REAL-TIME OBJECT TRACKING USING ARTIFICIAL INTELLIGENCE

A Project report submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

P. Rajasri (319126512042) M. Deepika (319126512033)

S. Yeswanth (319126512056) V. Krishna Vamsi (319126512064)

Under the guidance of

Dr. B. Jagadeesh

Professor, Department of ECE

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade) Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)

2022-2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERINGING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, *Bheemili mandal, Visakhapatnam dist. (A.P)*

PRAGNANAM BRAHMA
ANITS

CERTIFICATE

This is to certify that the project report entitled **"REAL TIME OBJECT TRACKING USING ARTIFICIAL INTELLIGENCE"** submitted *by **P Rajasri (319126512042), M Deepika (319126512033), S Yeswanth (319126512056),V Krishnavamsi (319126512064)*** in partial fulfilment of the requirements for the award of the degree *of **Bachelor of Engineering in Electronics & Communication Engineering*** of Andhra University, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project guide	**Head of the department**
[illegible]	[illegible]
Dr. B. Jagadeesh	**Dr. B. Jagadeesh**
B.E, M.E, Ph.D., FIE, FIETE, MIEEE.	**B.E, M.E, Ph.D., FIE, FIETE, MIEEE.**
Professor, Department of ECE	**Professor, Department of ECE**
ANITS	**ANITS**
Professor Department of E.C.E. Anil Neerukonda Institute of Technology & Sciences Sangivalasa, Visakhapatnam-531162	**Head of the Department** Department of E C E Anil Neerukonda Institute of Technology & Sciences Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Dr. B. Jagadeesh**, Professor, Department of Electronics and Communication Engineering, ANITS, for his guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. B. Jagadeesh** Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank all **non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

Project students:
P. Rajasri (319126512042)
M. Deepika (319126512033)
S. Yeswanth (319126512056)
V. Krishna Vamsi (319126512064)

CONTENTS

TITLE **Page.No**

LIST OF FIGURES

ABSTRACT

Modern world is drenched with an enormous amount of visual data. Object tracking is the keystone of many computer vision applications. Identifying objects from an image or a video sequence is a primary and demanding task in today's world. The purpose of visual object tracking in successive video frames is to detect or connect target objects. It has many applications in the real world, including surveillance, face detection medical image processing, traffic control and analysis, and many more. While many approaches and breakthroughs in this field have led to the evolution of a huge set of unique algorithms, it remains a very stimulating problem. Due to the ginormous set of environmental and other factors, it is next to impossible to propose a global tracking algorithm. However, the selecting the most suitable algorithm does not depend only on the concept of the algorithm but also on its implementation. In this paper, we try to use YOLO or "You Only Look Once" with several configurations of the moving image feature to recognize objects. Additionally, we suggest a system that, by adaptively regulating the cycle of object detection and tracking, can provide real-time performance in different edge computing contexts.

Keywords: Object, detection, tracking, Artificial intelligence, Open CV, python

CHAPTER 1

INTRODUCTION

1.1 PROJECT OBJECTIVE:

Object detection is a crucial task that involves finding one or more objects of interest in still images or video data. It encompasses a wide range of methods, including image processing, pattern recognition, artificial intelligence, and machine learning, to accurately identify and locate objects within an image or video frame, it offers a wide range of potential applications, including improved human-computer interaction, monitoring militarily restricted areas, alerts of harmful commodities in manufacturing, and preventing traffic accidents. Balance between accuracy and computing costs is a challenge since multi-target detection application scenarios in the real world are frequently complicated and varied.

However, achieving a balance between accuracy and computing costs is a significant challenge in object detection. Real-world scenarios can be complex and varied, with multiple targets and varying environmental conditions, which can affect the accuracy and efficiency of object detection algorithms. Finding the right trade-off between accuracy and computational resources is essential to ensure that object detection systems are practical and effective in real-world applications.

1.2 PROJECT OUTLINE:

Object tracking is increasingly important as technology advances daily. In autonomous driving, multiple object tracking is crucial because it allows for the detection and prediction of other vehicles' and pedestrians' movements in the actual world. Blind persons have many difficulties, yet this real-time tracking enables them to reach their destination independently. Drivers find it difficult to see items on the track in a railway system at night, thus by using real-time object detection, we can easily find the objects and avoid railway accidents. Therefore, we made the decision to perform the REAL-TIME OBJECT TRACKING USING ARTIFICIAL INTELLIGENCE. We read a few publications in this field because we are interested in this project. As a result, we were extremely inspired to create a system that could identify and track things in a real-time setting.

CHAPTER 2

OBJECT DETECTION AND TRACKING

2.1INTRODUCTION:

Object detection and image recognition are related but distinct computer vision techniques. Object detection goes beyond image recognition by not only identifying objects in an image but also predicting their locations with bounding boxes. Object detection algorithms use machine learning or deep learning techniques to analyze images or videos and identify objects of interest, and then draw bounding boxes around them to indicate their precise locations. These bounding boxes are often accompanied by labels that describe the objects detected, providing more detailed information about the objects in the image compared to image recognition, which simply assigns labels to entire images without identifying specific objects or their locations. Object detection is widely used in various applications such as autonomous vehicles, surveillance systems, facial recognition, and augmented reality, among others.

2.1.1 Modes and types of object detection:

Traditional machine learning-based methods for object detection often involve handcrafted feature engineering, where various aspects of an image, such as color histograms or edges, are extracted as features, and then these features are used as input into a separate regression model to predict the location of objects and their labels. These methods can work well in certain scenarios but may require manual tuning and are limited by the effectiveness of the handcrafted features.

In contrast, deep learning-based techniques, such as convolutional neural networks (CNNs), have emerged as state-of-the-art approaches for object detection. CNNs are capable of automatically learning relevant features from raw image data during the training process, eliminating the need for handcrafted features. CNNs can learn complex patterns and representations from large amounts of labeled data, making them highly effective in detecting objects with high accuracy.

2.1.2 Why is object detection important?

In that it enables us to comprehend and evaluate scenes in photos or videos, object detection is closely tied to other related computer vision techniques like image recognition and image segmentation.

Deep learning-based object detection techniques, such as region-based CNNs (R-CNN), You Only Look Once (YOLO), and Single Shot MultiBox Detector (SSD), have become state-of-the-art methods for accurate and real-time object detection in various applications, such as autonomous driving, surveillance, and image retrieval systems, among others. They offer higher accuracy and faster processing speeds compared to traditional ML-based methods, making them the preferred choice in many practical scenarios.

What but there are significant variations. Image segmentation develops a pixel-level comprehension of a scene's elements while image recognition just produces a class label for an identified object. The distinctive capacity of object detection to locate items inside an image or video sets it apart from these other tasks. This enables us to count such things and later track them.

- Crowd counting
- Self-driving cars
- Video surveillance
- Face detection
- Anomaly detection

2.1.3 INTRODUCTION TO OBJECT TRACKING:

Finding and following a particular object over time in a video or picture sequence is called object tracking. It entails identifying the object in each frame, connecting it to the appropriate object across frames, and calculating its velocity and other attributes.

Object tracking has a wide range of uses, such as in robotics, augmented reality, self-driving automobiles, and video surveillance. There are numerous methods for tracking objects, from straightforward motion-based strategies to more sophisticated deep learning systems. Dealing with occlusions, changes in appearance, and monitoring several objects at once are a few of the difficulties in object tracking.

2.2 DIGITIAL IMAGE PROCESSING:

The spectrum of computerized picture preparation is represented by the need for extensive test effort to increase the viability of suggested solutions for a particular problem. The extensive amount of testing and experimentation that is typically required before arriving at an acceptable solution is a key characteristic concealed in the design of picture preparation frameworks. This trademark explains that it generally takes a significant amount of planning and quick modelling to reduce the time and cost needed to arrive at an appropriate framework execution.

2.2.1 What is DIP?

Digital image processing is a field of study that involves analyzing, modifying, enhancing, and interpreting digital images using various algorithms and techniques. It has many benefits over analog image processing, as it allows for a wider range of algorithms to be applied to the input data, and it can prevent issues like noise and distortion from accumulating during processing.

Digital image processing is a multidimensional field, as images can exist in two dimensions (e.g., grayscale or color images) or even more dimensions (e.g., volumetric medical images). It has been greatly influenced by advancements in computers, which have provided the computational power necessary for processing large amounts of image data. Additionally, the development of discrete mathematics theory has played a crucial role in the advancement of digital image processing algorithms and techniques.

Digital image processing has a wide range of applications, including computer vision tasks such as object detection, image recognition, and image segmentation, as well as remote sensing, medical imaging, multimedia processing, and many other fields. It has revolutionized many industries and has become an essential tool in various scientific, industrial, and commercial applications.

2.2.2 What is an image?

Images can be captured and saved using a variety of media, including pictures, paintings, drawings, and digital data. Images are a visual depiction of an object or scene.

An picture is often represented in digital form as a rectangular array of pixels, each of which has a unique colour or grayscale value that affects how the image looks as a whole. An picture's resolution is determined by how many pixels make up the image; higher resolution images have more pixels and, consequently, more detail.

Information that can be conveyed through images includes patterns, textures, colors, and the look of things or scenes. They are employed in numerous fields, including those of photography, videography, computer graphics, and medical imaging processing computer vision applications

Fig 2.2.2 Digital image

Processing of an image:

Processing of an image involves three levels. They are low level, medium level, high level.

Low level processing:

- Pre-processing to remove noise.
- Contrast enhancement
- Image sharpening

Medium level processing:

- Segmentation
- Edge detection
- Object extraction

High level processing:

- Image analysis
- Scene interpretation

2.2.3 Why image processing?

Various uses for image processing exist, depending on the application. The most frequent causes for using image processing are listed below:

- Image processing can be used to enhance an image's visual quality, for instance by lowering noise, boosting contrast, or adjusting colour balance.

- Information extraction: Image processing can be used to extract information from an image, including the identification of objects or regions of interest, measurement of attributes like length or area, and the detection of patterns or features.
- Compression: Image processing can be used to reduce the size of image data, which is essential for effectively transmitting and storing huge volumes of image data.
- Identification and recognition: Image processing can be used for tasks like object, licence plate, or facial recognition, which have significant applications in automation, security, and surveillance.
- Medical imaging: To assist in diagnosis and treatment planning, image processing is widely employed in medical imaging, such as in X-ray, CT, and MRI scans.
- Robotics: Robots can sense and interact with their surroundings by using image processing, for as by identifying and tracking objects or navigating through challenging areas

The ability to extract usable information from images, improve visual quality, and enable machines to perceive and interact with their surroundings are all made possible by image processing, which is a potent tool that may be utilized in a variety of applications.

2.3 GRAY SCALE IMAGE:

An image that is purely grayscale has shades of gray, ranging from black to white, as its only colors. Each pixel in a digital grayscale image is represented by a single brightness value, with lower values denoting darker shades and higher values denoting lighter shades.

Since grayscale images may be saved and processed more effectively than color images while still transmitting crucial information about the intensity or brightness of the underlying data, they are frequently utilized in a variety of industries, including medical imaging, computer graphics, and image processing.

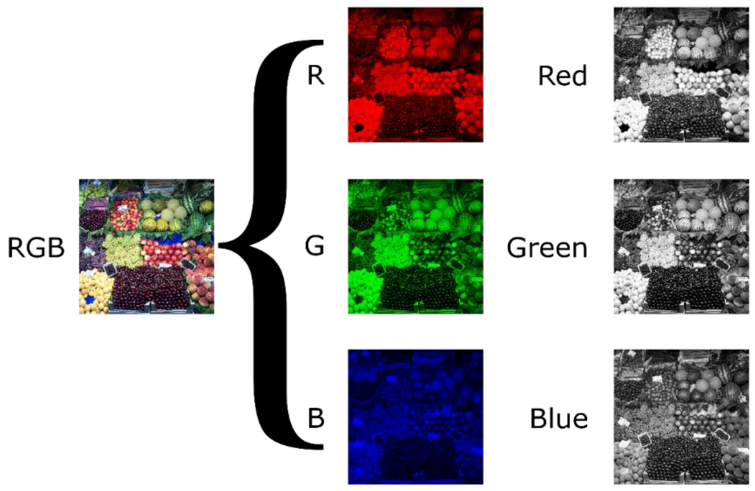

Fig 2.3 Composition of RGB from three gray scale images

2.4 COLOR IMAGE:

As opposed to a grayscale image, which just contains different shades of gray, a color image is one that contains colors. Each pixel in a digital color image has numerous values that represent the intensities of the image's primary hues, which are commonly red, green, and blue. (RGB).

Because they offer more visual information than grayscale images and are more suited to representing real-world landscapes and objects, color images are frequently utilized in a variety of industries, including photography, computer graphics, and medical imaging. Color images are valuable in a variety of image processing and computer vision applications because they may be utilized to extract additional information through color-based segmentation or feature extraction.

2.5 RELATED TECHNOLOGY:

2.5.1 R-CNN:

The object detection technique R-CNN (Region-based Convolutional Neural Network) is based on deep learning. Ross Girshick, et al. introduced it in a study published in 2014. R-CNN is a multi-stage approach that first creates candidate regions that could contain an object, or object proposals, and then uses a CNN to extract features from each suggested region. (Convolutional Neural Network). The object is then classified using these characteristics, and its placement within the suggested region is clarified.

Because it was able to attain more accuracy and better localization than earlier object detection techniques like sliding windows and selective search, R-CNN represented a substantial improvement. Then came quicker R-CNN, which increased R-CNN's efficiency and speed by sharing Using a region proposal network (RPN) and convolutional features, object suggestions are produced in a single forward pass. Since then, R-CNN has undergone significant advancements through several versions, including Mask R-CNN, which expands the algorithm to predict object masks, and Cascade R-CNN, which employs a cascade of classifiers to further enhance accuracy. R-CNN and its variations are still widely utilized in many fields, including surveillance, robotics, and self-driving automobiles, where they are effective methods for object detection.

2.5.2 SINGLE SHOT MULTI OBJECT DETECTOR:

A deep learning-based item recognition algorithm called Single Shot Multibox Detector (SSD) can recognize and categorize numerous objects in a single neural network pass. SSD is a one-stage detector, which means that it simultaneously creates object proposals and classifies them. Wei Liu, et al. first mentioned it in an article published in 2016. In order to produce item proposals at various scales and aspect ratios, SSD first extracts features from a picture using a convolutional neural network and then applies a set of default bounding boxes to the features. The final item detections are created by classifying and honing these proposals.

