapes. However, this will be prevented by the use of two classes in the detection of drones. Another close-up front view of the drone is provided in the 3.23b second image. In this view, the drone and rotors are identified easily and the model will learn to detect them in similar images. We can also see that the legs beneath the rotors act as the boundary at the bottom of the bounded box. The attachment with the body of the drone can also be thoroughly observed in this image.

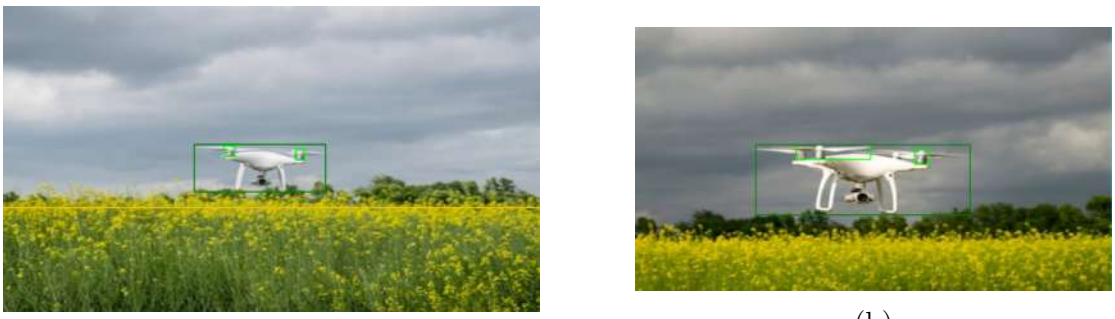
(a) (b)

Figure 3.25: Labelled image of the rotor and the drone

In the 3.24a first image, the drone is difficult to detect and it is being held by a person. The rotors of the drone are also annotated carefully although only two of them are visible. The background of this image as well as the fact that a person is holding the drone will be good information to learn for the model. It will learn to recognize the drone even when it is held close by a person with the help of the boundary boxes. In the 3.24b second image, the drone can be seen hovering at a distance from the person. Its rotors are also not very noticeable. The distant view in addition to the bounding boxes will help the model to identify the drone with such scenes. The sky behind the drone in the background helps to make it prominent and particularly easier to be detected. The images provide interesting and valuable information for the model to learn.

Two slightly distinct views can be seen in these 3.25 images. The drones can be identified easily in the 3.25a first image, however, the background of that image is complex in terms of the diversity in the scene. The drone is placed on a box can its rotors are in motion. The drone and its rotors are annotated correctly which can help the model in localizing and classifying the objects with better precision. A controller can also be seen near the drone, which can also qualify as a target object if the image is not annotated properly. The 3.25b second image appears to be an augmented version of the 3.25a first image. It provides a distorted view of the scene captured in the first image. The rotors are not very clear which leads to the annotation of only three rotors that are partially visible in the image. This image will help the model to recognize the drone in similar images.

In the 3.26a first image, the drone can be seen hovering over a field of yellow flowers. The main part of the drone's body appears in front of the sky and is easily distinguishable. However, the lower part of the body is slightly harder to detect in comparison. The two rotors that are visible in the front view have been annotated carefully. This image will



(a) (b)

Figure 3.26: Labelled image of the rotor and the drone

Figure 3.27: Labelled image of the rotor and the drone

prove to be valuable for the model's learning as it will teach the model to distinguish the entire drone from the fields and other similar objects in the image. In the 3.26b second image, a closer view of the drone is captured above the fields. In this image, the drone stands out from the trees and the field in the background. The two visible rotors have been annotated. The image can help the model to improve its accuracy of detecting comparable scenes.

In these 3.27 3.28 and 3.29 images we get a clear and close-up view of the drone and its rotors. The design and shape of the drone has been captured and annotated completely. In the 3.27 first image the rear view of the drone can be seen with great detail. The attachment of the drone can also be observed from behind. These features can help to identify the drone in rear view images of such drones. The rotors are at rest and distinct in the image. Only one of the rotor could not be annotated due to lack of visibility and its overlap with the drone body. In the 3.28 second picture, we can see that same drone during its flight and the rotors can be identified by the faint dark area above the rotor arms. This view also provides further insight into the drone's size and shape. The background also allows the drone to be distinguishable and noticeable in the

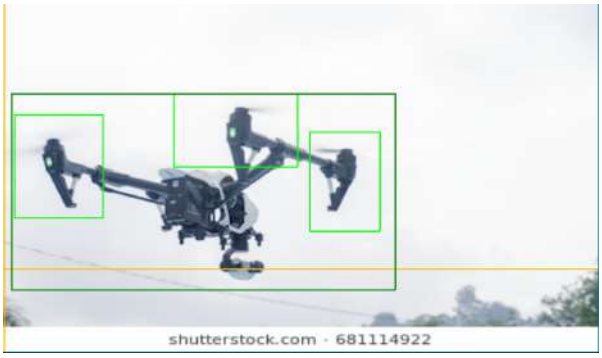


Figure 3.28: Labelled image of the rotor and the drone

Figure 3.29: Labelled image of the rotor and the drone

Figure 3.30: Labelled image of the rotor and the drone

Figure 3.31: Labelled image of the rotor and the drone

image. Both the 3.27 first and the 3.28 second image will help the model to recognize such kinds of drones with great accuracy in faded or plain backgrounds. The 3.29 third picture contains more objects in the background than the 3.27 and 3.28 two previous pictures. This makes it complex and marginally harder to distinguish the drone from the background. The drone as well as its rotors seem to be mixed with the colors in the background due to low contrast in some areas. The body of the drone as well as its attachment can be seen with quite detail. The rotor arms that are attached with the body of the drone, can be seen parked on the ground. This annotated image of the drone and its rotors will help the model to detect the drone more efficiently in similar complex backgrounds.

