Sparse R-CNN Object Detection Using Proposal Boxes

A DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE

Submitted by:

Yashasvi Pasbola

2K21/AFI/16

Under the supervision of

Dr. Shailender Kumar

(Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

JUNE 2023

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I,**Yashasvi Pasbola**,2K21/AFI/16 of M.Tech in Artificial intelligence, hereby declare that the report Dissertation titled "**Sparse R-CNN Object Detection Using Proposal Boxes**" which is submitted by me/us to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Associateship, Fellowship or other similar title or recognition.

Place: Delhi Yashasvi Pasbola

Date:

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

<u>CERTIFICATE</u>

This is to certify that the Report Dissertation titled "**Sparse R-CNN Object Detection Using Proposal Boxes**" which is submitted by **Yashasvi Pasbola** ,2K21/AFI/16 from department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the report work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi   Prof.Shailender Kumar

Date:

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGMENT

I would like to express my gratitude towards my mentor, Professor Dr. Shailender Kumar, for giving me the opportunity to work in an impressive area such as deep learning and his guidance, regular supervision, constructive feedback and suggestions in completing the project.

Also I would like to thank my parents, my brother and my friends for their love and continuous support throughout my life. Thank you for always giving me the strength to successfully finish this project.

Yashasvi Pasbola

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Artificial Intelligence has advanced a lot over the last few years to solve various challenges in the modern day world. One of these challenges is the recognition of images digitally and extracting important features.The topic of deep learning has created an impact in this field and the major algorithm that contributed is convolutional neural network.Convolutional neural network has been continuously improving to get much efficient algorithms to extract the features better.This report will be covering object detection various algorithms and how they work. The methods to detect objects are mainly divided into three groups: sparse methods, dense methods and dense to sparse.The dense methods divides image to grid to slide proposal box to detect the object, while the dense to sparse tries to find important regions where the object will be present using region proposal.

Both of these methods are great but to further improve the algorithm , sparse RCNN was introduced which takes away less user input parameters to calculate the location and detect where the object is present in the image. It uses iterative learning to predict where the object will be present in the image with different sizes of proposal boxes.Then it classifies and predicts what object it is thus creating more efficiency.

Sparse R-CNN Object Detection Using Proposal Boxes

1.Introduction

1.1.Introduction to Convolutional neural networks

Convolutional neural networks (CNNs) are exerting a significant influence on the world. Presently, deep learning falls within the realm of machine learning and specifically deals with unstructured data types like images, videos, and music, processing them effectively. Within deep learning, CNNs serve as a foundational framework and have successfully addressed various challenges, including object detection, face recognition, and advanced medical treatments. In the context of face recognition, for instance, facial attributes like the mouth, nose, and eyes are identified and stored as a feature vector. This feature vector is then compared with those of other individuals to recognize their faces. In medical applications such as X-ray and MRI imaging, CNNs have significantly contributed to accurate image recognition. Machine learning algorithms utilising CNNs can readily identify conditions like broken bones, tumours, and cancer by comparing the images with a database of previous patient records. In the United States, autonomous vehicles are increasingly prevalent, and they also employ CNNs to perceive the environment and detect objects or people in close proximity. Additionally, document analysis tasks, such as character recognition from images and replication, can be effectively accomplished using CNNs.

Until now, researchers and developers have explored and devised various types of convolutional neural networks (CNNs) for object detection. The fundamental concept behind a CNN is the utilisation of a kernel that slides over an image to extract significant features. Among the different approaches, three main types stand out: dense methods, dense-to-parse methods, and sparse methods.

Dense methods employ sliding window features, where anchor boxes of various sizes are used to detect objects. By sliding these anchor boxes across the image, the network aims to identify objects present within the image.

Dense-to-parse methods, on the other hand, rely on region-based techniques to detect edges and extract important features for object detection. This approach focuses on parsing regions of interest to identify objects accurately.