The term "multibox" refers to the handling of objects of varied sizes and forms using default boxes (or anchors) of various sizes and aspect ratios. This method makes SSD more effective than older two-stage detectors like R-CNN by enabling it to recognize objects at various scales and aspect ratios in a single pass.

Overall, SSD is a well-liked and successful object identification method that has been utilized in a variety of applications, including robotics, surveillance, and autonomous vehicles.

2.5.3 ALEXNET:

AlexNet is an image categorization system built using deep convolutional neural networks. It was created by Geoffrey Hinton, Alex Krizhevsky, and Ilya Sutskever and presented in a paper in 2012. On the ImageNet dataset, which has millions of annotated photos and is considered as a benchmark for image classification, AlexNet was one of the first deep learning models to reach state-of-the-art performance.

Eight layers, comprising five convolutional layers and three fully linked layers, total 60 million parameters in the AlexNet architecture. Additionally, it makes use of a number of significant methods, including dropout regularisation to stop neuronal co-adaptation and rectified linear units (ReLU) for activation functions.

A breakthrough in deep learning was made when AlexNet was able to considerably increase the accuracy of picture classification in comparison to earlier techniques. Due to its performance, later deep learning models like VGG, ResNet, and Inception were made possible.

2.5.4 YOLO:

Joseph Redmon, et al. created the deep learning-based object detection method YOLO (You Only Look Once) in 2016. YOLO is a one-stage object detector, which implies that in a single forward pass of the neural network, it directly predicts bounding boxes and class probabilities for objects in a picture. Based on a convolutional neural network, the YOLO method predicts multiple bounding boxes, each with a confidence score and class probabilities, for each cell in a grid formed by the input image. The class probabilities describe the likelihood that the object belongs to each of the specified classes, while the confidence score represents the likelihood that an object is present in the bounding box.

YOLO is renowned for its efficiency and speed because it can process photos in real time using common technology. However, especially for little items or objects that are close to one another, its accuracy may be lower than some alternative object identification methods, such as two-stage detectors like R-CNN. Since its debut, YOLO has undergone multiple iterations that have addressed some of its shortcomings and increased its accuracy. These iterations include YOLOv2, YOLOv3, and YOLOv4. Real-time object detection is crucial in many applications, including surveillance, robotics, and self-driving automobiles, where YOLO and its variants are commonly used.

2.5.5 VGG:

A deep convolutional neural network architecture called VGG (Visual Geometry Group) was created for image recognition and categorization. Karen Simonyan and Andrew Zisserman from the University of Oxford first discussed it in an article published in 2014. VGG is distinguished by its depth, with models ranging from VGG11 to VGG19, which contain 11 to 19 layers, respectively, and its use of relatively small (3x3) convolutional filters. Max pooling, fully linked layers, and the rectified linear unit (ReLU) activation function are further techniques used by VGG. Following VGG's state-of-the-art performance in the ImageNet Large Scale Visual Recognition Challenge in 2014, many additional computer vision projects have adopted its design as a benchmark.

VGG is noted for its vast number of parameters, which can result in lengthy training times and high memory needs, but it is also known for its computational complexity. VGG has made a significant contribution to the field of deep learning overall by demonstrating the efficacy of using deep architectures and very small convolutional filters for image recognition and classification.

2.5.6 MOBILENETS:

A family of effective deep convolutional neural network designs called Mobile Nets was created for embedded and mobile vision applications. It was first mentioned in a 2017 article by Google researchers Andrew G. Howard et al.

Mobile Nets are intended to perform a variety of computer vision tasks, including object detection and image classification, with high accuracy while being smaller and faster than conventional deep neural networks. Mobile Nets are based on a collection of lightweight operations that lower the network's computational burden and parameter count.

By dividing a regular convolution into a depth wise convolution and a pointwise convolution, Mobile Nets' depth wise separable convolutions are used. The pointwise convolution merges the output of the depth wise convolution into a new set of channels after the depth wise convolution filters each channel of the input separately. With this method, the network's calculation and parameter requirements are reduced. There are various variations of mobile nets, including mobile net, mobile net v2, and mobile net v3. Every iteration is more accurate and effective than the one before it. In many different applications, including object detection and recognition on mobile devices, robots, and self-driving automobiles, where processing speed and memory utilization are crucial factors, Mobile Nets have been extensively used.

2.5.7 TENSOR FLOW:

An open-source software library called TensorFlow is used for differentiable programming and dataflow across a variety of tasks. It is generally used to create and train deep neural network models for machine learning, which are utilized for applications like speech and picture recognition, natural language processing, and autonomous vehicles. The Google Brain team created TensorFlow, which was originally made public in 2015. It offers a complete set of tools and interfaces for developing and deploying machine learning models, including higher-level APIs for both novices and experts as well as lower-level APIs for modifying models and trying out novel approaches.

TensorFlow is renowned for its adaptability, efficiency, and capacity for distributed training across a number of GPUs or CPUs. It also comes with a host of helpful features, including support for model deployment across many platforms, including mobile devices and the web, data pre-processing and data augmentation, visualization tools, and more.

TensorFlow features a robust ecosystem of community-contributed tools and extensions in addition to its core library, such as TensorFlow Hub, TensorFlow Lite, and TensorFlow.js, which add extra functionality for certain use cases and platforms. With applications in a variety of industries and fields, TensorFlow has grown to be one of the most well-liked and commonly used machine learning frameworks in the world.

2.6 APPLICATIONS OF REAL TIME OBJECT TRACKING:

Real-time object tracking has a wide range of applications in various fields. Here are some examples:

Surveillance and Security: The use of real-time object tracking in video surveillance systems allows security staff to immediately take action by following suspicious people or items in real-time.

Automotive Industry: Advanced driver assistance systems (ADAS) can use object tracking to detect and track automobiles, pedestrians, and other things, which helps to reduce accidents and enhance safety.

Robotics: Robots can interact with their environment more successfully and carry out complex tasks by using real-time object tracking to find and track items.

Medical Imaging: Medical imaging apps that use object tracking can follow and assess the movement of organs and tissues in real-time, which helps with the diagnosis and treatment of a variety of medical problems.

Sports Analysis: In order to help coaches and analysts make better decisions and boost team performance, real-time object tracking can be used in sports analysis to follow the movement of players, balls, and other objects in real-time.

Virtual and Augmented Reality: In virtual and augmented reality applications, object tracking can be used to track the movement of real-world items and superimpose virtual elements on top of them to create more immersive experiences.

2.7 REAL TIME OBJECT TRACKING WORKFLOW AND FEATURE EXTRACTION:

The following workflow is often included in real-time object tracking:

Object Detection: The first stage is to use an object detection method to find objects in the video stream. By detecting items in every frame of the movie, this method generates a bounding box around each object.

Extraction of Features: The next stage is to extract features from the objects after they have been discovered. In order to do this, the pixels inside the bounding box must be examined and a number of features, including edge features, texture descriptors, and color histograms, must be computed.
Object tracking: The video's future frames' objects are tracked using the extracted features. Different tracking algorithms, including Kalman filters, particle filters, or correlation filters, can be used for this. These algorithms estimate the object's motion and location in the following frame using the features.
Re-detection: In some circumstances, the item may blur or leave the screen, making it challenging to track. The object detection technique may be employed once more in these circumstances to re-detect the object in next frame
Feedback and improvement: Based on system feedback, the tracking algorithm might be improved. For instance, the system may change the tracking algorithm parameters or re-detect the object if the object is not being tracked accurately.
Overall, effective feature extraction and the application of quick and precise tracking algorithms are the keys to real-time object tracking. These are necessary for the system to maintain accurate object tracking over time while keeping up with the video stream.

CHAPTER 3

DEEP LEARNING

3.1 INTRODUCTION:

Deep learning, a branch of machine learning, use neural networks to discover intricate patterns and connections in data. Deep learning aims to make it possible for machines to learn from massive amounts of data and base predictions or judgements on those predictions.

Artificial neural networks that mimic the functioning of human brains are frequently used to create deep learning models. These networks are made up of layers of interconnected nodes, often known as "neurons," which are used to process and change data.

Large amounts of labelled data are fed into the neural network as part of the deep learning learning process, and the weights and biases of the neurons are changed to reduce the error between the expected output and the actual output. Backpropagation is the procedure that enables the network to steadily enhance its performance over time.

State-of-the-art performance in a variety of tasks, including picture and audio recognition, natural language processing, and autonomous vehicles, has been attained using deep learning. Additionally, it has facilitated innovations in fields like robotics, computer vision, and drug development.

The ability of deep learning to automatically learn complex features and representations from data, its scalability to big datasets and complicated models, and its generalizability to new and unexplored data are some of its main benefits.

Deep neural networks can be difficult to train and optimise, and deep learning also needs a lot of data and computer power. Despite this, deep learning is a science that is always evolving and growing thanks to the constant development of new architectures, methods, and applications..

Convolutional Neural Networks (CNNs): A class of neural network that is frequently employed in the processing of images and videos. CNNs are frequently used for tasks like image classification, object identification, and image segmentation because they extract features from pictures using convolutional layers.

Recurrent neural networks (RNNs) are a subclass of neural networks that are employed in tasks involving sequence modelling, including speech recognition, language translation, and natural language processing. RNNs can represent sequences of different lengths because they employ recurrent connections to convey information from one time step to the next.

Generative Adversarial Networks (GANs): A class of neural network used for generative tasks like text production, music composition, and image and video synthesis. The generator and discriminator networks of GANs are trained jointly in a two-player game to generate realistic output.

Transfer learning is a method that uses previously learned models as a jumping off point for new tasks. Transfer learning is advantageous when there is a lack of training data for a new task since it enables the model to apply information obtained from earlier tasks.

Autoencoders: A class of neural network used for unsupervised learning tasks including dimensionality reduction and data compression. A decoder reconstructs the original input from the latent representation in an autoencoder, which consists of an encoder that compresses input data into a latent representation.

A kind of machine learning called reinforcement learning is applied to decision-making tasks in dynamic situations. Applications like gaming, robotics, and autonomous driving use reinforcement learning algorithms, which gradually learn how to act in the environment to maximise a reward signal.

A typical type of neural network utilized in image processing applications like image identification, object detection, and picture segmentation is the convolutional neural network. Convolutional layers are the foundation of CNNs, which use them to extract significant characteristics from input images. Each layer in a CNN is made up of a collection of teachable filters, often known as kernels or convolutional kernels. Each filter glides over the input image to execute a convolution operation and is small (3x3 or 5x5 pixels, for example). A feature map, which is the result of the convolution operation, highlights specific patterns or aspects of the input image that are crucial for the task at hand.

In order to down sample the feature maps and condense their size, CNNs also employ other kinds of layers, such as pooling layers. Moreover, pooling layers can aid in strengthening the network's resistance to minute changes in the input image.

Eventually, the final classification or regression operation is frequently carried out by fully connected layers, which are comparable to the layers in a conventional neural network.

The ability of CNNs to automatically learn features from unprocessed input data eliminates the need for manual feature engineering, which is one of their key advantages. For applications like object recognition, where the patterns that characterize, an object can be exceedingly complicated and challenging to specify explicitly, this makes CNNs very successful.

3.2 Feedforward and feedback networks:

A feedforward network is a network with hidden layers, inputs, and outputs. Only one way can be travelled by the signals (forward). A layer that performs calculations receives input data. Based on the weighted sum of its inputs, each processing element performs computations. The updated values become updated input values for the subsequent layer (feed-forward). This keeps going over each layer, resulting in the output. For instance, feedforward networks are frequently employed in data mining.

A feedback network has feedback channels, such as a recurrent neural network. This implies that they can use loops to have signals moving in both directions. Neuronal connections can be made in any way. Since this kind of network contains loops, it transforms into a non-linear dynamic system that evolves continually until it achieves an equilibrium state. For solving optimization problems, feedback networks are frequently utilized to find the optimal configuration of related components.

3.3 Weighted Sum:

Features from a training set or the outputs of neurons in a lower layer can both be inputs to a neuron. Each synapse at each point of connection between two neurons has a different weight. You must go along the synapse and pay the "toll" if you want to move from one neuron to the next (weight). The neuron then adds up the weighted inputs from all the incoming synapses and applies an activation function.

All of the neurons in the following layer receive the result from it. We refer to changing the weights on these synapses when we discuss updating weights in a network. The weighted outputs of all the neurons in the layer before are added to form a neuron's input. The weight of the synapse that connects each input to the present neuron is multiplied by each input. Each neuron in the current layer will have 3 different weights, one for each synapse, if the preceding layer had 3 inputs or neurons.

In a nutshell, a node's output is determined by its activation function.

The transfer function or activation function converts input signals into output signals. The output values are mapped onto a range, such as 0 to 1 or -1 to 1. The pace at which the cell's action potentials fire is represented by an abstraction. It is a numerical indicator of how likely it is that the cell may ignite. The function is binary at its most basic level: either yes (the neuron fires) or no (the neuron doesn't fire). The output can be anywhere in a range or can be either 0 or 1 (on/off or yes/no). An output of 0.9, for instance, would indicate a 90% chance that your image is, in fact, a cat if you were using a function that maps a range between 0 and 1 to assess the likelihood that an image is a cat.

3.4 Activation function:

The transfer function or activation function converts input signals into output signals. The output values are mapped onto a range, such as 0 to 1 or -1 to 1. The pace at which the cell's action potentials fire is represented by an abstraction. It is a number that indicates how likely it is that the cell will set off. The function is binary at its most basic level: either yes (the neuron fires) or no (the neuron doesn't fire). The output can be anywhere in a range or can be either 0 or 1 (on/off or yes/no).
What are our options? There are other activation mechanisms, but these four are the most prevalent ones**.**

3.4.1 Threshold function

This function takes steps. The function returns 0 if the total of the input values falls below a predetermined threshold. If it is more than zero or equal to zero, it will pass on 1. It is a very strict, simple, yes-or-no function.

3.4.2 Sigmoid function

Logistic regression employs this function. It progresses from 0 to 1 smoothly and gradually, in contrast to the threshold function. On the output layer, it is helpful and frequently utilised for linear regression.