In these 3.30, 3.31 and 3.32 images we can see the white colored drone in several scenes that include close-up as well as distant views. The color of the drone and the background in the first two pictures(3.30 and 3.31), makes the drone look quite detectable in those scenes. However, the rotors are somewhat difficult to detect as they are in motion during its flight. The trees in the background provide great learning opportunity for the model as they can help to teach the model in detecting drones within similar scenes. The three rotors that are visible have been annotated as well as the entire area covered by the

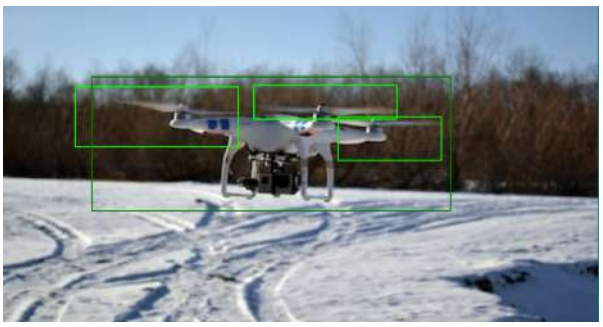


Figure 3.32: Labelled image of the rotor and the drone

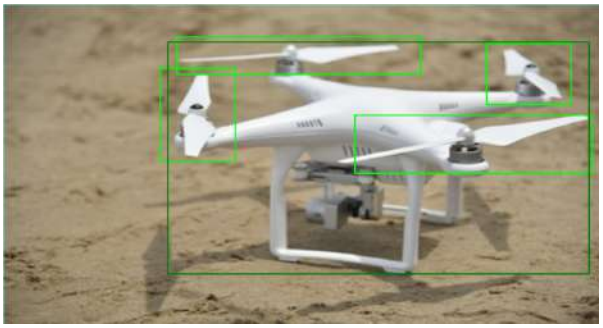


Figure 3.33: Labelled image of the rotor and the drone

drone.

The 3.31 second image is also very similar to the 3.30 first image in terms of the background. The view of the drone is rather distant and tilted as compared to the first picture. This will help the model to detect the drone at different lengths within similar images.

In the 3.32 third image, we can see that the color of the drone matches to a great degree with the snow below the drone. The background containing the trees and the snow covered ground greatly challenges the model to detect a drone in the image. The rotors of the drone are not completely visible in the image however the two rotors that are facing the camera can be detected easily. This images can help the model to detect white drones in partially snowy backgrounds.

These 3.33, 3.34 and 3.35 images provide us with different views of drones that are very valuable for the model's learning. The drones are clearly visible in all of the images. In the first image, we can also spot the rotors as they are not in motion and the blades can be detected separately from the body of the drone. The background enables the drone to stand out in the picture. A part of the shadow is also bounded by the box

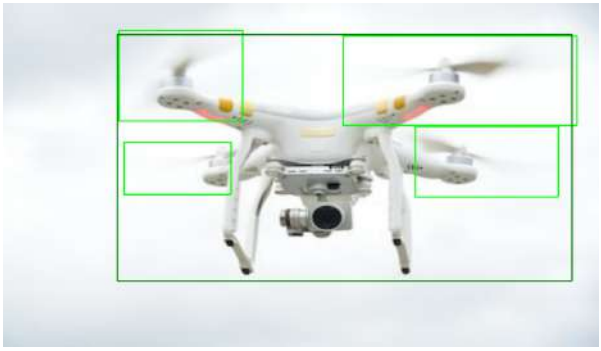


Figure 3.34: Labelled image of the rotor and the drone

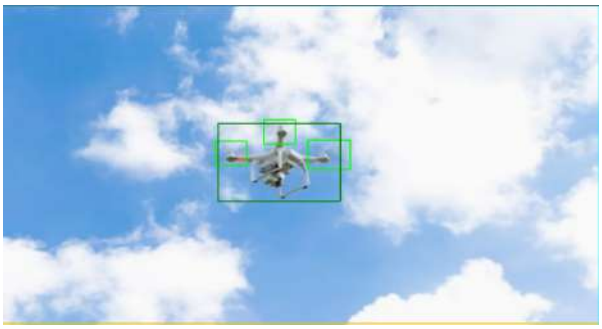


Figure 3.35: Labelled image of the rotor and the drone

which enables the model to learn about similar features of the drones to look for in an image.

In the 3.34 second picture, the bottom view of the drone is shot. Although, the drone is at some distance from the camera, the rotating blades of the rotor can be identified with ease. In this view all four rotors have been annotated and the drone is clearly detectable due to the plain background in the image.

The 3.35 third picture provides a farther view of the drone from the ground. Only three rotors have been annotated with faint indication of their blades above them in the image. We can clearly see the body of the drone can it has been bounded correctly. The background slightly hinders the visibility of the drone and so it acts a learning opportunity for the model. This image will train the model to detect drones in such farther shots.

In these 3.36, 3.37 and 3.38 images we can see some different designs of multi-rotor drones that were not present in previously witnessed images. The drones are smaller in size but are equipped with four rotors as seen previously. These views of the drones

Figure 3.36: Labelled image of the rotor and the drone

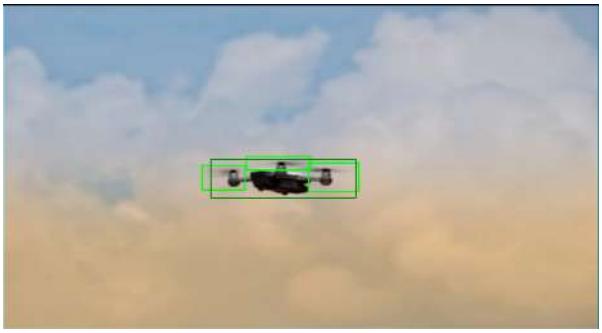


Figure 3.37: Labelled image of the rotor and the drone

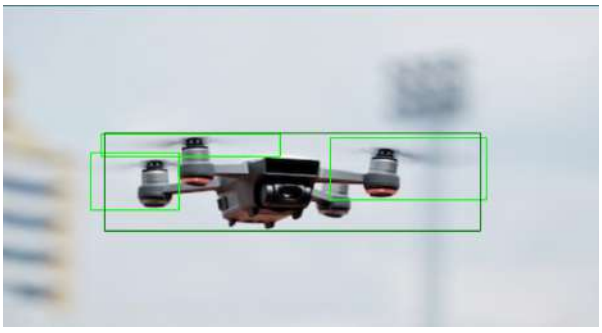


Figure 3.38: Labelled image of the rotor and the drone

Figure 3.39: Labelled image of the drone

are trickier to detect due to the small body and rotor arms. The rotors appear distinct from the body and thus the drone can be detected with the help of the rotors. In the 3.36 first image the drone is sitting on a rough dirt ground and its rotor blades are not very clearly visible from such a distance. The rotor arms have been annotated as well as the entire drone. This picture can help the model to detect drones in images containing similar backgrounds. In the 3.37 second image, we can see the drone over the sky and it is evident that due to the distance and low lighting conditions, the body of the drone is not clearly visible. The drone and rotors have been annotated for the model to learn the detection of these objects in similar scenarios.