The latest addition is the sparse method, which utilises proposal boxes to detect objects. These proposal boxes are trained using previous images and attempt to predict the likely location of objects within the image.

Overall, these different approaches to object detection within CNNs have been extensively researched and developed, each with its own advantages and applications [1].

Let's start by exploring the evolution of convolutional neural networks (CNNs) from the nineteenth century to the present day. We'll delve into the fundamental workings of CNNs, including how they extract features, the role of the kernel, and the sliding window feature. We'll also examine different technologies utilised for object detection, such as Python [58], Detectron2 [29], and OpenCV [56].

Next, we'll delve into a comparison of various methods used for object detection, focusing on their advancements and improvements. Specifically, we'll discuss dense methods, dense-to-sparse methods, and sparse methods. We'll examine the key methodologies and functioning of Sparse R-CNN [1], exploring its different modules and how they interact with one another. We'll also compare Sparse R-CNN with other models in the field.

Lastly, we'll assess the results obtained from the model, considering its performance and effectiveness.

Object detection is the process of finding the location of an object as well as classifying what the object is.First let us see the history of object detection and how it continuously evolved over the years.

1.2.History

The history of convolutional neural networks can be traced back to the nineteenth century, where the foundation of artificial neural networks was established. In 1943, McCulloch and Pitts introduced the MP model, marking the initial development of artificial neural networks [2].

Advancements in the field continued in the following decades, particularly in the 1950s and 1960s, when Rosenblatt modified the MP model and introduced the single-layer perceptron model [3],[4]. However, the perceptron model faced limitations in solving linearly non-separable problems, as demonstrated by the XOR problem.

To address this challenge, another researcher named Hinton et al. proposed a groundbreaking solution in 1986. They introduced the multilayer feedforward network, which was trained using the error backpropagation algorithm. This model enabled the learning of complex representations and successfully tackled non-linear problems [5].

Delay Neural Network [6]. This model utilised a one-dimensional neural network and found significant application in speech recognition tasks.

Building upon this progress, Zhang [7] developed the first two-dimensional convolutional neural network in the same year. This network, known as the shift-invariant artificial neural network, made advancements in handling two-dimensional data effectively.

In 1989, LeCun et al. introduced a convolutional model specifically designed for recognizing zip codes [8]. Notably, this model marked the initial usage of the term "convolution" in the context of neural networks. Referred to as the original version of LeNet, this model laid the foundation for further advancements in convolutional neural networks.

In the 1990s, several shallow networks were developed, including the general regression neural network [8] and chaotic neural network [9]. However, one of the most widely recognized and influential networks during this period was LeNet-5 [10].

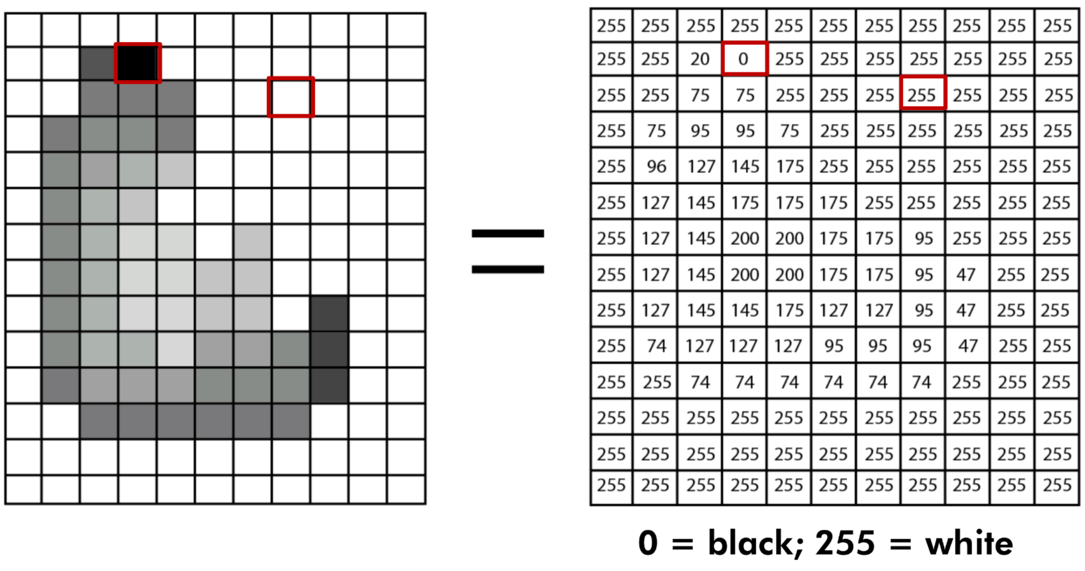
The evolution of convolutional neural networks faced significant challenges that hindered the accuracy of the models. One key obstacle arose when increasing the depth of neural networks. At the time, the backpropagation algorithm encountered various issues that were difficult to resolve. These problems included gradient vanishing, local optima, gradient exploding, and overfitting.