3.4.3 Hyperbolic Tangent Function

The sigmoid function and this function are extremely similar. Nevertheless, in contrast to the sigmoid function, which ranges from 0 to 1, the value ranges from -1 to 1. Although this does not closely resemble what occurs in the brain, this function produces better results when neural networks are being trained. Sometimes, while being trained with the sigmoid function, neural networks become "stuck." This occurs when there is an abundance of extremely negative input that keeps the output close to zero and interferes with learning.

3.4.4 Rectifier function

In the world of neural networks, this activation function may be the most common. It is the most effective and makes biological sense. It has a kink, but following the kink at 0, it is smooth and progressive. For instance, your output would be "no" or a percentage of "yes" if this were the case. There is no need for normalization or other intricate calculations with this function.
When machines can carry out tasks that would normally require intellect from a human, artificial intelligence becomes crucial. It falls under the category of machine learning, where computers may

pick up knowledge and gain insight through experience without the aid of humans. Artificial neural networks, algorithms modelled after the human brain, learn from a lot of data in deep learning, a subset of machine learning. The idea behind deep learning is based on human experience; a deep learning algorithm would repeatedly do a task in order to improve the result. Learning is made possible by the numerous (deep) layers of neural networks. Deep learning can be trained to solve any problem that requires "thinking" to solve.

CHAPTER 4

YOU ONLY LOOK ONCE (YOLO)

4.1 INTRODUCUTION:

YOLO (You Only Look Once) is a real-time object detection system developed by Joseph Redmon and his colleagues at the University of Washington. The YOLO algorithm is made to identify objects quickly and precisely in still and moving picture frames.

YOLO breaks an image into a grid of cells and forecasts bounding boxes and object class probabilities for each cell, in contrast to typical object detection algorithms that use region-based approaches like R-CNN. Using only one forward trip across the network, compared to previous techniques, this method allows YOLO to detect objects.

The YOLO algorithm is built using a deep convolutional neural network (CNN) architecture, which employs many layers to extract characteristics from the input image. Bounding boxes and item class probabilities are predicted for each cell in the picture grid by the network's final layer.

YOLO's quickness is one of its main benefits. Real-time image processing capabilities of the algorithm make it ideal for use in robotics, surveillance, and autonomous vehicles. Also, YOLO has a reputation for accuracy, consistently producing cutting-edge outcomes on a variety of benchmark datasets. The YOLO technique is simple for developers to utilize and modify for their applications because it has been included into several well-known deep learning frameworks, such as TensorFlow, PyTorch, and Darknet. The algorithm's most recent iteration, YOLOv5, has increased speed and accuracy even further, making it a popular option for real-time object recognition jobs.

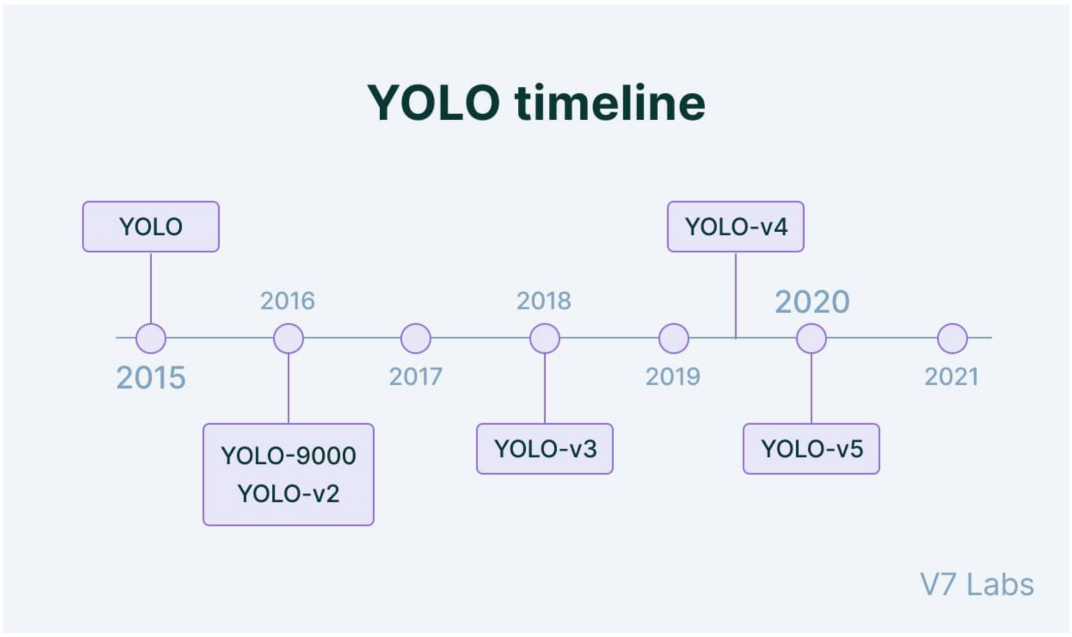

Fig 4.1 YOLO timeline

4.2 YOLO VERSIONS:

The YOLO method has been created in a number of iterations over the years, each with enhancements to accuracy, speed, and other elements. Here is a quick rundown of some of the most famous YOLO variations:

1. YOLOv1: The original version of YOLO was introduced in 2015 and used a single neural network to predict object classes and bounding boxes. While it was fast, it struggled with small objects and had lower accuracy compared to later versions.
2. YOLOv2: Released in 2017, YOLOv2 addressed some of the limitations of the original algorithm by incorporating features like batch normalization, anchor boxes, and multiscale prediction. These changes improved accuracy and made the algorithm more robust to different object sizes.
3. YOLOv3: YOLOv3 was introduced in 2018 and further improved accuracy by using a larger backbone network, a feature pyramid network, and a better loss function. It also introduced new features like dynamic anchor boxes, which adapt to the shape of the object being detected, and multi-label classification.
4. YOLOv4: Released in 2020, YOLOv4 was a major update to the algorithm, introducing several new features like CSPNet, Spatial Pyramid Pooling (SPP), and Mish activation function. These changes improved accuracy, speed, and memory efficiency, making YOLOv4 one of the best-performing object detection algorithms.

5. YOLOv5: YOLOv5 was introduced in 2020 as a lightweight version of the algorithm, designed for faster and more efficient object detection. It uses a simpler architecture and introduces techniques like anchor-free object detection and focal loss. YOLOv5 achieved state-of-the-art accuracy on several benchmark datasets while being significantly faster than previous versions.

4.2.1 YOLO version1:

YOLOv1 is the first version of the YOLO object detection algorithm, introduced in 2015 by Joseph Redmon and his team at the University of Washington. YOLOv1 was a groundbreaking algorithm that introduced a new approach to object detection, allowing for real-time detection and classification of objects in images and videos.

Using a single neural network, the YOLOv1 model predicts item classes and bounding boxes in an image. Each cell in the input image's grid-like split is responsible for recognizing objects that are included within it. The system then forecasts bounding boxes and object class probabilities for each cell in the grid, allowing for the identification of several things during a single run through the network.

The quickness of YOLOv1 is one of its main benefits. The algorithm's ability to interpret photos in real-time makes it ideal for use in robots and surveillance systems. In contrast to subsequent versions, YOLOv1 exhibited poorer accuracy and had trouble detecting tiny objects.

24 convolutional layers and 2 fully linked layers made up the deep convolutional neural network (CNN) architecture employed by YOLOv1. The PASCAL VOC dataset, which includes pictures from 20 different item categories, was used to train the network. On this dataset, the algorithm produced state-of-the-art results and had an average accuracy of 63.4% while detecting objects in real time.

Although YOLOv1 has since been outperformed by more recent iterations of the algorithm, it represented a significant advance in the field of object recognition and opened the door for future quicker and more precise systems.

4.2.2 YOLO version2:

The second iteration of the YOLO object identification system, dubbed YOLOv2 (You Only Look Once version 2), was unveiled in 2017 by Joseph Redmon and his colleagues at the University of Washington. The second version of the algorithm, known as YOLOv2, fixed some of the issues of the first version.

The inclusion of anchor boxes, which allowed the algorithm to more correctly recognise objects of various sizes and aspect ratios, was one of the major advancements in YOLOv2. The method employs anchor boxes, which are pre-defined boxes of various sizes and shapes, to forecast the positions and sizes of objects. This strategy enables the program to more precisely locate items of various sizes.

Additionally, batch normalization was included in YOLOv2, which increased the network's performance and stability. The system also employed a multi-scale strategy to identify items of various sizes, which improved its ability to handle tiny things.

The deep convolutional neural network (CNN) architecture employed by the YOLOv2 algorithm included 53 convolutional layers and 4 max-pooling layers. On the COCO dataset, which includes pictures from 80 different item categories, the network was trained. On this dataset, the algorithm produced cutting-edge findings and had a real-time item detection accuracy of 78.6% on average.

Overall, YOLOv2 was a significant advancement over YOLOv1, achieving more accuracy while still delivering real-time performance. The technique was a popular choice for real-time object identification jobs since it considerably increased accuracy and stability through the use of anchor boxes and batch normalization.

4.2.3 YOLO v3:

The third iteration of the YOLO object identification algorithm, dubbed YOLOv3 was released in 2018 by Joseph Redmon and his colleagues at the University of Washington. The accuracy and speed of the algorithm are further improved by YOLOv3, which draws on the advancements made by YOLOv2.

The inclusion of a feature pyramid network (FPN), which enables the algorithm to recognise objects at various sizes, was one of the main advancements in YOLOv3. In order to identify objects of various sizes, the FPN is utilised to extract information from the neural network's various layers.

YOLOv3 also introduced a new detection head that predicts object categories and bounding boxes at three different scales, allowing the algorithm to detect objects of different sizes more accurately. In addition, the algorithm used a technique called "swish" activation, which improved the accuracy of the network.

A deep convolutional neural network (CNN) architecture of 106 convolutional layers was utilised by the YOLOv3 algorithm. The COCO dataset, which includes photos from 80 different item categories, was used to train the network. On this dataset, the algorithm produced state-of-the-art results and had an average accuracy of 81.4% while detecting objects in real time.

Overall, YOLOv3 is a big step forward over YOLOv2, and it increased accuracy while still preserving real-time performance. The algorithm's accuracy and stability were greatly enhanced with the addition of the new detection head and feature pyramid network, making it a popular option for real-time object identification jobs.

4.2.4 YOLO v4:

The YOLO object identification algorithm's fourth and most recent iteration, dubbed YOLOv4, was unveiled in 2020 by Alexey Bochkovskiy and his colleagues at the AI research firm Ultralytics. YOLOv4 outperformed YOLOv3 in benchmarks for object identification, achieving cutting-edge performance.

The implementation of a more potent backbone network, known as CSPDarknet53, which increased the accuracy and speed of the algorithm, was one of the main advancements in YOLOv4. Additionally, YOLOv4 added a number of new methods, including as cut mix regularization, self-adversarial training, and mosaic data augmentation, which increased the algorithm's accuracy and resilience.

In order to increase the accuracy of object recognition, YOLOv4 also created a new architecture known as the "YOLOv4 neck" that mixes features from many network levels. To extract characteristics from various sizes, the architecture combines spatial pyramid pooling (SPP) and route aggregation network (PAN) components.

On the COCO dataset, the YOLOv4 algorithm produced cutting-edge results, with an average accuracy of 43.5% on the test-dev dataset. On a Tesla V100 GPU, the algorithm was also able to recognise objects in real-time at an average frame rate of 65 FPS.

Overall, YOLOv4 outperformed YOLOv3 and produced cutting-edge outcomes in object detection benchmarks. The algorithm's accuracy and speed were greatly enhanced by the adoption of a more potent backbone network, fresh data augmentation methods, and the YOLOv4 neck design, making it a popular option for real-time object recognition jobs.

4.2.5 YOLO v5:

A deep learning-based object identification method called YOLOv5 was created by Ultralytics and made public in 2020. YOLOv5 is a novel approach to object identification that makes use of a distinct architecture and training procedure rather than an official upgrade to the YOLOv4 algorithm.

YOLOv5 has a distinct architecture known as a "backbone" network that is based on a well-liked object detection model called EfficientDet, in contrast to earlier versions of YOLO. The detection head uses the information that the backbone network extracts from the input picture to make predictions about the position and kind of objects in the image.

YOLOv5 is trained using a new training method called "Scaled-YOLOv4", which involves scaling the input images during training to improve the accuracy of the model. This training method allows YOLOv5 to achieve state-of-the-art accuracy on object detection benchmarks while maintaining real-time performance.

One of YOLOv5's important characteristics is its capacity for "anchor-free" recognition, which enables it to recognise objects with various sizes and aspect ratios. Anchor boxes, which were required in earlier iterations of YOLO to identify objects of various sizes and shapes, are no longer necessary thanks to this method.

On the COCO dataset, the YOLOv5 algorithm produced cutting-edge results, with an average accuracy of 50.0% on the test-dev dataset. On a Tesla V100 GPU, the algorithm was also able to recognise objects in real-time at an average frame rate of 200 FPS.

In comparison to earlier iterations of YOLO, version 5 represents a novel approach to object identification that makes use of a distinct architecture and training methodology.

The inference hardware, the size of the input image, and the difficulty of the detection job are some of the variables that affect the speed and accuracy of YOLOv5. In general, YOLOv5 is a very accurate object identification method that is quite quick.

Depending on the size of the input image and the difficulty of the detection job, the Ultralytics team claims that YOLOv5 can accomplish real-time object recognition at up to 140 frames per second (FPS) on a Tesla V100 GPU. YOLOv5 is now one of the quickest object identification algorithms on the market as a result.

On the COCO dataset, YOLOv5 produced findings that were state-of-the-art in terms of accuracy, with an average precision of 50.0% on the test-dev dataset. This is a substantial improvement above the accuracy attained by earlier iterations of YOLO.

It's important to note that YOLOv5's accuracy may be increased any further by employing transfer learning or fine-tuning the algorithm on certain datasets. Additionally, by tuning the algorithm's hyperparameters and employing more potent hardware for inference, YOLOv5's speed and accuracy may be increased.