The 3.38 third image is a close-up view of the drone during its flight. We can see that the three visible rotors have been annotated and its blades are faintly noticeable due to them being in operation. In this image we can also clearly see the shape and design of the drone's body in the air. The background allows the drone to become the main objective in the picture.

3.5.4 Testing Dataset - Single Class

The testing dataset(one class) is a customized dataset that contains a collection of drone images that were obtained from various sources on the internet. These images were then annotated on the basis of drones with the help of an annotation software known as "LabelImg". The primary aim of building this dataset was to compare the detection accuracy of the annotated Drone-net dataset. The images in this dataset contain multi-rotor drones that vary in number. The drones in the images are present in different backgrounds, angles and lighting conditions. Some of the images from this dataset are presented below.

Figure 3.40: Labelled image of the drone

Figure 3.41: Labelled image of the drone

The above 3.39 image shows a drone flying at relatively a close distance from the ground. The drone body has been annotated by the green bounding box. The drone is located in the center of the image but at such an angle that its fourth rotor is not visible. An attachment in the form of a camera can be seen which is usually for aerial photography, however it can be a security vulnerability as well in some sensitive scenarios. This image is different from the previous images as it has a different background with little clouds, light sky, and trees and can help the model in detecting drones patrolling near the ground.

This 3.40 image shows a drone along with its accessories. The drone present in the top right corner of the image is annotated correctly within the green bounding box. There is a payload attached at the front of the body. The image quality is not so good, however, it gives the model a variation in the training data to detect drones with X-shaped quadcopter frames. T

The above 3.41 image is a close-up of a dark-colored drone. The light green bounding box encloses the drone and there are six visible rotors installed on it. The drone is located very close to the camera and appears as a silhouette or shadow. The drone body is having multiple payloads and the rotors are extended at a distance from the body and are thus distinguishable. Background conditions are rainy and the sky has dark clouds at distances, which provides the model with variations in the training dataset.

Figure 3.42: Labelled image of the drone

Figure 3.43: Labelled image of the drone

The background makes it simpler for the model to detect drones in similar images.

This 3.42 image is different from the rest of the images as it does not only show a drone, rather shows an image of a drone along with its parts and accessories. A light green bounding box is used to annotate the drone as a whole and the four rotors are also visible in the image. In this drone, the rotors are such that each one appears different in physical dimensions from the others which enables the model to learn additional representations. Different parts of the drone such as quadcopter frame, battery, power distribution cables, motors are also shown in the image.

This 3.43 image shows a close-up of a drone with no particular background. There are four rotors that can be seen in the image. The drone is annotated by the light green bounding box. This drone has a different shape from the previous images; the rotors are extended from the body but not much elevated like in the previous images and the drone body is ellipsoid in shape. There is a small camera attached to the body. Such images help the model in learning the sole representations for drone detection as there is no background or other noise to interfere with the detection process.

The 3.44 image above shows a drone present on the ocean. The drone is having three visible rotors as well as a rotor that is not visible in the image. We can see that the entire drone body is annotated by the bright green bounding box. In terms of its body

Figure 3.44: Labelled image of the drone

Figure 3.45: Labelled image of the drone

and payload, the drone is nominal, having a compact body and a camera. The drone is blending with the background at some places in this image. This kind of training data can help the object detection model in recognizing drones that show camouflaging properties in dim environments and thus generalize well.

This 3.45 image shows a stationary drone placed on the ground. The four rotors are clearly visible in the image. The drone is annotated by the light green bounding box. There is a camera attached to the drone and room for placing additional payload on the landing gear. This image differs from the previous ones in that the background is very different and provides variation to the model for learning generalized representations and patterns. An interesting thing to bear in mind is that the light green bounding box also encloses some parts of the joystick which can reduce the performance of the model.

The above 3.46 image shows a close-up of a drone with a darker green background which appears to be blurred. All Four rotors of the drone are visible in the image and the drone itself has been annotated by the bounding box. The rotors are somewhat difficult to identify as they blend with their background, but the rest of the drone body stands out from its background. The drone appears to have a compact body with a small camera attached to it. Such images help the model in detecting drones flying on

Figure 3.46: Labelled image of the drone

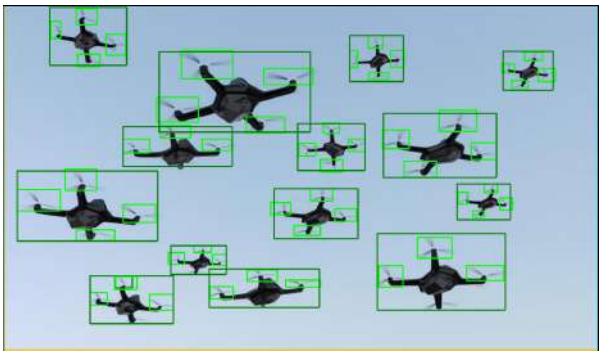


Figure 3.47: Labelled image of the rotor and the drone

areas with greenery and foliage in the background flying at relatively closer distances.

3.5.5 Testing Dataset -Two Classes

The testing dataset is a customized dataset that contains a collection of 31 drone images that were obtained by us from various sources on the internet. These images were then annotated on the basis of drones and rotors. The primary aim of building this dataset was to compare the detection accuracy of the annotated Drone-net dataset with a similar dataset that possesses drone and rotor based annotation. The images of this dataset were annotated using the LabelImg software. The images in this dataset contain multi-rotor drones that vary in number. The drones in the images are present in different backgrounds, angles and lighting conditions. Some of the images from this dataset are presented below.

In this 3.47 image, several drones can be seen at a distance and they are bounded by the dark green outlined box in addition to the light green boxes that bound the rotors of the drone. As some of the drones are far away from the camera, only three of the four rotors can be detected and are identified by the bounding boxes while for some of the

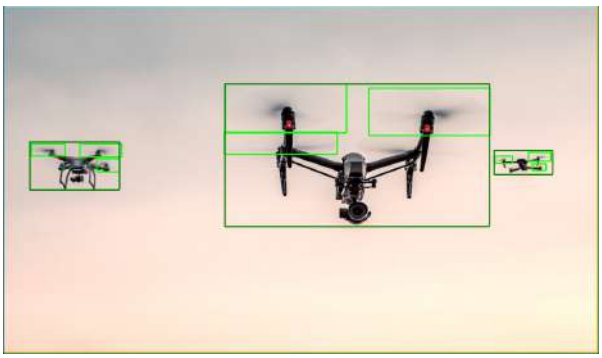


Figure 3.48: Labelled image of the rotor and the drone

Figure 3.49: Labelled image of the rotor and the drone

drones all four rotors can be detected. The bottom part of the bodies of the drones is more visible and noticeable than the upper parts due to the camera angle. The part of the background which contains the drone is a clear blue sky and the drone stands out completely in that area.