In 2006, Hinton et al. proposed a groundbreaking model [11] that introduced two key features. Firstly, it incorporated multi-layer hidden features, enabling high adaptive feature learning. This allowed the network to automatically learn and extract meaningful features from the input data. Secondly, the model introduced a layer-wise training approach, which helped overcome the challenges associated with training deep neural networks. By training one layer at a time, starting from the bottom and gradually moving upwards, the difficulties in optimising deep learning networks were mitigated.

In 2012, Alex et al. achieved remarkable success in the ImageNet Large Scale Visual Recognition Challenge [12]. They utilised a deep convolutional neural network architecture, known as AlexNet, which significantly outperformed other methods in image classification tasks. This breakthrough result attracted widespread attention and sparked a surge of interest in convolutional neural networks. It served as a catalyst for further advancements and the accelerated development of this field.

1.3.Image Representation and processing

An image is typically represented as a two-dimensional array consisting of pixels. Each pixel represents a specific point in the image and contains a value ranging from zero to two hundred fifty-five, indicating the intensity of light or colour at that particular location [59]. In computer systems, images are commonly displayed and processed using pixels as the fundamental unit of representation on a screen.



255	255	255	255	255	255	255	255	255	255	255
255	255	20	0	255	255	255	255	255	255	255
255	255	75	75	255	255	255	255	255	255	255
255	75	95	95	75	255	255	255	255	255	255
255	96	127	145	175	255	255	255	255	255	255
255	127	145	175	175	175	255	255	255	255	255
255	127	145	200	200	175	175	95	255	255	255
255	127	145	200	200	175	175	95	47	255	255
255	127	145	145	175	127	127	95	47	255	255
255	74	127	127	127	95	95	95	47	255	255
255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255

Figure 1.Image representation in form a two dimensional array [40].

After storing an image, we have the flexibility to perform various operations on it, such as rotation, scaling, and transformation. This process is known as image processing. Image processing allows us to manipulate and modify the image to enhance its quality, extract relevant features, or prepare it for further analysis using machine learning models.

Sampling

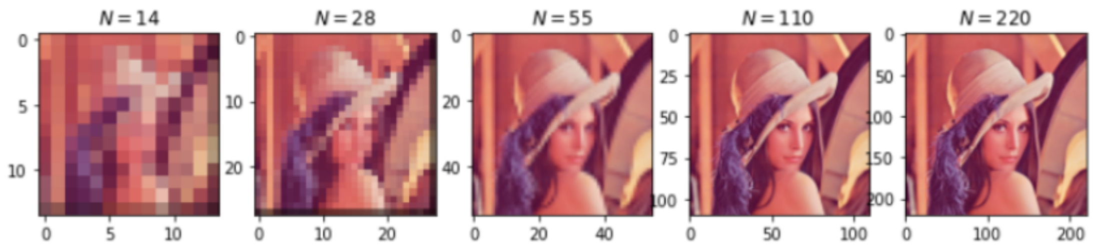


Figure 2.Sampling of image [37].