4.3 YOLO V4 ARCHITECTURE:

YOLO v4 is an object detection algorithm that is known for its high speed and accuracy. Here are the key features and architecture of YOLO v4:

1. Backbone network: YOLO v4 uses a CSPDarknet53 architecture as the backbone network, which is a latest version of the Darknet53 architecture used in YOLO v3. The CSPDarknet53

architecture includes cross-stage partial connections between layers, which helps to improve the flow of information through the network.

2. SPP and SAM modules: The neck network of YOLO v4 includes SPP (Spatial Pyramid Pooling) and SAM (Spatial Attention Module) modules, which help to improve the receptive field and localization of small objects. The SPP module performs spatial pyramid pooling on the feature maps to capture objects at different scales, while the SAM module applies attention mechanisms to focus on relevant regions of the feature maps.
3. Prediction head: Prediction head of YOLO v4 is a series of CNN layers followed by a global average pooling layer and several fully connected layers. It predicts the bounding boxes, object scores, and class probabilities for objects in the image.
4. Training pipeline: YOLO v4 uses a custom training pipeline that includes several advanced techniques, such as random training shapes, mosaic data augmentation, and self-adversarial training. These techniques help the model become more accurate and robust.
5. Model optimization: YOLO v4 includes several model optimization techniques, such as dynamic convolution, Mish activation, and Drop Block regularization. These techniques help to make the model more accurate while shrinking its size.
6. Multi-scale training and testing: YOLO v4 supports multi-scale training and testing, which enables the model to detect objects at different scales and resolutions.

Overall, YOLO v4 is a highly optimized and efficient object detection algorithm that achieves state-of-the-art performance on several benchmarks. Its advanced architecture and training pipeline make it a popular choice for a wide range of applications, including autonomous driving, robotics, and surveillance.

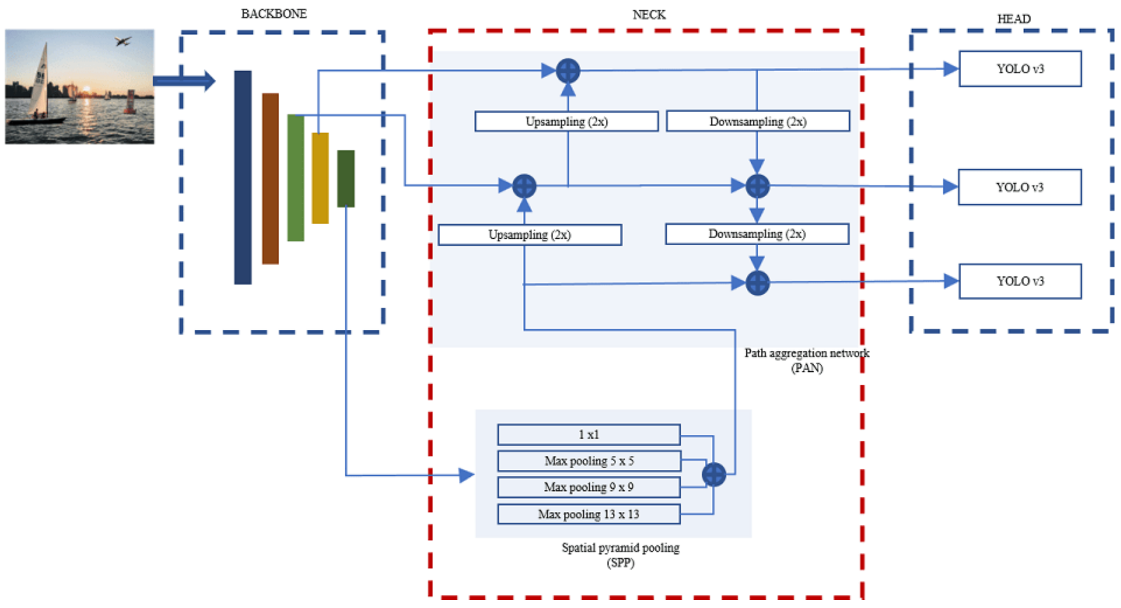

Fig 4.3 YOLO V4 architecture

4.3.1 Predict Objects Using YOLO v4:

- In order to identify groups of items in a picture, YOLO v4 employs anchor boxes. See anchor box for object detection for more information on anchor boxes. YOLO v4 forecasts these three characteristics for each anchor box, just like v3 did:

  • Predicts the object of each anchor box using intersection over union (IoU).

  In tasks involving object recognition and picture segmentation, intersection over union is a common assessment metric. For a specific item in an image, the overlap between the predicted bounding box (also known as the segmentation mask) and the ground truth bounding box (also known as the segmentation mask) is measured using the IoU method.

  The area of intersection between the predicted bounding box and the ground truth bounding box is compared to the area of union between the two bounding boxes to produce the IoU metric.

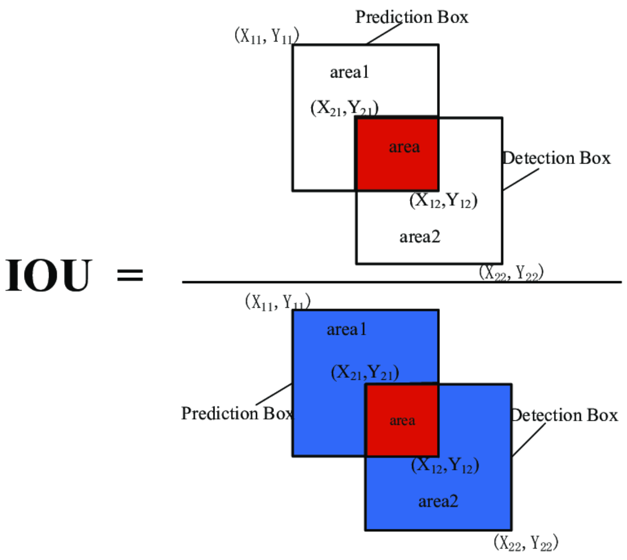

Fig 4.3.1 IoU

The IoU value ranges from 0 to 1, with 0 signifying a complete match between the expected and ground truth bounding boxes and 1 signifying a complete lack of overlap. A greater IoU value denotes better segmentation or detection precision.

The performance of object identification and segmentation algorithms is frequently assessed using IoU as a measure during training and testing. Models with greater IoU values are chosen over those with lower IoU values because they are thought to be more accurate. IoU is also employed as a threshold for removing erroneous segmentations or detections. For instance, for a model to be regarded a legitimate detection or segmentation, it may need to have an IoU higher than a predetermined threshold (for instance, 0.5).

- Anchor box offsets — Refines the anchor box position.

In object identification algorithms like YOLO (You Only Look Once) and Faster R-CNN (Region-based Convolutional Neural Network), anchor boxes play a crucial role. A predetermined box form and size called an anchor box is used to indicate potential placements for items in a picture.

The objective of object detection is to identify and categorize items in a picture, and in order to achieve this precisely, the algorithm must forecast where the objects will be in the image. The algorithm splits the picture into a grid of cells in order to do this, and then it assigns a set of anchor boxes of various sizes and aspect ratios to each cell in the grid. For example, an anchor box might be rectangular with dimensions of 32x32 pixels, 64x64 pixels, or 128x128 pixels, and with aspect ratios of 1:1, 1:2, or 2:1.

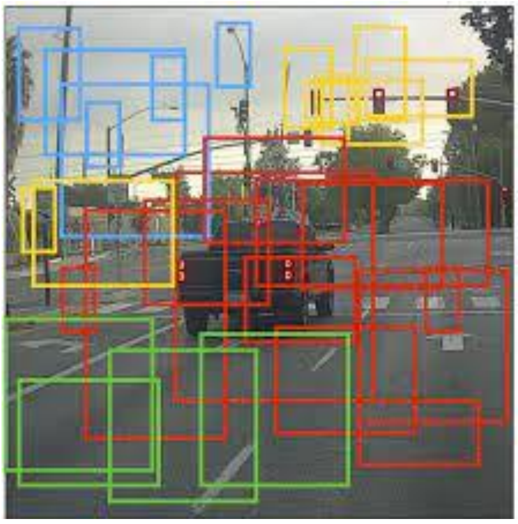

Fig 4.3.2 Anchor boxes

In order to fit the size and form of the objects in the image, the algorithm modifies the parameters of the anchor boxes during training. The likelihood of an object appearing in each cell and the coordinates of the bounding box that best matches the object are then predicted by the method.
Anchor box usage increases the accuracy of the detection results and helps the algorithm handle objects of various sizes and forms. Additionally, anchor boxes lessen the amount of potential bounding boxes that the algorithm must consider, which aids in hastening the detection process.

- 27Class probability — Predicts the class label assigned to each anchor box.

Class probability in object detection is the likelihood or confidence level that an object in a picture belongs to a specific class. Cars, people, bicycles, and traffic lights are just a few examples of the types of things that are commonly categorized by object identification algorithms into a preset set of classes.
The system gains the ability to give each class for each observed item a likelihood score during training. For instance, if the algorithm identifies a vehicle in an image, it may give the "car" class a probability score of 0.8 while giving lower probabilities to other classes.
During testing or inference, the algorithm uses the class probability scores to determine the most likely class for each detected object. The algorithm may use a threshold value to filter out detections with low probability scores, as these are more likely to be false positives.

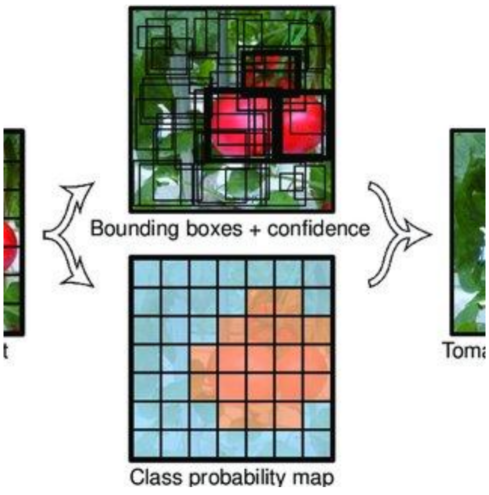

Fig 4.3.3 Class probability mapping

Class probability scores are an important output of object detection algorithms because they provide information about the confidence level of the algorithm's predictions. Higher probability scores indicate greater confidence in the detection result, while lower probability scores indicate a higher likelihood of error. Object detection systems often use class probability scores to evaluate and compare the performance of different algorithms or models.

4.4 METHODOLOGY:

Object detection and tracking using YOLOv4 involves using the YOLOv4 model to detect objects within an image or video, and then using an algorithm to track those objects over time. Here are the basic steps involved. In Object detection, the YOLOv4 model is used to detect objects within each frame of the input video or image sequence. This involves passing each frame through the model and using the output to draw bounding boxes around each detected object. In Object tracking: Once the objects have been detected in the first frame, an object tracking algorithm can be used to track them over time. One popular algorithm for object tracking is the Kalman filter, which uses a set of equations to estimate the position and velocity of each object over time. Data association: As new frames are processed, the objects may move, change shape, or become partially occluded.
To maintain accurate tracking, the algorithm needs to associate each new detection with the correct object from the previous frame. This can be done using various techniques, such as matching the location and size of the bounding boxes or using appearance-based features.

Input: To use YOLOv4 for object detection, you would typically need an image or video dataset as input. The input dataset should consist of images or videos that you want to analyze for objects of interest. The images or videos should be in a compatible format, such as JPEG or MP4.

Feature extraction: Feature extraction in YOLOv4 involves extracting features from the intermediate layers of the neural network, rather than using the network for object detection.

Learning feature vectors: YOLOv4 can be used for learning feature vectors through a process called transfer learning. Transfer learning is a technique where a pre-trained neural network, such as YOLOv4, is used as a starting point for a new task, and then fine-tuned on a new dataset.

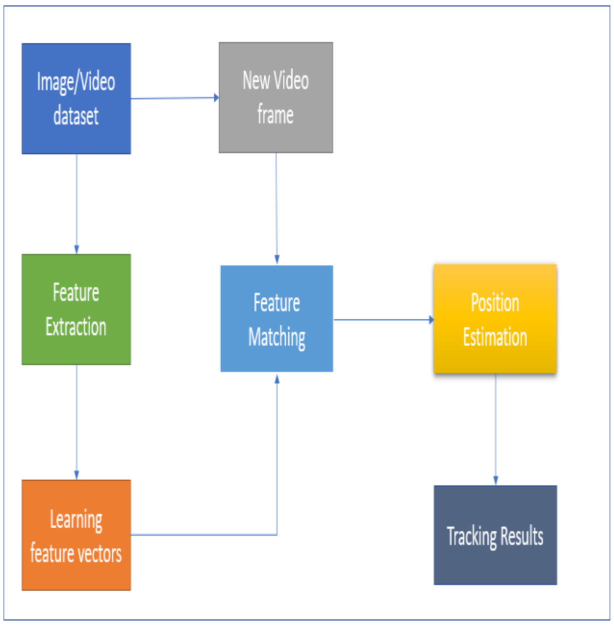

Fig 4.5 Block diagram for object tracking

Feature matching: Once you have extracted feature vectors from your input images using YOLOv4, you can then compare these feature vectors to a reference set of feature vectors to find matches.

Position estimation: The x and y coordinates of the box's top-left corner, combined with its width and height, are commonly used to describe the bounding box's coordinates. The centre of the bounding box and its separation from other objects or interesting areas of the picture or video may both be determined using these parameters.

Tracking results: The tracking algorithm typically uses the bounding boxes of the detected objects to predict their position in the next frame, and then compares these predictions to the actual positions of the objects in the next frame to update their position estimates. The algorithm can also use additional information such as object velocities or previous motion trajectories to improve the accuracy of the position estimates.

4.5 CONVOLUTION NEURAL NETWORK:

In applications for image identification and classification, the term "CNN" refers to a class of neural networks. CNNs are designed to process and analyze pictures in a way that is similar to how individuals perceive visual information, and they are based on the structure and function of the visual cortex in animals.

The fundamental components of a CNN are convolutional layers, pooling layers, and fully connected layers. A convolutional layer employs a series of learnable filters (also known as kernels or weights) to search the input picture for certain traits or patterns. The result of the convolutional layer is a collection of feature maps that emphasize the presence of specific qualities in the input picture. The pooling layer, which down samples the feature maps and lowers the dimensionality of the data, increases the network's computational efficiency. A prediction is made using the learned features by the fully connected layers, which are typical neural network layers that take the output of the convolutional and pooling layers. They are used for tasks involving classification or regression.