This 3.48 image provides a view of three dark-colored drones that have four rotors attached with their bodies through the arms. The drones are bounded by the dark green outlined box in addition to the light green boxes that bound the rotors of the drone. However, only three rotors could be identified and bounded in this image. The rotors can be seen to be in a rotating motion that makes its detection slightly difficult due to the rotor blades appearing almost invisible next to the arms. The drone body also contains a camera attached to its front side which provides a different look to the drone. The bottom part of the body of the drone is more visible and noticeable than the upper part of the drone body. The drone is present in a clear area of the background and can easily be differentiated in similar backgrounds.

In this 3.49 image, the five rotors of the drone are clearly identified by the light green

Figure 3.50: Labelled image of the rotor and the drone

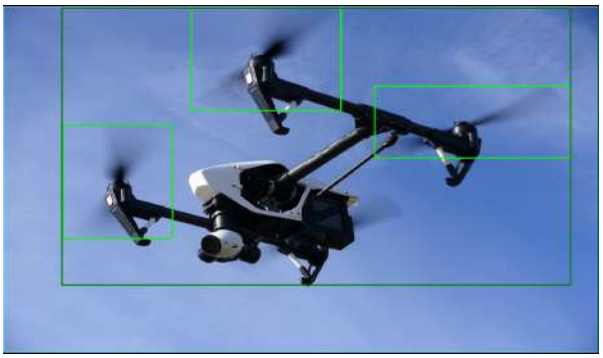


Figure 3.51: Labelled image of the rotor and the drone

bounded boxes but the sixth rotor is not visible. The rotors can be seen to be in a rotating motion but are detected due to their color being different than the arm. As the drone is present near the camera, its features are visible with great detail. The part of the background which contains the drone is a clear blue sky and the drone stands out completely in that area. The drone which is bounded by the box having a dark green outline can also be detected with the help of the arms that hold together the body and upon which the rotors have been installed. The bottom part of the body of the drone is more visible and noticeable than the upper part of the drone body.

This 3.50 image provides a close-up view of the drone. The drone has six rotors that have been annotated. In this image, the six rotors of the drone are clearly identified by the light green bounding boxes. The drone body also contains a camera attached at the bottom which provides the model with an interesting feature to learn. The drone is present in a clear area of blurred green color and can easily be differentiated from the background.

This 3.51 image provides a close-up view of the drone that has four rotors attached

Figure 3.52: Labelled image of the rotor and the drone

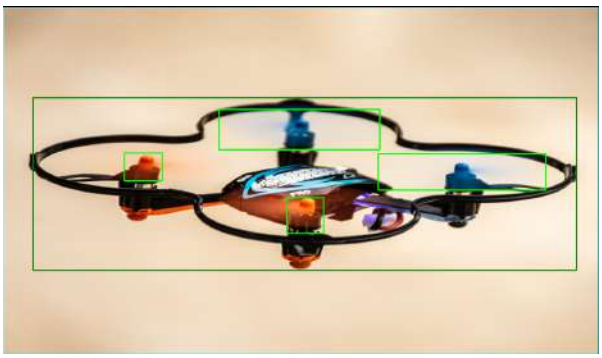


Figure 3.53: Labelled image of the rotor and the drone

with its body that are connected through the unique set of rotor arms. However, only three rotors could be identified and bounded in this image. Since the drone is in flight, the rotor blades are in motion. The drone body also contains a camera attached to its front side which provides a unique appearance to the drone. The drone is present in a clear sky background and the model can learn to distinguish and detect drones in similar scenes with the help of this image.

This 3.52 image provides a close-up view of the drone that has six rotors attached to its body. The arms of the rotors extend away from the body and this makes them a prominent feature of the drone. The rotors are rotating due to which the detection is relatively difficult to achieve. The drone body also contains a camera and several other attachments at the bottom side which gives a bulky appearance to the drone. The bottom part of the drone can be seen in detail. The body of the drone as well as the other equipment attached to it can also be clearly seen in the picture. The plain background will assist in detecting the drone easily in similar images.

The design of the drone in this 3.53 image is slightly different than the previous images.

Figure 3.54: Labelled image of the rotor and the drone

Figure 3.55: Labelled image of the rotor and the drone

The design is more compact and the rotors are bounded on the outside by a petal-like structure. The drone and the rotors have been annotated. We can see that the rotors are not completely visible and therefore they are harder to detect. The background is plain due to which the detection and localization of the drone become easier for the model. This image will help the model to detect such a different kind of drone.

This 3.54 image provides a close-up view of the drone that has four rotors attached with its body through the arms. The drone is bounded by the dark green outlined box in addition to the light green boxes that bound the rotors of the drone. All the four rotors could be identified and bounded in this image. The rotors can be seen to be in a rotating motion but their detection can easily be seen. The drone body also contains a camera attached to its bottom side which provides a different look to the drone. The bottom part of the body of the drone is more visible and noticeable than the upper part of the drone body. The drone is present in a clear blue sky background and can easily be differentiated into similar backgrounds.

In this 3.55 image, the drone can be seen at a distance and it is marked with the help

Figure 3.56: Labelled image of the rotor and the drone

Figure 3.57: Labelled image of the rotor and the drone

of the dark green outlined box. As the drone is far away from the camera, only three of its four rotors can be detected and are annotated. The drone is present in front of a clear blue sky whereas the mountain and the other objects in the scene are away from it. Due to poor lighting and illumination, the drone has a dark shade. This prevents the model from learning about its other features. The background consists of several shades of the sky which will be helpful for the learning of the model.

The drone can be seen clearly in this image. It has been annotated correctly along with its rotors that are visible in the 3.56 image. The arms are quite noticeable alongside the body. However, only three rotors could be identified and bounded in this image. Due to the camera angle, three rotors are visible. This will help the model recognize drones with similar features and conditions. The drone body also contains a camera attached to its distinct front side. The drone is present in a partially clear area of the background and it will not be particularly difficult for the model to detect it correctly. The trees behind the drone also present the model with a variation in the background.

In this 3.57 image, the outline and shape of the drone can be seen clearly. However, due

Figure 3.58: Labelled image of the rotor and the drone

to the effect of the sunlight on the drone, it has become dark and thus its features are not visible. This will teach the model to detect the drone from its shadow or shape outline. The three rotors are in motion but due to that, they are more noticeable because of the contrast offered by the background. This picture has a different shade of the sky in comparison with some of the previous images. The background will assist the model to detect drones within images that contain the sun as well as dark shadows of other objects.