Sampling is a technique used to determine the spatial resolution of an image at different intervals. By increasing the value of N (the sampling interval), the resolution of the image increases, meaning more pixels are included in the image. Conversely, when the value of N is lower, the image appears blurrier as fewer pixels are present.

In the given figure, it can be observed that increasing the value of N leads to an increase in image resolution. However, at certain values of N, such as 110 and 220, there might not be a significant difference in resolution. In such cases, it is possible to use a lower resolution value to conserve computational resources while still maintaining an acceptable level of detail for the intended purpose, such as in a machine learning model [53]. This approach allows for a balance between image quality and computational efficiency.

Quantization

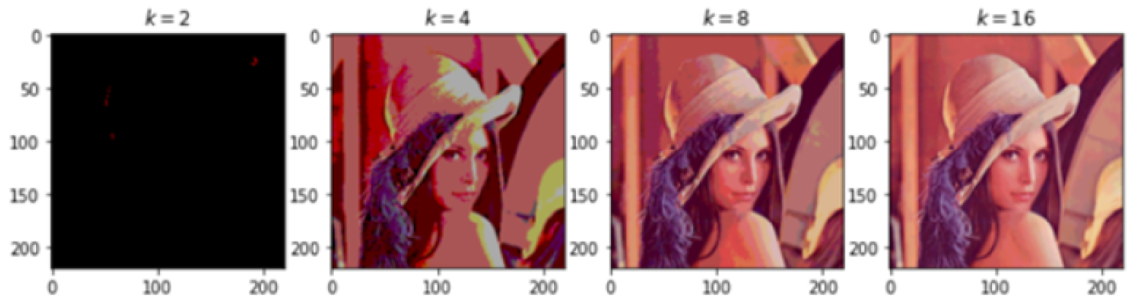


Figure 3.Quantization of the image [37].

Quantization is a process in image processing that involves determining the intensity levels or colours of an image. In the provided figure, it demonstrates the impact of the parameter k on the image. Here, k represents the number of colours or intensity levels that an image can be represented by.

For black and white images, where only shades of grey are present, a higher value of k may not be necessary. Since black and white images have a limited range of intensity values, employing a large number of colours or intensity levels would be unnecessary and wasteful in terms of storage and computational resources [53]. Therefore, a lower value of k can be sufficient to represent the image accurately and efficiently.

Image types

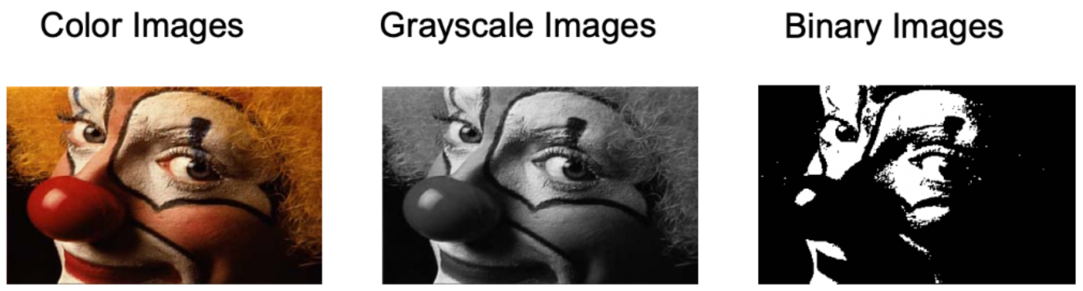


Figure 4.Image types [38].

In the provided figure, there are three main types of images depicted:

Binary Images: Binary images consist of only two colours, typically black and white. These images are represented by pixels that can be either on (white) or off (black). Binary images are commonly used in applications such as image segmentation, edge detection, and shape analysis.