CNNs provide several advantages over other machine learning techniques for categorising images. They may immediately learn features from the picture data without the need for human feature engineering. Additionally, they can capture spatial connections between features, which is essential for jobs involving picture identification. Finally, because they can support different picture sizes and orientations, they are more resistant to changes in the input data.\

CNNs have been used in several applications, including object detection, facial recognition, and natural language processing. Since they have gained state-of-the-art performance in several benchmarks for image identification, they continue to be a popular focus of research in computer vision and machine learning.

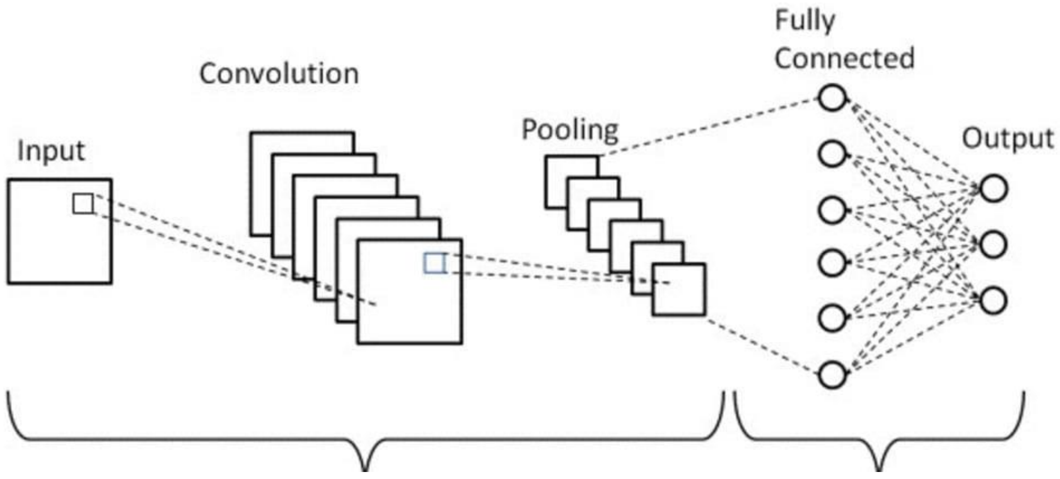

Fig 4.5 CNN

4.5.1 How do CNN work?

Artificial neural networks often employed in computer vision and image recognition applications are convolutional neural networks, also referred to as CNNs. The core of a CNN is composed of convolutional layers, pooling layers, and fully linked layers. Here is how they work:

1. Convolutional layers: A convolutional layer convolves a series of filters over an input picture to create a feature map. The dot product between the filter weights and the picture's pixel values is calculated for each point by each filter, which is a collection of weights that glides across the image. The resultant feature map has only one value.
2. Pooling layers: To minimize the spatial dimensions of the feature maps, a pooling layer is frequently added after each convolutional layer. The most used pooling method is max pooling, which outputs the highest value from a small area of the feature map to the following layer.
3. Fully connected layers: The output is flattened into a one-dimensional vector and fed into a fully connected layer after multiple convolutional and pooling layers. Each neuron in this layer is linked to every neuron in the layer above, similar to a standard neural network layer. The prediction about the input picture is then based on the output of the fully connected layer.

CNNs use backpropagation to optimize the weights in the filters and fully connected layers during training. By adjusting these weights, the network learns to recognize patterns and features in images, allowing it to make accurate predictions on new, unseen images.

4.6 CNN LAYERS:

4.6.1 Convolutional layer:

In this layer, all major computation takes place. This layer requires three things: input data, a filter, and a feature map. A kernel or feature detector are common names for the filter. This filter checks the image's receptive fields to see if a specific feature is present or not. Convolution is the name given to this technique.

Let's suppose for the time being that the input is a 3D pixel matrix representing a colour image. This implies that the height, width, and breadth will all have RGB values. A 2D array of weights representing a portion of the image makes up the feature detector. The size of the receptive field is determined by the filter, which is commonly a 3x3 matrix.

After the filter has been applied to a portion of the image, the dot product between the input pixels and the filter is calculated. The output array is then given this dot product. The filter travels in small, repeated stages until it has completely engulfed the input picture. Feature maps, activation maps, or convolved features are the outcome of a series of dot products made from the input and the filter.

A convolutional layer applies a series of filters to an input picture in order to create a feature map.

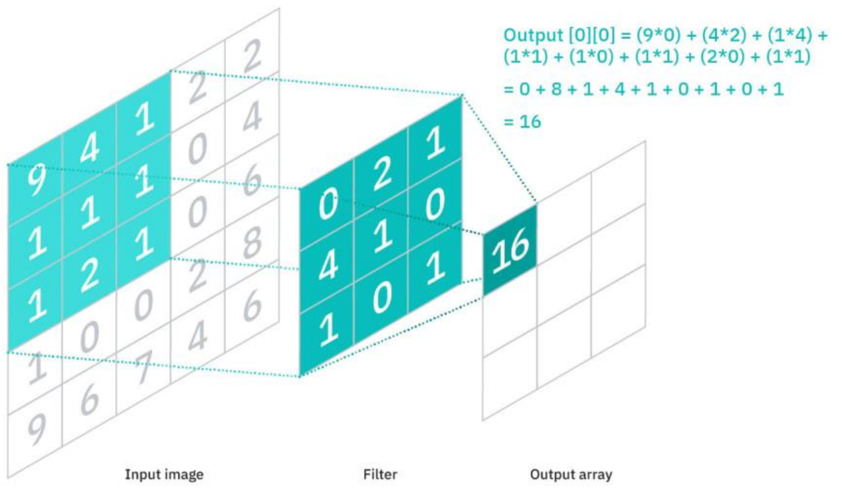
Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
16
Input image
Filter
Output array

Fig 4.6.1 Convolution layers

In their role as feature extractors, the convolutional layers learn the feature representations of the input pictures. Convolutional layers' neurons are arranged into feature maps. A collection of trainable weights, often referred to as a filter bank, connects each neuron in a feature map's receptive field to a group of nearby neurons in the layer above. A fresh feature map is produced by convolving inputs with learnt weights, and the outputs are then sent through a nonlinear activation function.

All neurons inside a feature map have weights that are confined to being equal, even though different feature maps within the same convolutional layer have varied weights so that several features can be retrieved at each location.

As its name implies, the convolutional layer is essential to how CNNs operate. The primary focus of the layers parameters is the use of learnable kernels.

These kernels entirely fill the depth of the input, although typically having a limited spatial dimension. Each filter is convolved across the spatial dimensions of the input as soon as the data reaches the convolutional layer, producing a 2D activation map. These activation maps are visibly discernible.

The scalar product is calculated for each value in that kernel as we travel through the input. The network kernels will consequently develop the ability to "fire" when they identify a certain feature in a specific spatial place in the input. These are frequently referred to as activations.

The core element of the kernel is covered by the input vector, which is then calculated and replaced with a weighted sum of any surrounding pixels and of the input vector itself.

The whole output volume of the convolutional layer will be created by stacking the activation maps that are associated with each kernel along the depth dimension.

As we alluded to before, training ANNs with inputs like images results in models that are too big to learn effectively. This is due to the entirely coupled structure of traditional ANN neurons, hence each neuron in a convolutional layer only connects to a small piece of the input volume to prevent this. The dimensionality of the region is referred to as the size of the neuron's receptive field. The magnitude of the connectivity through the depth is almost always equal to the depth of the input.

Each neuron in the convolutional layer would have a total of 108 weights when the receptive field size is set to 6 6 and the input to the network is an RGB-colored image with 64 64 dimensions, for example. (6 6 3), where 3 denotes the degree of connectivity inside the depth of the volume. Consider that a typical neuron seen in other forms of ANN might have 12, 288 weights to put this into context.

One can regulate the depth of the output volume that the layers produce by manually altering the number of neurons in each convolutional layer to the same region of the input. Many ANN models show this to be the case, with every neuron in the hidden layer being directly connected to every other neuron in the preceding layer. While reducing this hyperparameter can greatly reduce the network's total number of neurons, it can also significantly reduce the model's ability to recognise patterns.

We can also provide the stride in which we establish the depth around the spatial dimensions of the input in order to place the receptive field. For example, if we set the stride to 1, the resulting receptive field would be densely overlapping and would result in extremely large activations. The quantity of overlapping will be reduced and the outcome will have lower spatial dimensions as a result of lengthening the stride.

The simple process of zero-padding entails padding the input's boundary. More control over the dimensionality of the output volumes is effectively provided.

It's important to understand that by using these techniques, we will alter the spatial dimensionality of the convolutional layers' output.

If we use an image input with any real dimensionality, despite all of our existing efforts, we will still find that our models are enormous. Nonetheless, methods have been developed to dramatically cut down on the convolutional layer's overall parameter count.

If an area feature can be calculated at one particular spatial location, it is probably helpful in another, according to the principle of parameter sharing. If each individual activation map is restricted to have the same weights and bias, the total number of parameters in the output volume will be significantly decreased.

As a result, as the backpropagation stage advances, each neuron in the output will represent the overall gradient of which may be added throughout the depth, only updating one set of weights as opposed to all of them.

4.6.2 Pooling layer:

This layer, sometimes referred to as down sampling, performs dimensionality reduction, lowering the number of input parameters. It is necessary to reduce the amount of computing power needed to process the data. Also, it supports the process of efficiently training the model by extracting prominent features.

Similar to the convolutional layer, it applies the filter to all of the input data, but this filter doesn't have weights. Instead, the kernel fills the output array with values from the receptive field using aggregation functions.

Pooling comes in two varieties:

a. Max Pooling: This filter returns the highest value from the area of the image that is covered by the kernel.

b. Average Pooling: The filter outputs the mean of all values from the kernel-covered area of the picture.

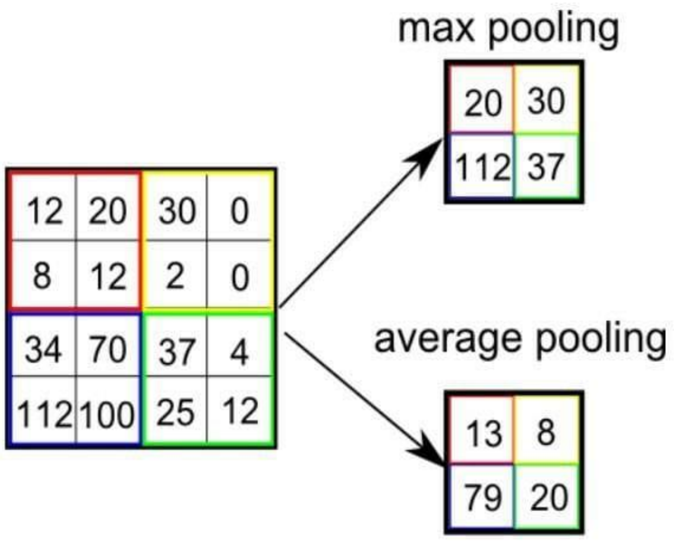

Fig 4.6.2 Pooling layer

The features present in an input image are summarized by the convolution layer of a convolutional neural network.

The disadvantage of the method is that the output feature maps are sensitive to the placement of the features in the input. One way to address this sensitivity is to downscale the feature maps. The result is that the down sampled feature maps, also known as "local translation invariance" in technical terms, are more resistant to changes in the position of the feature in the image.

Pooling layers provide a mechanism for down sampling feature maps by summarizing the existence of features in specific feature map patches. Two common pooling methods that, respectively, summarize a feature's average presence and its most active presence are average pooling and maximum pooling.

In this course, you will study how the pooling operation works and how to apply it to convolutional neural networks.

After finishing this tutorial, you will be aware of the following: • Down sampling feature detection in feature maps requires pooling.

• How to calculate and put average and maximum pooling in a convolutional neural network into practice.

• Convolutional neural network global pooling techniques.

4.6.3 Fully connected layers:

The output layer and the pixel values of the input image are not directly coupled in partially connected layers. Yet, each node in the output layer of this layer is straight-lined to a node in the layer beneath. This layer performs the classification process using features that were gathered from earlier layers and various filters.

ReLu functions are typically used in convolutional layers and pooling layers, whereas SoftMax activation functions are typically used in this layer to accurately classify inputs. The chance that this function returns can be between 0 and 1.

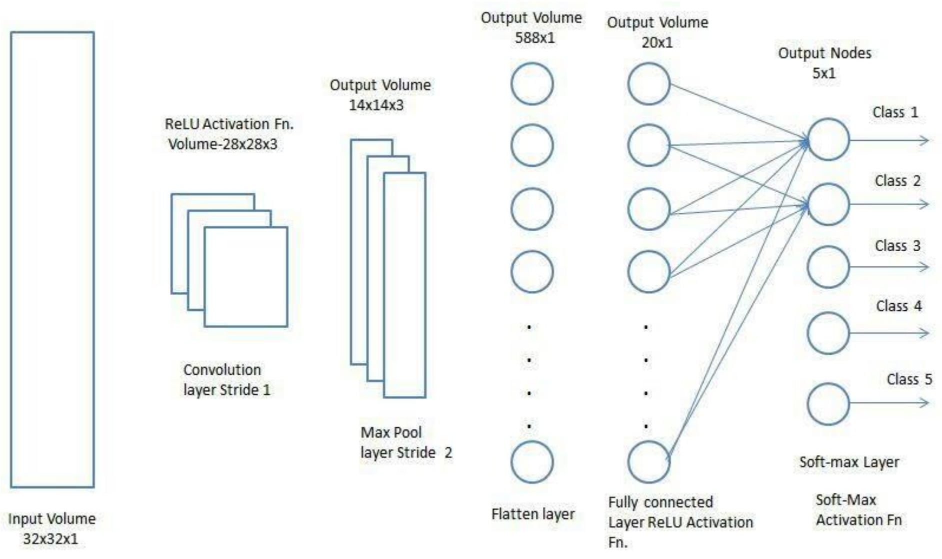

Fig 4.6.3 Fully connected layers

A convolutional neural network's convolutional layers methodically apply previously learned filters to input images to create feature maps that list the features present in the input.

Convolutional layers exhibit good performance, and stacking convolutional layers in deep models permits learning of high-order or more abstract features by layers deeper in the model while learning of low-level features by layers closer to the input, such as lines.