The drone in this 3.58 image is at a considerable distance from the camera and only a few rotors of this drone could be successfully bounded with the help of the light green outlined boxes. The drone body also contains attachments at the bottom of its body. The bottom part of the body of the drone is more visible and noticeable than the upper part of the drone body. The drone is present in a clear bluish area of the background. The scene contains a person close by and several trees in the background. This can help the model to detect the drone in such scenes that contain many different objects within its frame. The slightly faded colors of the sky make the drone stand out in comparison to the other objects present in the picture.

The drone captured in this 3.59 image is equipped with six rotors. These rotors are attached to the body through the arms that are particularly visible in the picture. Due to the camera angle, all the six rotors could be identified and bounded in this image. We can also see the attachments at the bottom of the drone. The drone also has noticeable features that can be recognized by the model. The bottom part of the drone is the main focus of this image. The background allows the model to detect the drone easily by identifying the rotors and the body.

In this 3.60 image, the drones can be observed in contrasting and diverse scenarios. The

Figure 3.59: Labelled image of the rotor and the drone

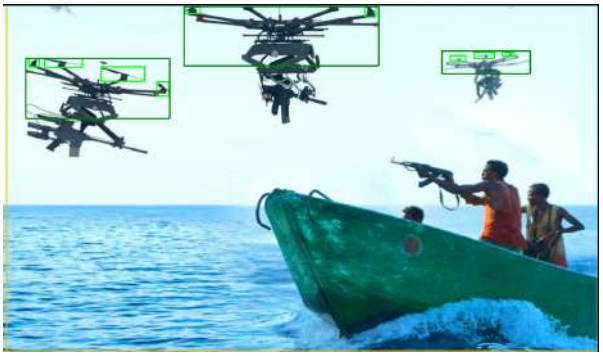


Figure 3.60: Labelled image of the rotor and the drone

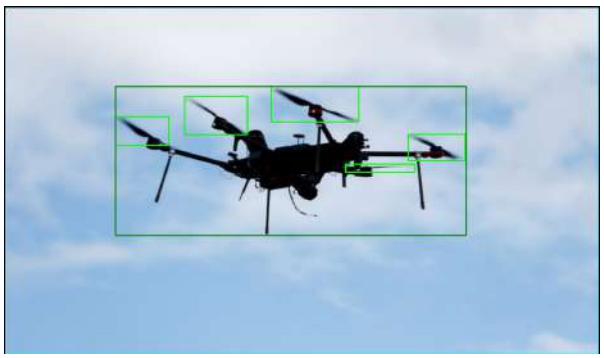


Figure 3.61: Labelled image of the rotor and the drone

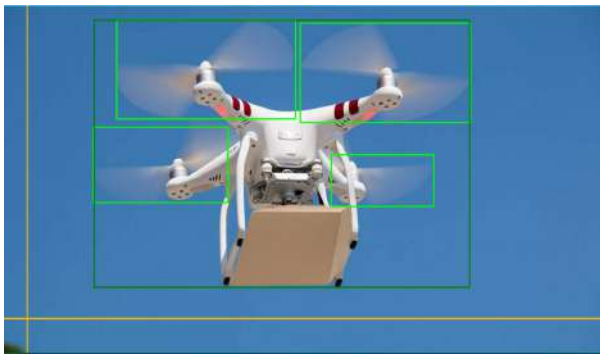


Figure 3.62: Labelled image of the rotor and the drone

drones are present near as well as farther away from the point of view of the camera. We can also see three similar drones and their rotors in this image. The rotors of these drones have also been annotated although some of the rotors are partially covered by the bounding box. The drones appear to be carrying weapons but the bounding boxes cover only the main parts of the drone. The background is different in comparison with previous images. This image will help the model to detect drones in similar images. Several other objects can be detected as drones in this image but the annotation will prevent that with the help of the bounding boxes.

The above 3.61 image depicts a drone with five rotors. Major parts annotated are the drone body and the individual rotors. The drone is marked in almost the center of the image and is present at a close distance to the camera. In this image, the weather conditions are cloudy and the sky is very light, making the drone appear as a silhouette which makes it tricky for the object detection model. This is because in such background conditions, different aerial objects such as planes, birds, trees appear as silhouettes and thus there is a larger risk of false-positive predictions.

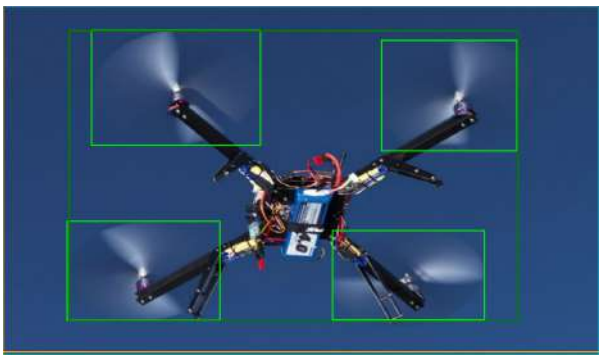


Figure 3.63: Labelled image of the rotor and the drone

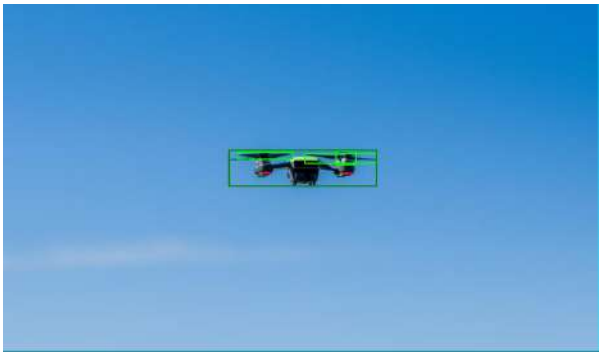


Figure 3.64: Labelled image of the rotor and the drone

The above 3.62 image shows a drone with four rotors annotated by bounding boxes. The drone is placed very close to the camera and in the center of the image and is carrying a box as a payload. The drone body in this image is very compact; such variations in the training dataset help the object detection model to find representations that generalize better on deployment and reduce overfitting. The weather conditions in this image are bright and clear with essentially no difficulty related to the background for the object detection model since the drone completely stands out from its background.

The above 3.63 image shows a drone having four rotors and a very minimalist body. It is placed in the center at a very close distance to the camera. In this image, the background conditions are dark but still bright enough for the camera not to lose the image and color quality and the drone does not appear as a silhouette. Due to the high picture quality and closer distance, there is a lesser chance of false positives in such images, but a higher risk of overfitting. Such images help the model to detect drones placed at closer distances.