Grayscale Images: Grayscale images have a range of intensity values from zero to two hundred fifty-five. These images contain shades of grey, where darker shades correspond to lower intensity values and lighter shades correspond to higher intensity values. Grayscale images are widely used in various image processing tasks, such as image enhancement, feature extraction, and texture analysis.

Coloured Images: Coloured images are composed of three colour channels: red, green, and blue (RGB). Each pixel in a coloured image is represented by a combination of intensity values for these three channels. By varying the intensity values for each channel, a wide range of colours can be achieved. Coloured images are commonly used in applications such as digital photography, computer vision, and multimedia systems.

By leveraging the appropriate image type, different image processing techniques and algorithms can be applied to perform specific tasks and extract meaningful information from the images [54].

Hue, Saturation and Value

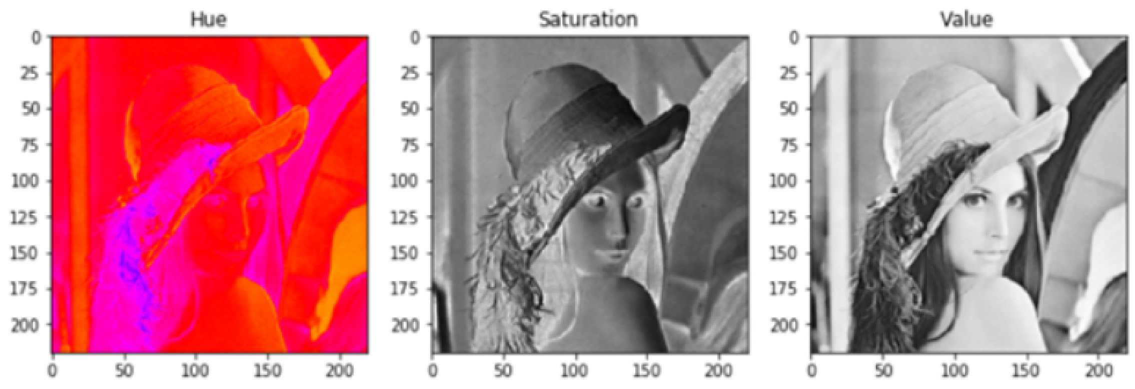


Figure.5 Hue , Saturation and Value [37].

In colour theory and image processing, the concept of colour can be described using three main components: hue, saturation, and value (HSV model). These components provide different aspects of the colour information:

Hue: Hue refers to the dominant colour in an image or pixel. It represents the specific colour itself, such as red, blue, or green. Hue is measured in degrees on a colour wheel, where each angle corresponds to a specific colour. For example, 0 degrees represents red, 120 degrees represents green, and 240 degrees represents blue.

Saturation: Saturation determines the purity or brilliance of a colour. It indicates the intensity or vividness of the hue. Higher saturation values indicate more intense and vibrant colours, while lower saturation values result in more muted or desaturated colours. A saturation value of 0 results in a grayscale image, as all colours are desaturated.

Value: It determines how bright or dark a colour appears. A higher value corresponds to a lighter colour, while a lower value indicates a darker colour. Value is often used to control the overall brightness of an image or adjust the lightness of individual colours.

The HSV colour model provides a more intuitive and perceptually relevant representation of colour compared to the RGB model. By manipulating the hue, saturation, and value components, various colour adjustments and transformations can be performed in image processing to achieve desired visual effects and enhancements [54].

1.4.Deep Learning

Deep learning, a subset of machine learning, aims to replicate the functionality of the human brain by mimicking its processes. The brain consists of neurons that communicate with each other to facilitate various functions in the body. Biological neurons have three main components: the cell body, dendrites, and an axon.

The cell body of a neuron contains the nucleus and other essential components for its functioning. Dendrites are branch-like structures that receive signals from other neurons. These signals, in the form of electrical impulses, travel through the dendrites and reach the cell body.