One of the drawbacks of convolutional layer feature maps is that the precise position of the input characteristics is preserved. This suggests that even little adjustments to the feature's position in the input picture will result in a distinct feature map. This might be caused by re-cropping, rotation, shifting, and other small changes to the provided picture.

In signal processing, down sampling is a widely used technique to address this problem. In this situation, the input signal is converted into a lower resolution version while keeping the essential structural elements and excluding any unnecessary fine information.

Convolutional layers can be used to do down sampling by adjusting their across-image stride. Using a pooling layer is a more dependable and common method.
A new layer known as a pooling layer is added after the convolutional layer. For instance, when a nonlinearity (like ReLU) has been applied to the feature maps generated by a convolutional layer, the layers of a model may seem like follows:

1. Input Image
2. Convolutional Layer
3. Nonlinearity
4. Pooling Layer

The insertion of a pooling layer following the convolutional layer is a typical layer ordering pattern in a convolutional neural network that may be repeated once or more times in each model.
The pooling layer performs individual processing on each feature map, creating a new set of the same number of pooled feature maps.
Pooling is carried out in order to apply a pooling operation—similar to a filter—on feature maps. The pooling operation's or filter's size, which is generally 22 pixels applied with a stride of 2 pixels, is lower than the feature map's size.
Because each dimension is divided in half, each feature map will always be compressed by a factor of 2, making each feature map just half as big in terms of pixels or values. For instance, adding a pooling layer to a feature map of 66 (36 pixels) will result in an output pooled feature map of 33. (9 pixels).
The pooling operation is specified rather than instructed. There are two often used functions in the pooling operation**:**
Average Pooling: Calculate the average value for each patch on the feature map.
Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.
A condensed version of the input features is produced by employing a pooling layer and down sampling or pooling feature maps. They are helpful because a pooled feature map with a feature in the same place is produced when a feature's location in the input, as identified by the convolutional layer, changes little. The ability that pooling adds is referred to as the model's invariance to local translation.

4.7 Artificial Neural Networks:

A computational model called an artificial neural network (ANN) is modelled after the form and operation of biological neural networks in the human brain. It is made up of numerous interconnected nodes, or neurons, arranged in layers. Up until a final output is produced, each neuron receives inputs, processes them, and transmits the results to the layer of neurons below it.

Applications for machine learning and artificial intelligence, such as speech and image recognition, natural language processing, and predictive analytics, frequently use ANNs. A collection of input-output pairs is used to train the network, which then learns to identify patterns and relationships in the data and provide predictions using that knowledge.

ANNs come in a variety of forms, including as convolutional, recurrent, and feedforward networks. The simplest sort of network is a feed-forward network, which has an input layer, one or more hidden layers, and an output layer. Recurrent networks are made to handle data sequences, such as text or time series data. Convolutional networks, which can automatically train to recognise features in images, are frequently employed for image identification applications.

Prior to their development, artificial neural networks (ANNs) were believed to be unable to address complicated issues. But, in order to train efficiently, they need a lot of data and computer power, and it can be challenging to understand and explain how they make decisions.

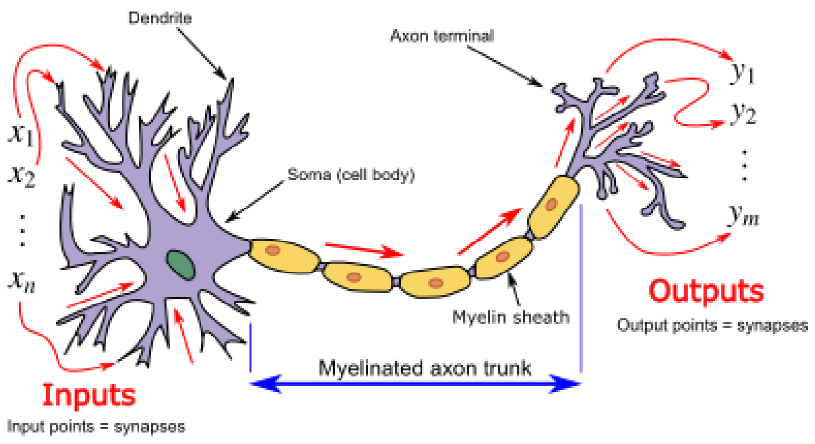

Fig 4.7 Neural network

The human brain is made up of 86 billion neurons, or nerve cells. They are connected to a million more cells through axons. Dendrites take in information from sensory organs as well as external inputs. These inputs generate electric impulses, which swiftly move across the brain network. A neuron has the choice to either cease processing the message or pass it through to another neuron for processing.

ANNs, which resemble the organic neurons present in the human brain, are made up of several nodes. The neurons are connected by links, and they communicate with one another. The nodes have the capacity to accept input data and process it using simple operations. The results of these activities are received by other neurons.

4.7.1 Neural network:

A computational model known as a "neural network" is modelled after the form and operation of biological neural networks seen in the human brain. It is made up of numerous interconnected nodes, or neurons, arranged in layers. Up until a final output is produced, each neuron receives inputs, processes them, and transmits the results to the layer of neurons below it.

Artificial intelligence and machine learning applications, such as speech and image identification, natural language processing, and predictive analytics, frequently use neural networks. A collection of input-output pairs is used to train the network, which then learns to identify patterns and relationships in the data and provide predictions using that knowledge.

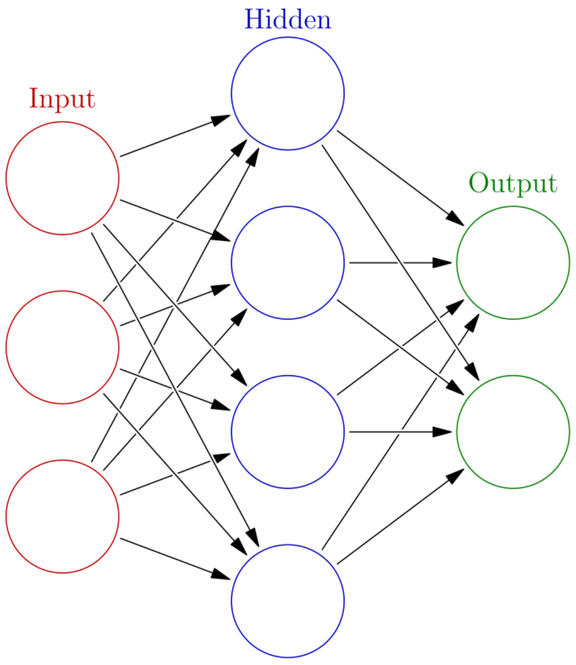

Fig 4.7.1 A simple neural network

Neural networks come in a variety of shapes, such as feedforward, recurrent, and convolutional networks. The simplest sort of network is a feed-forward network, which has an input layer, one or more hidden layers, and an output layer. Recurrent networks are made to handle data sequences, such as text or time series data. Convolutional networks, which can automatically train to recognize features in images, are frequently employed for image identification applications.

There are many different types of neural networks, including feedforward, recurrent, and convolutional networks. A feed-forward network, which contains an input layer, one or more hidden layers, and an output layer, is the most basic type of network. Data sequences, like text or time series data, are what recurrent networks are designed to manage. Convolutional networks are often used in image recognition applications because they can automatically learn to recognise features in images.

4.8 Training of CNN:

In order to get the intended network output, CNNs and ANNs in general require learning algorithms to alter their free parameters. Backpropagation is the most popular algorithm used for this purpose. Backpropagation determines how to modify a network's parameters in order to minimize errors that have an impact on performance by computing the gradient of an objective function. Overfitting, which is poor performance on a held-out test set after the network has been trained on a small or even large training set, is a typical issue with training CNNs, and DCNNs. This poses a significant barrier to DCNNs and has an impact on the model's capacity to generalize on unknown data.

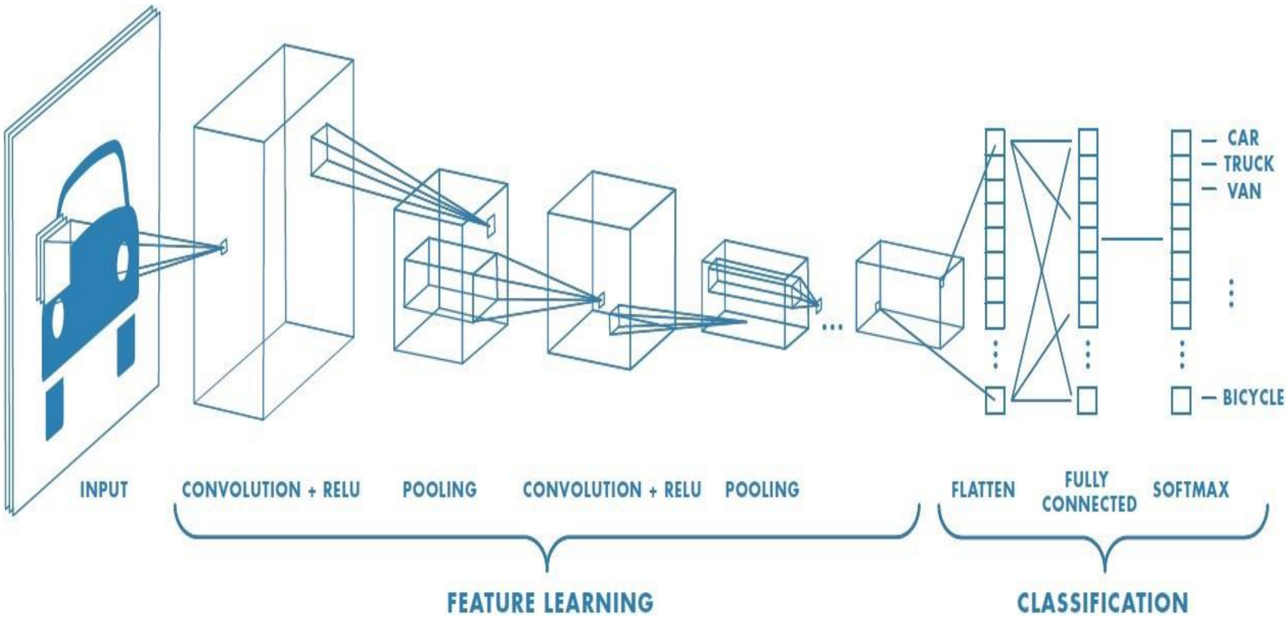

Fig 4.8.1 Training process of CNN

The "training" of the neural network is the process of changing the weights' values.

The CNN starts off by using random weights. During CNN training, a sizable dataset of pictures with their matching class labels is fed into the neural network (cat, dog, horse, etc.). Each image is processed by the CNN network with values assigned at random, and after that, comparisons are made with the class label of the original image. If the output does not match the class label (which typically occurs early in the training process), the CNN algorithm makes a slight modification to the CNN neurons' weights to ensure that the output accurately fits the class label image. Using a process called backpropagation, adjustments are being made to the weights' values. Backpropagation streamlines tuning and facilitates modifications for greater precision. Every iteration of the picture dataset's training is referred to as a "epoch." Throughout the training process, the CNN travels through many series of epochs, altering its weights by the necessary little amounts. The neural network gets a little bit better at correctly categorizing and predicting the class of the training images after each epoch step. The weights are adjusted, but as CNN becomes better, the weight modifications get smaller and smaller. We employ a test dataset to assess the accuracy of the CNN after it has been trained. A collection of labelled photos that were not included in the training phase make up the test dataset. Each image is sent to CNN, whose output is then compared to the test image's real class label. The test dataset essentially assesses how well the Network performs predictions. A CNN is considered to be "overfitting" if its accuracy is good on training data but poor on test data. This is the result of the dataset being too small (training).

4.9 CNN Architecture:

The CNN architecture mainly consists of a list of layers that transform a 3-dimensional input volume of pictures into a 3-dimensional output volume. A N*N filter is applied to the input picture by connecting each neuron in the subsequent layer to a little portion of the output from the layer before. This is an important information to remember.

It employs M filters, or feature extractors, which take out features like corners, edges, and so on. This is a list of the layers [INPUT-CONV-RELU-POOL-FC] that make up convolutional neural networks (CNNs).

• INPUT — As its name implies, this layer stores the raw values of the pixels. Raw pixel values relate to the actual data of the image. INPUT [64643], for instance, is a 3-channeled RGB picture with 64 width, 64 height, and 3 depths.

• CONV—This layer, which is one of the building blocks of CNNs, is where most of the computation takes place. For instance, if we apply 6 filters to the INPUT [64643], the volume [64646] might result.

• RELU, sometimes referred to as a rectified linear unit layer, which augments the output of the layer preceding it using an activation function. In addition, the network would become non-linear due to RELU.

• POOL Another element of CNNs is this layer, also referred to as the pooling layer. Down sampling, which comprises separate operations on each slice of the input and spatial scaling, is the main purpose of this layer.

• FC refers to this layer as the fully connected layer, or more particularly as the output layer. It produces a volume of size 1*1*L, where L is the integer corresponding to the output class score, from which the output class score is calculated.