The above 3.64 image shows a drone flying at a considerably farther distance from the

Figure 3.65: Labelled image of the rotor and the drone

Figure 3.66: Labelled image of the rotor and the drone

camera. The fact that the drone is not exactly in the center of the image unlike the above images helps the model in generalizing. Notice that the drone body and individual rotors are far from the camera so the bounding boxes are very small compared to the previous images. There is a clear sky with very few clouds in this image, and the darker body of the drone is distinct from its background. Such images help the model in detecting drones that are far from the camera.

This 3.65 image shows a drone flying at a considerably closer distance from the ground at an inclined angle from the camera. The drone body and three of its rotors are annotated. There is an urban background in the image with noise in the background, which can be tricky for object detection. In such conditions, there is a higher risk of false-positive predictions as other objects such as the tree crowns or the lampposts can also be detected as drones, so such images in the training dataset are helpful for the object detection model to learn and differentiate the exact representations for drones.

The above 3.66 image shows a drone flying at relatively a close distance from the ground. The drone body and three rotors are annotated by green bounding boxes. The drone

Figure 3.67: Labelled image of the rotor and the drone

is located in the center but at such an angle that its fourth rotor is not visible. An attachment in the form of a camera can be seen which is usually for aerial photography, however it can be a security vulnerability as well in some sensitive scenarios. This image is different from the previous images as it has a different background with little clouds, light sky, and trees and can help the model in detecting drones patrolling near the ground.

The above 3.67 image shows a close-up of a drone with a darker green background which is blurred. Four rotors and the drone have been annotated by the bounding boxes. The rotors are somewhat difficult to identify as they blend with their background, but the rest of the drone body stands out from its background. The drone appears to have a compact body with a small camera attached to it. Such images help the model in detecting drones flying on areas with greenery and foliage in the background flying at relatively closer distances.

3.5.6 How to train to detect custom objects:

YOLO (V4 and V3):

1. For training, download the pre-trained weights-file (162 MB). [1]
2. Create a configuration file named Yolo-obj.cfg. This file and the yolov4-custom.cfg both have the same content.
3. Copy the content in yolov4-custom.cfg file and:

[1] Yolov4 initial weights `https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137`

- Set batch=64 instead of the line batch
- Set subdivisions=17 instead of the line subdivisions
- Set max_batches = classes*2000 = 4000 (number should not be greater than 6000 and not lesser than the number of training images) instead of the line max_batches. For example, max_batches=4000 is valid if the model is trained for 2 classes
- steps should be replaced by 80 to 90 percent of max_batches parameter. steps=3200 or steps=3600 are valid choices width and height (network size) should be set to any multiple of 32, for example, width and height can be set to 416
- The line classes=80 should be changed to accommodate the number of objects in the model. In our case, it should be classes=2. This should be done for each of the 3 yolo layers.
- Set filters=(classes+5)x3 instead of filters=255 before each yolo layer in the 3 convolutional layers. In our case, this will be filters= (2+5) x3 = 21. This should be done only for the last convolutional layer before each yolo layer.
- Set filters=(classes+9)x3 instead of filters=57 when using Gaussian_yolo layers.
- This is done for the 3 convolutional layers before each Gaussian_yolo layer and results in 33 in our case (filters=(2+9)x3).

4. In the directory `build‣darknet‣x64‣data` , create a new file named obj.name. This file contains the object names, separated by newlines.
5. In the same directory, create another file named obj.data containing: classes = 2 train = data/train.txt valid = data/test.txt names = data/obj.names backup = backup/ (classes are the number of objects)
6. Move image files (.jpg) of the objects in the directory `build‣darknet‣x64‣data‣obj`

7. Label each object on the images in the dataset. For this purpose, the visual GUI software Yolo_mark `https://github.com/AlexeyAB/Yolo_mark` can be used for making bounding boxes and generating annotation files. This software works for both yolo v2 and v3.

8. Make a file named train.txt in the directory `build‣darknet‣x64‣data` having the filenames for the images in the dataset. Filenames are separated by newlines, and the path is set relative to darknet.exe.

9. Move the pre-trained weights for convolutional layers to the directory `build‣darknet‣x64`

10. Start training using the command line. For Linux, the command

    ```
    $ ./darknet detector train data/obj.data yolo-obj.cfg yolov4.conv.137
    ```

    can be used to initiate training

11. Training with mean average precision (mAP) for every fourth epoch can be started instead by setting valid=valid.txt or train.txt in the obj.data file and passing the –map command-line argument.

    ```
    $ ./darknet detector train data/obj.data yolo-obj.cfg yolov4.conv.137
    ```

12. The resulting weights are stored in the yolo-obj_final.weights file at path `build‣darknet‣x64‣backup`

13. These weight files can be used to train the model in batches. Training can be resumed later using the weight file to continue training from where it stopped before.

4 Evaluation

In this section the evaluation obtained with Performance metrics.Performance metrics are a measure of how successful the results are in classification problems.after definions of confusion metrics mentioned are provided.

4.1 Accuracy

Accuracy gives a summary of a model's performance and is sufficient in cases where only the overall correctness of the algorithm is important and a detailed breakdown of results is not required. Accuracy has a simple scoring strategy which can be shown by the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

4.2 Confusion Matrix

A confusion matrix provides a detailed report of a classifier's performance. It describes the model's performance on each of the classes. A confusion matrix is a tabular representation of the comparison between predictions and actual outputs

4.2.1 True Positives

These are the data points that are correctly predicted as positive.