Once the signals are processed in the cell body, the resulting output is transmitted through the axon. The axon is a long, slender projection that carries the electrical signals away from the cell body and transmits them to other neurons or target cells.

By simulating the behaviour of neurons and their interconnectedness, deep learning algorithms attempt to learn and extract meaningful patterns and representations from data, similar to how the human brain processes information [39].

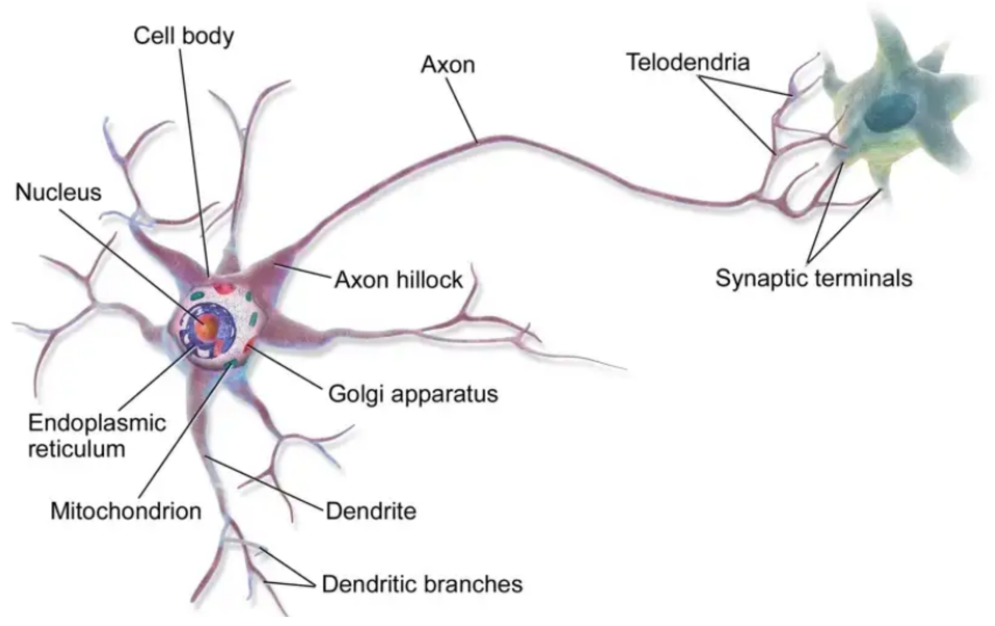


Figure 6.Biological Neuron [36].

Artificial neuron

Artificial neurons, also known as perceptrons or artificial neural units, share similarities with biological neurons and possess the following key features:

Incoming Links: Artificial neurons receive input signals from other neurons or external sources through incoming links. These links represent the connections between neurons in a neural network.

Weights: Each incoming link to an artificial neuron is associated with a weight. These weights determine the strength or importance of the input signal. The weights can be adjusted during the learning process to optimise the performance of the neural network.

Activation Function: An activation function is applied to the weighted sum of the input signals within the neuron. This function determines the output or activation of the neuron based on the input signals and their associated weights. The activation function introduces non-linearity to the neuron, enabling it to model complex relationships and make non-linear decisions.

By combining these features, artificial neurons are capable of processing and propagating signals through a neural network, enabling the network to learn and make predictions based on the input data. The weights and activation functions play crucial roles in shaping the behaviour and decision-making capabilities of artificial neural networks.