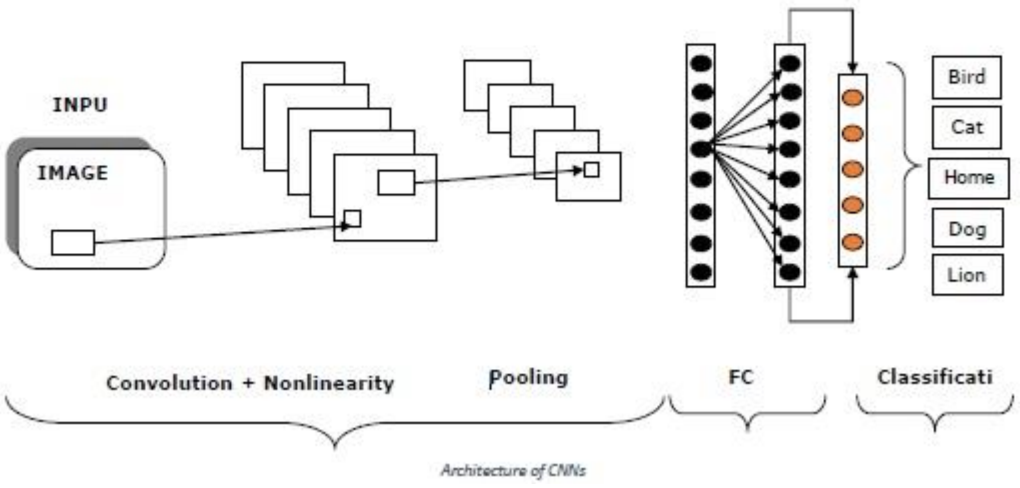

Fig 4.9 Architecture of CNN

CHAPTER 5

SOFTWARE TOOLS

5.1 INTRODUCTION:

The YOLO method was developed and implemented using a variety of software tools. Among the most popular software applications are:

1. Darknet: YOLO is implemented using the framework for open-source neural networks which is Darknet. supports CPU and GPU acceleration and is written in C and CUDA. Darknet offers a selection of models that have already undergone training for tasks involving computer vision which include object detection.
2. OpenCV: For processing videos and images, OpenCV is an open-source computer vision library. It offers a selection of features for object tracking, feature identification, and image editing. Darknet can utilize OpenCV to pre-process input image input and post-process YOLO output.
3. Python: Python is a popular programming language that is used for YOLO implementation as well as other machine learning and deep learning activities. TensorFlow, Keras, PyTorch, and Scikit-learn are just a few of the machine learning libraries and frameworks available in Python.
4. CUDA: Developed by NVIDIA for GPU acceleration, A parallel computing environment, and programming language is called CUDA and on NVIDIA GPUs, Darknet and CUDA can be utilized to accelerate the YOLO algorithm.
5. Google Collaboratory: Also known as Colab, this cloud-based development environment gives users free access to TPUs and GPUs for deep learning and machine learning applications. Darknet, Python can be used with Colab to train and test YOLO models.

A wide range of capabilities are offered by these software solutions for the implementation and improvement of the YOLO algorithm. Several combinations of these tools can be utilized to achieve optimum performance depending on the precise task at hand and the necessary resources.

5.2 OPEN CV:

Developers can create software programs that process and analyze visual data using the OpenCV software which is a suite for computer vision and machine learning. OpenCV, which was created by Intel in 1999, has undergone constant development by the open-source community and is now extensively utilized in fields including robotics, autonomous driving, security, and entertainment. For applications in image processing, computer vision, machine learning, and deep learning for recognizing faces and objects, respectively, motion analysis, and picture segmentation, OpenCV offers a complete range of functions and algorithms. The C++-based, performance-optimized library offers bindings for Python, Java, and other languages. OpenCV has gained popularity among researchers thanks to its wide array of tools and methods, both companies and individuals interested in developing computer vision applications. For the library to continue to be a leader in the computer vision industry, new features are continually being added. A well-known open-source machine learning and computer vision machine learning library is OpenCV (Open-Source which a Computer Vision Library). Although a sizable development community now maintains it, Intel first developed it in 1999.OpenCV offers a set of potent tools and algorithms that let programmers handle and analyze visual input. Although the library was created in C++, it also has Interfaces for Python, Java, and MATLAB.

OpenCV's primary features include the following:

1. Processing of images and videos: OpenCV includes tools for reading, writing, and modifying many formats of photos and movies. A comprehensive range of image processing procedures are also included, including feature detection, morphological operations, and picture filtering.
2. Object Detection and Tracking: OpenCV comes with a number of techniques for detecting and tracking objects, such as HOG (Histogram of Oriented Gradients) and Haar cascades, features, and deep learning-based methods.
3. Machine Learning: Support vector machines (SVMs), decision trees, and neural networks are just a few of the machine learning methods that are included in OpenCV. Tasks requiring classification, regression, and grouping can be accomplished using these algorithms.
4. Deep Learning: TensorFlow, Keras, and PyTorch are just a few of the frameworks for deep learning that OpenCV supports. This makes it possible for programmers to create intricate deep learning models for tasks like segmentation, object detection, and picture recognition.
5. 3D Vision: OpenCV has tools for structure from motion, stereo vision, and 3D reconstruction. These technologies can be used for activities like 3D object tracking and depth estimation.

Robotics, autonomous driving, security systems, healthcare, and entertainment are just a few of the fields and applications where OpenCV is extensively employed. New features and enhancements are continuously being added to the library by the developer community. The initial alpha release of OpenCV was made public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five beta releases followed between 2001 and 2005. The initial 1.0 version was released in 2006. A version 1.1 "pre-release" was made accessible in October 2008. In October 2009, OpenCV's second big update was released. OpenCV 2 makes a significant enhancement to the C++ interface with the goal of facilitating easier, new functions, more type-safe patterns, and improved performance implementations for existing ones particularly Several sectors and applications, including robotics, autonomous driving, security, and healthcare, making extensive use of OpenCV. Formal releases now every six months, and development is now being done by a self-sufficient Russian team with financial support from businesses. In August 2012, a nonprofit organization called OpenCV.org, which runs a developer and user website, took over maintenance for OpenCV. In August 2012, a nonprofit organization called OpenCV.org, which runs a developer and user website, took over maintenance for OpenCV. A renowned OpenCV developer, ITSEEZ, and Intel reached an acquisition agreement in May 2016. Real-time computer vision is the main emphasis of the OpenCV (Open-source Computer Vision) collection of programming functions. It was first developed by Intel, and later on, Willow Garage and Itseez provided funding (which was later acquired by Intel). The library is open-source and free to use under the conditions of the BSD license. It offers C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS. OpenCV makes extensive use of real-time vision applications and makes use of MMX and SSE instructions when they are released. The development of a fully functional CUDA and OpenCL interface is ongoing.

5.3 GOOGLE COLAB:

Google Colab, sometimes also referred as Google Collaboratory, is a cloud-based platform that Google makes available to researchers and developers for the purposes of deep learning, machine learning, and data analysis. With the help of a web browser, users may write, execute, and share Python code in this environment's Jupyter Notebook format.

Google Colab has many salient features, including:

- Free to use: Google's powerful computer capabilities are available to those who create and use Colab notebooks for free.
- Collaborative: Users can share Colab notebooks with one another to collaborate in real-time.

- Simple access to GPUs and TPUs: Colab offers simple accessibility to powerful computing tools like GPUs and TPUs, which can greatly accelerate computations for machine learning workloads.
- Pre-installed libraries: TensorFlow, PyTorch, and sci-kit-learn are just a few of the pre-installed libraries that come with Colab and are frequently used in data science and machine learning.
- Google Drive integration: Colab's integration with Google Drive makes it simple for users to access and store their notes and data.

Overall, Google Colab is an effective tool for deep learning, machine learning, and data analysis, and it offers researchers and developers a convenient environment.

5.4 PYTHON:

A high-level, interpreted programming language is one of the main features Python is widely used in many different industries, such as artificial intelligence, data analysis, web development, scientific computing and more. Guido van Rossum first made this available in 1991, and since then it has grown to be most widely used programming languages worldwide.

Python's readability and simplicity, which make it simple to learn and use, are two of its distinguishing characteristics. While writing Python code, complicated ideas are frequently expressed using a combination of indentation and plain English keywords in a clear, simple manner. Python is an interpreted language, which means that an interpreter runs the code line by line rather than compiling it into machine code beforehand.

Many features, including as support for file I/O, networking, regular expressions, and others, are available in Python's extensive standard library. Additionally, there are numerous third-party libraries and frameworks that enhance Python's functionality for certain use cases. Examples include TensorFlow for machine learning, Django for web development, and NumPy for scientific computing. The Python programming language is frequently used to implement the YOLO (You Only Look Once) method. TensorFlow, PyTorch, and Darknet are just a few of the well-known deep learning frameworks that use the technique. Installing the appropriate deep learning framework and the desired YOLO implementation is often required to utilize YOLO in Python. To utilize YOLO with TensorFlow, for instance, you first need to install TensorFlow before downloading and configuring the YOLO implementation for TensorFlow. Once the YOLO implementation is configured, you may import an image or video and use Python code to send it across the YOLO network for object recognition. The YOLO algorithm will locate things in the picture or video and provide bounding boxes and labels for each object it finds.

5.5 OpenCV's applications:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human-computer interaction (HCI)
- Structure from motion (SFM)
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Mobile robotics
- Motion tracking
- Robotics
- Autonomous vehicles
- Medical imaging
- Augmented reality
- Security and surveillances
- Entertainment
- Agriculture

5.6 Google Colab application:

Due to its capacity to offer a potent computing environment for data analysis, machine learning, and deep learning activities without requiring users to have access to expensive hardware, Google Colab has a wide range of applications across numerous fields. Google Colab's popular applications include the following:

1. Data analysis: Using Python and well-known libraries like Pandas, NumPy, and Matplotlib, Colab offers an interactive environment for carrying out data analysis tasks. Users can import data from different sources, clean and prepare data, and carry out data analysis and visualisation.
2. Machine learning: Colab gives users access to strong computing tools like GPUs and TPUs, which can be utilised to train machine learning models more quickly and effectively. It is a well-liked option for machine learning research and development because it supports well-known machine learning frameworks like TensorFlow, PyTorch, and Scikit-learn.

3. Deep learning: Because it supports frameworks like Keras and TensorFlow, Colab is especially well-suited for deep learning jobs. Users can create new material using generative models like GANs, conduct image and text recognition jobs, train deep neural networks on massive datasets, and more.
4. Education: In the classroom and online, Colab is a well-liked tool for teaching Python and data science principles. It gives students an easy-to-use environment in which to develop and execute Python code, and it enables professors and students to exchange and collaborate on notebooks.
5. Research: Colab is used to conduct experiments, analyse data, and create new algorithms and models by researchers working in a variety of disciplines, including computer science, physics, and biology.

Overall, Google Colab is a flexible tool with a variety of applications, and as more users learn about it, the more popular it becomes.

5.7 Libraries in open CV:

For a variety of computer vision and machine learning activities, OpenCV (Open-Source Computer Vision Library) offers a large selection of libraries. The following are some of the key libraries that are included with OpenCV:

1. Core Library: The Core Library offers fundamental data structures and image processing capabilities, including Mat objects for storing images and functions for manipulating images.
2. Image Processing Library: This library offers functions for fundamental image processing operations as thresholding, filtering, and morphological procedures
3. Feature Detection and Description Library: This library offers algorithms, like SIFT, SURF, and ORB, for identifying and describing features in images.
4. Video Analysis Library: The Video Analysis Library offers tools for analysing videos, such as motion detection, object tracking, and optical flow
5. Machine Learning Library: This is a collection of methods for classification, regression, and clustering includes decision trees, k-NN, and SVM.
6. Deep Learning Library: TensorFlow, Keras, and PyTorch are just many of the deep learning frameworks that OpenCV supports.
7. Calibration Library: For 3D vision applications, this library offers capabilities for stereo calibration and camera calibration.
8. Object Detection Library: The Object Detection Library contains a number of algorithms, including deep learning-based methods, Haar cascades, and HOG features, for identifying things in pictures and videos.

9. 3D Vision Library: The 3D Vision Library offers tools for stereo vision, structure from motion, and 3D reconstruction.

These are only a few of the libraries that are offered by OpenCV. OpenCV offers a complete collection of functionalities for creating computer vision applications with its large array of tools and methods.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 RESULT:

The YOLO algorithm uses a deep convolutional neural network to identify objects in an input image. A single neural network is used by the algorithm to process the entire full image. The network then separates the image into areas, each of which contains bounding boxes and probabilities. Experiments were conducted on a database created from video recordings of automobiles on the road and highways, as well as from various types of movies. We have selected a traffic surveillance video that uses our method; the image displays object tracking images for surveillance cameras and includes several automobiles. The movie was broken into frames by the YOLO Algorithm, which carried out object detection. we have observed the object detection in the below image. The model's object detection is shown in the figure below. Bounding Boxes will appear when the objects are detected. The probability that object is a person is 0.99, the bench is 0.27 and the car is 0.94, the bicycle is 0.84 and the truck is 0.86 respectively. We have performed object detection in Yolo v3 and Yolo v4 respectively. We can observe the difference in the results shown in the images below. We chose Yolo v4 as our algorithm but compared it with Yolo v3 algorithm. Yolov4 is an enhancement to Yolov3. Now let us observe the results shown in the figures given below.