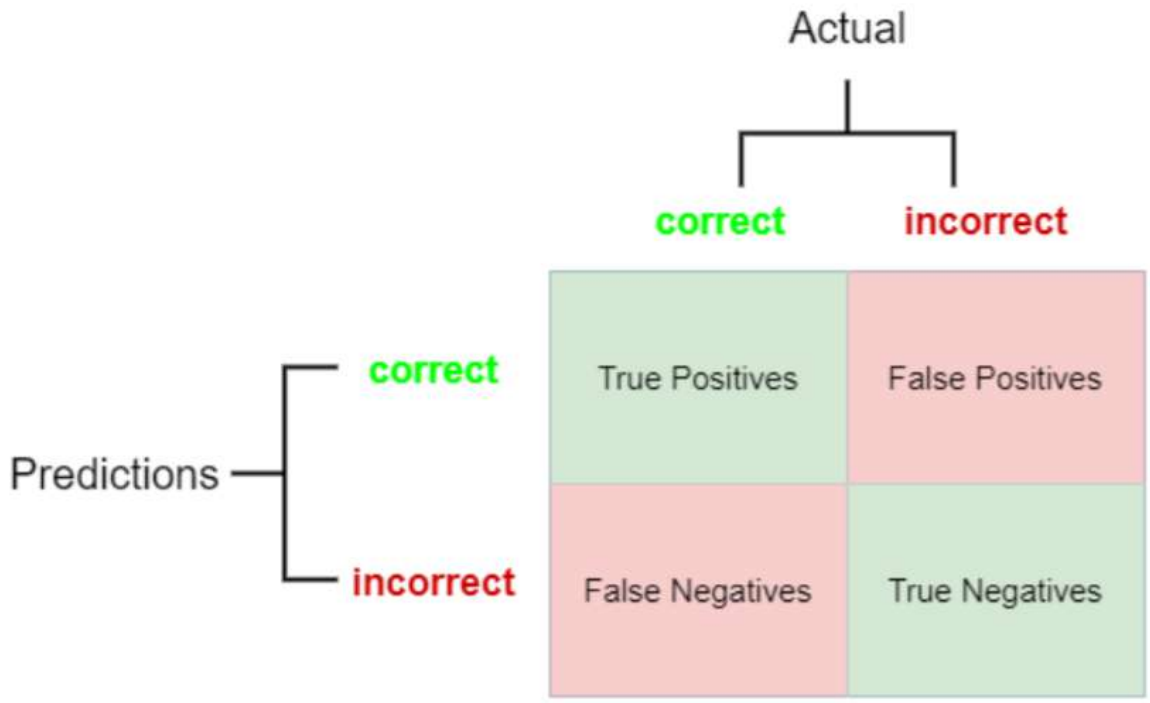


Figure 4.1: Confusion Matrix

4.2.2 True Negatives

The data points that are correctly classified as negative.

4.2.3 False Positives

These are the data points that are negative in reality but are classified as positive.

4.2.4 False Negatives

These are the data points that are incorrectly classified as negative. Our aim is to minimize the false positive and false negative score, however it depends on the problem statement to decide whether to focus on minimizing false positive or false negative more.

4.3 Precision and Recall

Precision and Recall are summary statistics calculated by the combination of true positives, true negatives, false positives and false positives.

4.3.1 Precision

tells how likely it is for a positive prediction to be correct. It is the ratio of true positives to total positives. Precision is a good measure to use when we care about false positives more than false negatives for example in email classification, classifying non-spam email as spam (false positive) might result in loss of important information.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

4.3.2 Recall

is essentially the opposite of precision, that is, it is a measure of how many true positives are predicted as positive. It is the ratio of total positives to all positives. Recall is given more importance when we care more about false negatives, for instance if a sick patient is classified as non-sick (false negative) is a scenario where false negative score has more associated cost.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

4.4 F1 score

F1 score is a function of precision and recall and provides a balance between both the scores especially when the dataset is imbalanced.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4.4}$$

4.5 Intersect over union (IoU) score

IoU is a metric that allows us to determine how close the predicted bounding box and actual bounding box are. Mathematically, it is the ratio of overlapping areas of two boxes to their combined area. IoU scores are between 0 and 1, where closer to 1 score means higher precision.

5 Results

In this section the results obtained after comparison of datasets mentioned in the table below are provided. The comparison was based on the results of the accuracy of detection of drones as well as rotors for the datasets that were annotated. It can also be observed that the accuracy of drone detection of the Rotor-Drone dataset is significantly greater than that of the original Drone-net dataset.

5.1 Drone net dataset and Testing dataset (Single Class)

On the drone net dataset, YOLOv4 trained on one class resulted in 0.97 precision score which means that over 96% of the positive predictions were correctly classified. There were 258 true positives, 8 false positives and 8 false negatives. Recall score and F1 score were 0.97. The average IoU score came out to be 78.6%.

Single class classification on the test dataset trained on best weights resulted in 0.97 for precision, recall and F1 scores. There were 259 true positives, 7 false positives and 7 false negatives with an average IoU score of 78.03% and mean average precision (mAP) score of 97.15%.

Table 5.1: Results comparsion

	Drone-net dataset	**Testing dataset (single class)**	**Rotor-Based drone-net advance dataset (RBDNA) two classes**	**Testing dataset (two classes)**
Drone detection	97.03%	97.15%	97.52%	98.86%
Rotor detection	-	-	79.19%	76.22%
Layer	161	161	161	161
Yolo	4	4	4	4

5.2 Drone-net extension dataset and Testing dataset (Two Classes)

On the Rotor drone binary classification, precision was around 0.80, and recall was 0.83. The F1 score was 0.82. There were 936, 228 and 194 true positives, false positives and false negatives respectively with an IoU score of 64.61. The mean average precision (mAP) score was 88.36%.

Binary classification on the testing dataset trained on best weights resulted in a classifier that was 87.54% precision on average with 60.27% IoU score. The presion, recall and F1 scores were 0.77, 0.80 and 0.79 respectively with 911 true positives, 265 false positives and 221 false negatives.

6 Discussion

In this chapter, we will analyze the results and discuss the methods and techniques that were implemented to achieve these results.

The primary objective of our research was to accurately detect multi-rotor drones with the help of a deep learning model. In order to achieve this task, we chose the YOLOv4 model for our implementation. The next step was to find a dataset that could be used to train the model so that it should detect multi-rotor drones accurately. One dataset called "Drone-net" was relevant to our domain, and it provided drone images with a single (drone) class annotation. This dataset delivered sufficient accuracy when we trained our model with it. However, we aimed to improve this accuracy so that the detection can be carried out with better accuracy.

For this purpose, we used a technique called "Fine-Grain Labelling" [Che+18]. A label can sometimes be defined in many ways according to its application. This can be understood using the following example of drones, an image that contains a multi-rotor drone can be labeled as a "drone", similar to what had been done in the Drone-net dataset. This same image can also be labeled with a drone class and a rotor class, thus adding more specifications to the drone. This concept can be referred to as label granularity [Che+18]. The experimental results after fine-grain labeling show that the network's optimization and generalization capabilities were improved significantly. The reason behind this improvement is that it provokes the network to acquire more information about the features which results in better detection accuracy. It has also been observed that by using fine-grain labels, 40% of the total training data can achieve higher accuracy than a model that has been trained with a complete dataset that does not have fine-grain labels [Che+18].