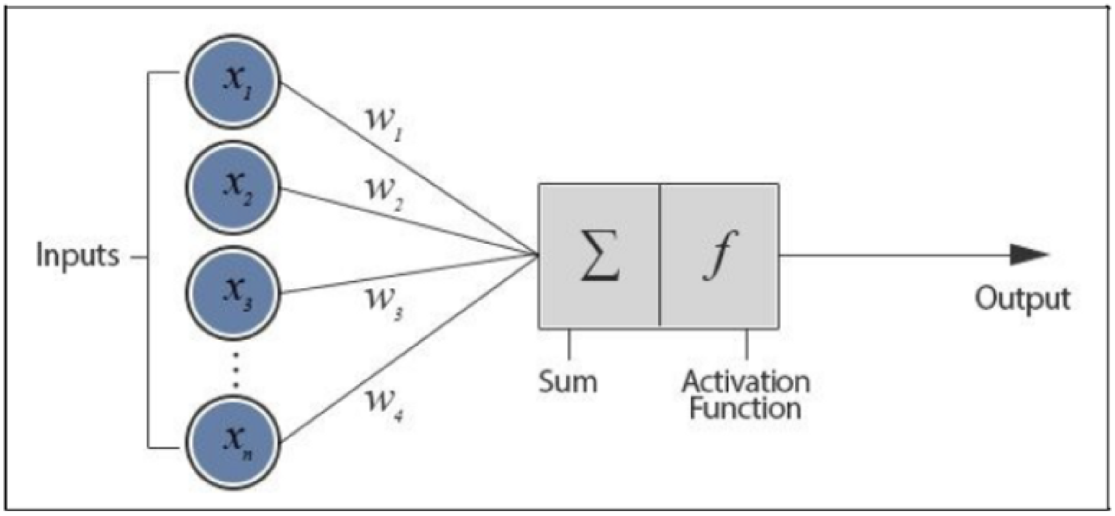


Figure 7.Representation of a neural network [35].

In the above figure each input is linked and each link has a particular weight associated with it [52].The output equation is given by the following formulae

Output = Activation function (Sum of (wi * xi) + bias)

FeedForward networks

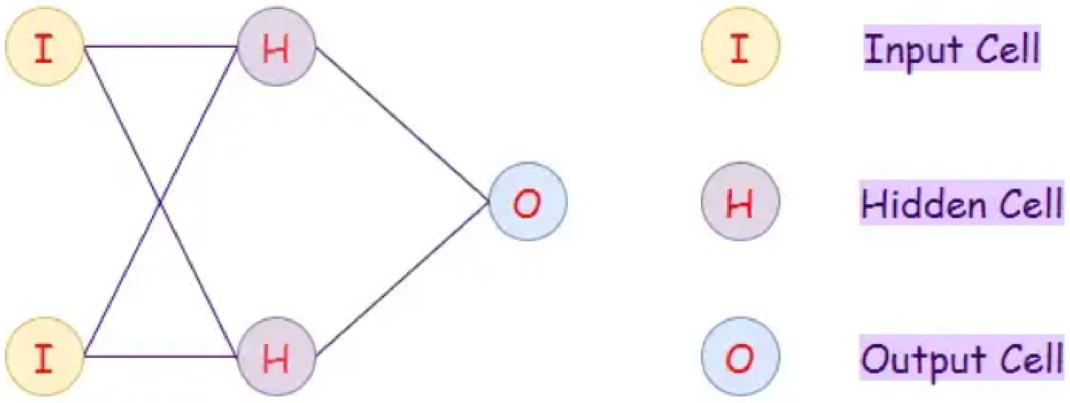


Figure 8.Feed Forward networks [34].

Feedforward neural networks typically consist of three main layers: the input layer, hidden layer(s), and output layer. Let us see the use of each layer:

Input Layer: The input layer is the starting point of the neural network. It receives raw data or features as input and passes this information forward to the hidden layers. Each input node in the input layer represents a specific feature or input variable.

Hidden Layer(s): Hidden layers are located between the input layer and the output layer. These layers perform computations on the input data using weights associated with the connections between neurons. The hidden layers apply mathematical operations and transformations to the input data, calculating the weighted sums of the inputs and passing them through activation functions. The number of hidden layers and the number of neurons in each hidden layer can vary depending on the complexity of the problem being solved.