Object detection for traffic surveillance video

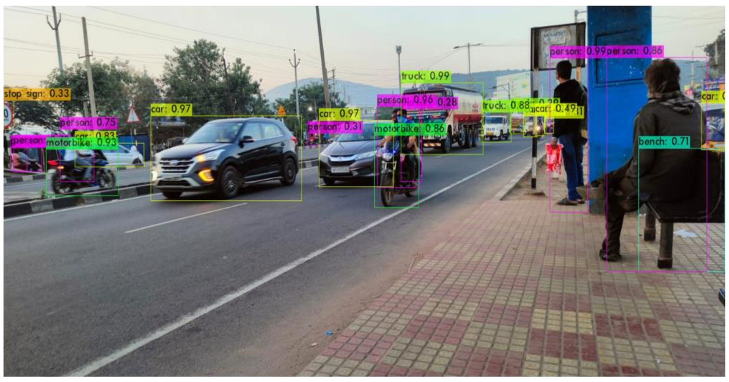

Fig: 6.1 Object detection using yolo v3

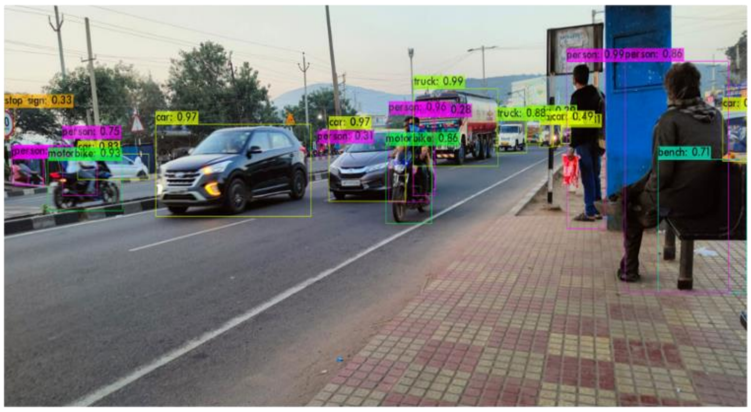

Fig: 6.2 Object detection using yolo v4

Object tracking for traffic surveillance video

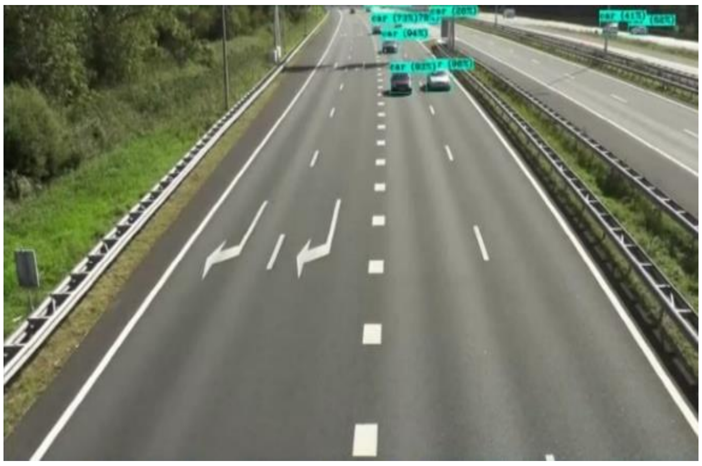

Fig 6.3.1. Object tracking in a video at frame 1

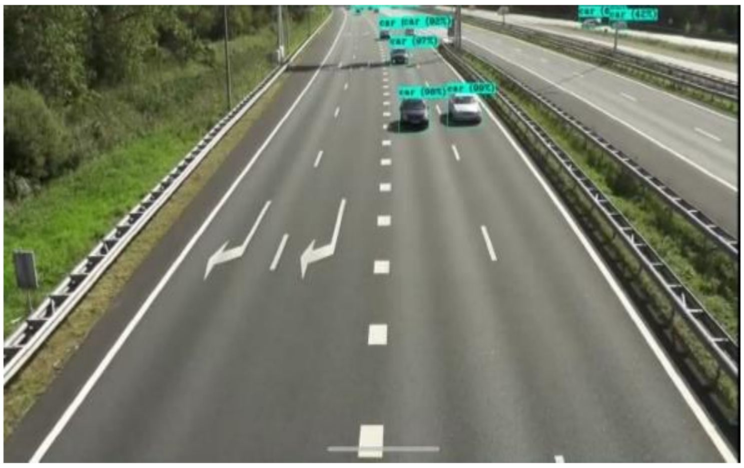
Fig 6.3.2. Object tracking in a video at frame 2

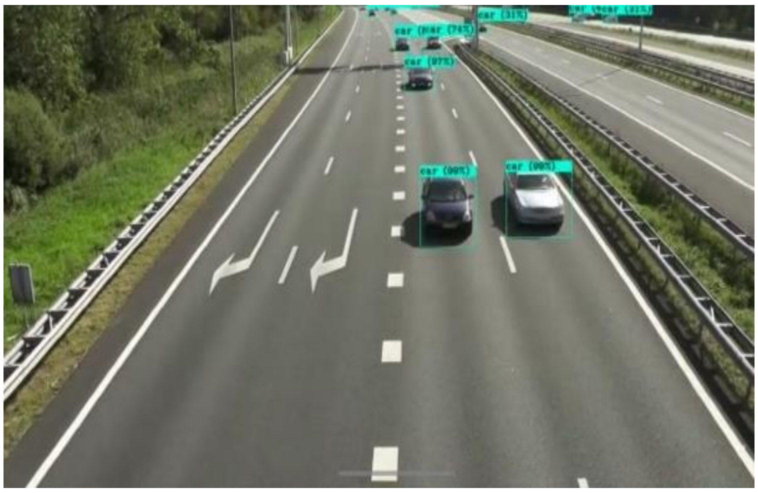
Fig 6.3.3. Object tracking in a video at frame 3

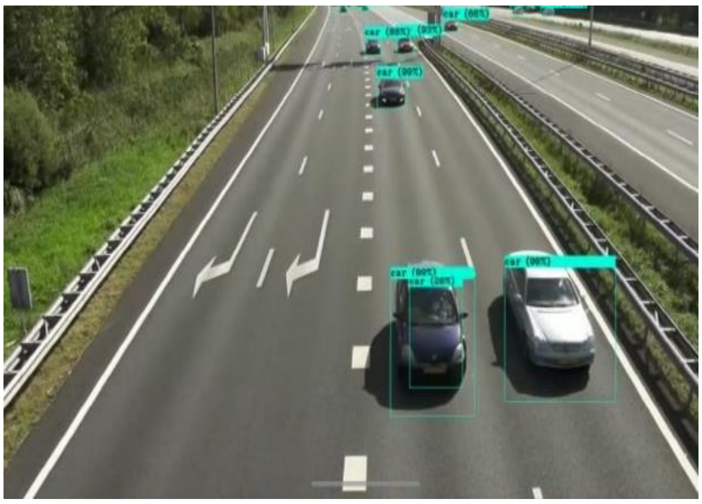

Fig 6.3.4. Object tracking in a video at frame 4

+

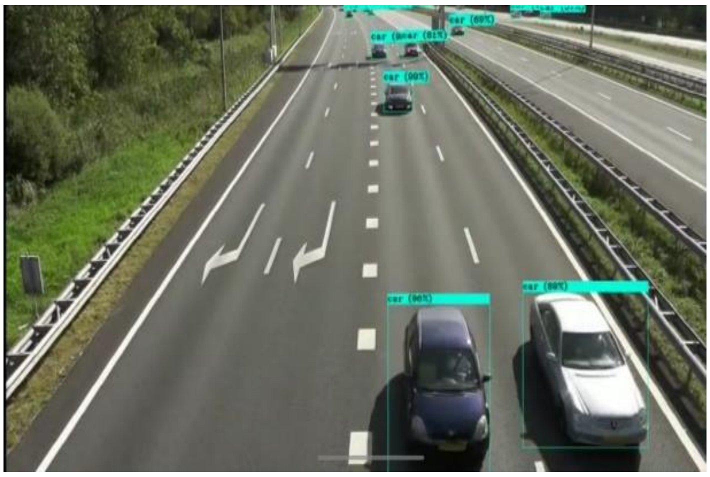

Fig: 6.3.5 Object tracking in a video at frame 5

Figure depicts the images of object tracking for surveillance video, which contains several automobiles. We have selected a traffic surveillance video that is an application of our technique. The first frame of the video was used for object recognition by the YOLO Algorithm. The specific detected image was tracked in the following frames. As depicted in the picture, objects are tracked throughout a variety of frames and time intervals.

Real-time object tracking using YOLO

Using the YOLO technique, we carried out real-time object recognition and tracking. Entity detection is carried out via YOLO, which presents the likelihood of the projected class while treating the issue as a regression problem. Convolutional neural networks (CNN) are used by the YOLO method to detect and track objects in real-time. A webcam is used to record the images and videos used as input. Following entity detection and the determination of each object's confidence score, which indicates how accurate the object is as the output of each frame, these profiles are then provided to the machine learning model. The Machine Learning model was tested on several things, and as a result, it correctly identified the objects in the video frame with a good prediction probability and provided the confidence score, which represents how confident the model is and how accurately the boundary box appears depends on whether the box contains the object.

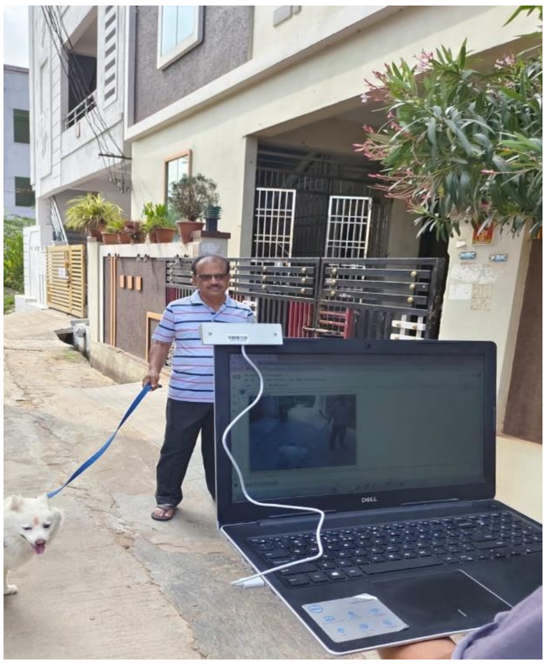

Fig:6.4 Illustration of Experimental set-up

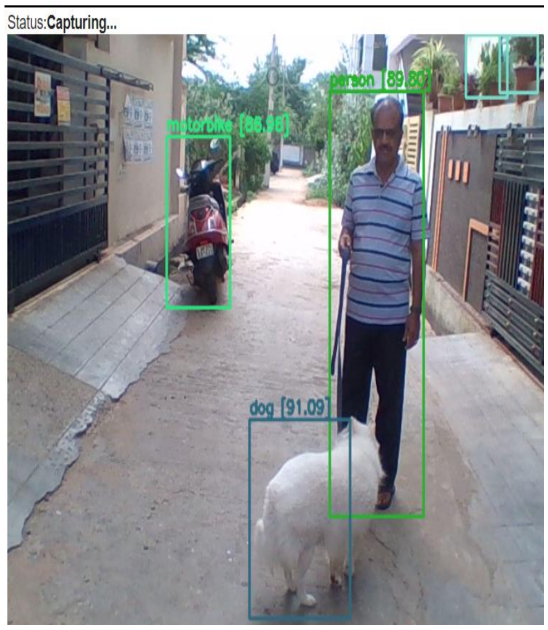

Fig:6.5 object detection by the model in the video

6.2 CONCLUSION:

In order to accomplish object recognition and tracking, this project work proposes a straightforward, reliable methodology that can be used with object detection and tracking. This technique makes use of the YOLO Algorithm, which can anticipate and categorize bounding boxes in a single forward pass. This approach can purposefully improve the performance of detection. Compared to traditional machine learning algorithms, it is substantially faster. On the basis of accuracy, robustness, and computational effectiveness, the algorithms are evaluated. In this project, a comparative analysis of various object-identification techniques including RCNN, Faster RCNN, and YOLO is conducted. We discovered that YOLO is a lot quicker and more accurate than other object detection techniques. Thus, we trained the algorithm using the datasets. Real-time detection and tracking of the objects are possible with the YOLO algorithm. The camera module, which is a device that can be linked to a computer or desktop, provides the necessary input image. The PC or desktop itself allows us to see the outcomes. This technology could be applied and used in a variety of fields, including traffic analysis, face detection, medical image processing, and security monitoring.

FUTURE SCOPE:

Object tracking is being more widely adopted by corporations, with uses ranging from personal security to workplace productivity. Object tracking is used across a wide range of image processing applications, including image retrieval, security, surveillance, automated driving systems, and machine analysis. There are still significant challenges in the realm of object detection. Regarding prospective outcomes for future use cases of object tracking, the possibilities are incalculable.

Applications for object tracking include traffic analysis, surveillance and security, video correspondence, robot vision, and activity. Counting people can also be done using object detection. It is used to analyze festival crowd measurements or retail performance. They will typically become more challenging when people leave the picture quickly, likewise because individuals are non-inflexible objects). Person detection is a pivotal and required task in any intelligent video surveillance system because it provides the information needed to understand the semantics of the video recordings.

Due to the potential for enhancing security frameworks, it has a noticeable augmentation to automotive applications. Human detection is a task that computer vision frameworks undertake for locating and tracking people. Finding every instance of a person in a photograph is the challenge of person detection, which has most commonly been accomplished by scanning the entire image at all possible scales and comparing a small area at each scale with the known arrangements of people.

REFERENCES:

[1] Mohana, HV Ravish Aradhya, "Object Detection and Tracking using Deep Learning and Artificial Intelligence for Video Surveillance Applications," (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 12, 2019, pp. 517-530.

[2] V. D. Nguyen et all., "Learning Framework for Robust Obstacle Detection, Recognition, and Tracking", IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 6, pp. 1633-1646, June 2017.

[3] P. Wang et all., "Detection of unwanted traffic congestion based on existing surveillance system using in freeway via a CNN-architecture traffic net", IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, 2018, pp. 1134-1139 .

[4] H. C. Baykara et all., "Real-Time Detection, Tracking and Classification of Multiple Moving Objects in UAV Videos", 29th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, 2017, pp. 945-950.

[5] K. Muhammad et all., "Convolutional Neural Networks Based Fire Detection in Surveillance Videos", IEEE Access, vol. 6, pp. 18174- 18183, 2018.

[6] Redmon, Joseph, et al." You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern Recognition', 2016.

[7] Aloysius, Neena, and M. Geetha." A review on deep convolutional neural networks." 2017 International .

[8] Z. Jiang, L. Zhao, S. Li and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny, " ArXiv, vol: abs/2011.04244, 2020.

[9] K. M. Babu and M. V. Raghunadh, "Vehicle number plate detection and recognition using bounding box method," 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 2016, pp. 106-110, Doi:

[10] K. V. Arya, S. Tiwari and S. Behwalc, "Real-time vehicle detection and tracking," 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2016, pp. 1-6, Doi: 10.1109/ECTICon.2016.7561327.

[11] M. A. Bin Zuraimi and F. H. Kamaru Zaman, "Vehicle Detection and Tracking using YOLO and Deep SORT," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2021, pp. 23-29, Doi: 10.1109/ISCAIE51753.2021.9431784.

[12] X. Gu, Z. Chen, T. Ma, F. Li and L. Yan, "Real-Time vehicle detection and tracking using deep neural networks," 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2016, pp. 167-170, Doi: 10.1109/ICCWAMTIP.2016.8079830.

[13] Z.Q. Zhao, P. Zheng, S.T. Xu, and X. Wu, "Object detection with deep learning,": A review. arXiv e-prints, arXiv:1807.05511, 2018.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection, " In 2016 IEEE conference on computer vision and pattern recognition, doi.org/10.1109/cvpr.2016.91 (pp. 779–788): IEEE.

[15] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517-6525, Doi: 10.1109/CVPR.2017.690.

[16] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018. arXiv preprint arXiv:1804.02767.

[17] A. Bochkovskiy, C.Y. Wang, and H.Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[18] Mahalingam, T. and Subramoniam, M. (2020), "A robust single and multiple moving object detection, tracking and classification", Applied Computing and Informatics, Vol. ahead-ofprint No. ahead-of-print,