Therefore, these observations compelled us to enhance the accuracy of our model by applying this technique to the existing Drone-net dataset manually. Thus, we used images from the Drone-net dataset and labeled them manually based on two classes;

rotor and drone. The labeling was done with the help of a software called “Labelimg” which is a popular graphical image annotation tool.

The modified dataset that we created by annotating the existing dataset was called the “Rotor-Drone” Dataset. This dataset was used to train our model in order to achieve better accuracy in our detection.

We also created a custom dataset that was meant to test the model for possible overfitting due to its training on the Rotor-Drone dataset. This custom dataset consisted of random multi-rotor drone images that were gathered from the internet. Similar to the Rotor-Drone dataset, the images of this dataset were manually annotated by the software based on the rotor and drone classes. Then we used this dataset for testing our model and got good results.

The outcome of our experiments was very positive as evident from the results. We achieved significant improvement in the accuracy of detection by training our model on the dual-class Rotor-Drone dataset that we created. The model also presented reasonable results when it was tested on our custom testing dataset.

7 Conclusion

In this concluding chapter of our thesis, a review of our accomplishments and results will be discussed along with the recommendations for future work.

The detection and classification of drones with better accuracy as compared to the pre-existing datasets, requires the availability of a specific dataset that contains drone and rotor classes. Due to the unavailability of such a dataset, we created a new dataset by annotating a pre-existing dataset based on the drone and rotor.

Thus, the newly created dataset consisted of the same multi-rotor drone images in the pre-existing dataset but was annotated with two classes. This dataset provided better accuracy as compared to the pre-existing drone dataset that contained only a single class.

In addition to creating this dataset, we also created another custom dataset containing images of multi-rotor drones that were obtained from various sources on the internet. These images were also annotated based on the drone and rotor similar to our previously created dataset. The main purpose of creating this dataset was to test our YOLO model for possible overfitting that might have happened due to its training on the previously created dataset.

The detection accuracy resulting from the Rotor-Drone dataset has been proved to be better than that of the pre-existing dataset. These results have been discussed in detail within the results section. The creation of a drone images dataset with rotor and drone classes is a great contribution to the scientific world. This dataset can be effectively used to detect drones with greater accuracy due to the drone and rotor class annotations.

7.1 Recommendations for future work

In the future, our model which has been trained on the multi-rotor drone dataset can be trained on a dataset containing images of fixed-wing drones with two classes, one for the drone and another for the rotor. This can be done with the help of transfer learning techniques and it can broaden the detection capability of our model, to include fixed-wing drones.

A Code

B Math

C Dataset

Bibliography

[19] Name *. *7 applications of computer vision*. Oct. 2019. URL: https://www.atriainnovation.com/en/7-applications-of-computer-vision/ (cit. on p. 12).

[Adm] Federal Aviation Administration(FAA). *FAA National Forecast FY 2019-2039 Full Forecast Document and Tables*. URL: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2019-39_FAA_Aerospace_Forecast.pdf (cit. on p. 2).

[21a] *Artificial neural network*. June 2021. URL: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1026722813 (cit. on pp. 6, 7).

[] *Back Propagation Neural Network: What is Backpropagation Algorithm in Machine Learning?* URL: https://www.guru99.com/backpropogation-neural-network.html (cit. on p. 9).

[Bro20] Jason Brownlee. *How to Choose Loss Functions When Training Deep Learning Neural Networks*. Aug. 2020. URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/ (cit. on p. 6).

[Bus20] Frederik Bussler. *Apple's M1 Chip is Exactly What Machine Learning Needs*. Dec. 2020. URL: https://medium.datadriveninvestor.com/apples-m1-chip-is-exactly-what-machine-learning-needs-507db0d646ae (cit. on p. 10).

[Che+18] Zhuo Chen et al. "Understanding the impact of label granularity on cnn-based image classification". In: *2018 IEEE international conference on data mining workshops (ICDMW)*. IEEE. 2018, pp. 895–904 (cit. on p. 76).

[Cho17] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2017 (cit. on pp. 11, 21, 23).

[Fuk88] K. Fukushima. "Neocognitron: A hierarchical neural network capable of visual pattern recognition". In: *Neural Networks* 1 (1988), pp. 119–130 (cit. on pp. 22–24).

[Goo+16] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016 (cit. on pp. 10, 11, 24).

[15] *Japan radioactive drone: Tokyo police arrest man*. 2015. URL: https://www.bbc.com/news/world-asia-32465624 (cit. on p. 2).

[LeC+89] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541 (cit. on pp. 22, 24).

[Mst20] Pulkit SharmaMy research interests lies in the field of Machine Learning, Deep Learning. Possess an enthusiasm for learning new skills, and technologies. *Image Segmentation: Types Of Image Segmentation*. Oct. 2020. URL: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/ (cit. on p. 14).

[] *mcculloch-pitts neurons*. URL: https://mind.ilstu.edu/curriculum/mcp_neurons/index.html (cit. on p. 6).

[Mee21] Vidushi Meel. *ANN and CNN: Analyzing Differences and Similarities*. June 2021. URL: https://viso.ai/deep-learning/ann-and-cnn-analyzing-differences-and-similarities/ (cit. on p. 21).

[Red16] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. 2016 (cit. on p. 35).

[Red17a] Joseph Redmon. *DroneNet Dataset*. 2017. URL: https://github.com/chuanenlin/drone-net (cit. on p. 35).

[Red17b] Joseph Redmon. *DroneNet Dataset*. 2017. URL: https://github.com/chuanenlin/drone-net/issues/1 (cit. on p. 36).

[RM07] Alessandro Rizzi and John McCann. "Computer algorithms that mimic human vision must respond to the spatial content in images". In: *SPIE Newsroom, doi* 10.2.1200705 (2007), p. 0675 (cit. on p. 6).

[Rus+15] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252 (cit. on p. 25).

[SZ15] K. Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2015) (cit. on pp. 24, 25).

[Sri+14] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15 (2014), pp. 1929–1958 (cit. on p. 24).

[21b] *Tensor Processing Unit*. July 2021. URL: `https://en.wikipedia.org/w/index.php?title=Tensor_Processing_Unit&oldid=1029236102` (cit. on p. 10).

[Yan+16] Bin Yang et al. "CRAFT Objects From Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016 (cit. on p. 25).

[Zen+16] Xingyu Zeng et al. *Crafting GBD-Net for Object Detection*. 2016. arXiv: `1610.02579 [cs.CV]` (cit. on p. 25).

Eidesstattliche Erklärung

Passau, October 24, 2021

Syed Ayaz