Output Layer: The output layer is responsible for producing the final output or prediction of the neural network. It takes the processed information from the hidden layers and generates the desired output format. The number of nodes in the output layer corresponds to the number of output variables or classes in the problem being addressed.

Through the feedforward mechanism, data flows from the input layer through the hidden layers and finally reaches the output layer, producing the desired result. Each layer plays a crucial role in processing the information and contributing to the overall computation performed by the neural network [52].

Backpropagation

After initialising the weights in a neural network, which are typically random values, the model may not initially fit the data optimally, resulting in some error. To correct this error and improve the model's performance, backpropagation algorithms are commonly used. Here are the main steps of the backpropagation process:

Weight Initialization: Initially, the weights of the neurons in the network are randomly initialised. These weights determine the strength of connections between neurons and play a crucial role in the learning process.

Forward Propagation: The input data is fed forward through the neural network. The calculations are performed layer by layer, starting from the input layer, passing through the hidden layers, and finally reaching the output layer. Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.

Error Calculation: After obtaining the output of the neural network, the predicted values are compared with the actual values from the training data. The actual value and predicted value gives a difference which is how error is calculated.

Backward Propagation: Starting from the output layer the error is propagated backwards. By using a technique called chain rule, the error is distributed back to the previous layers. This process calculates how much each weight contributes to the overall error.

Weight Update: The calculated error is used to update the weights in the network. The weights are adjusted based on the gradient of the error with respect to the weights. This step aims to minimise the error and improve the model's performance. Various optimization algorithms, such as gradient descent, are commonly used to update the weights.

Iterative Process: Steps 2 to 5 are repeated iteratively for multiple epochs or until the error is minimised to an acceptable level. Each iteration helps the network learn and adjust its weights to better fit the training data.

By iteratively adjusting the weights based on the error, the backpropagation algorithm allows the neural network to learn and improve its performance over time, making it more accurate in making predictions or producing desired outputs [52].

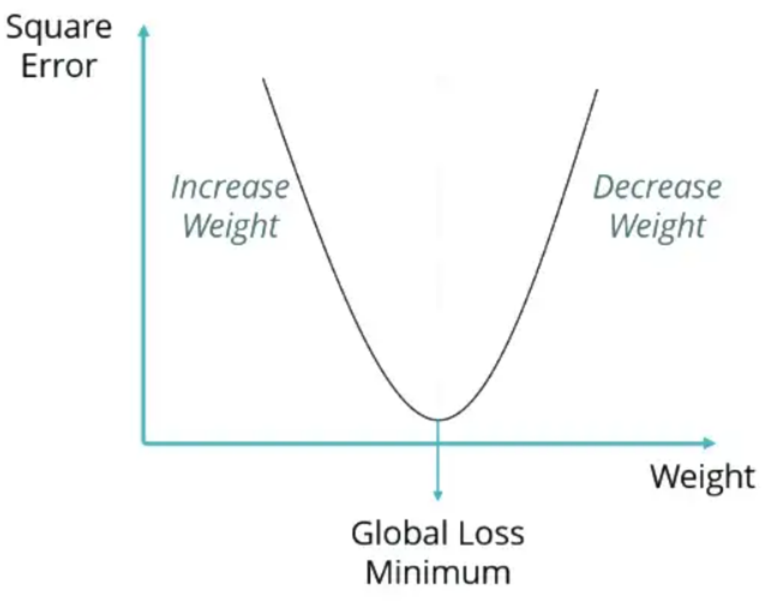


Figure 9.Finding the global minimum [39].

The above figure we try to find the global minimum by the backpropagation algorithms.

Now we know that deep learning algorithms are an integral part of machine learning which can be used to solve various challenges in these fields.

1.5.Object detection

In computer vision, the extraction of important features from an image is crucial for various tasks, including object detection, text recognition, and signature verification. Object detection specifically involves recognizing shapes and determining the location of objects within an image [12]. This field plays a fundamental role in computer vision, as it provides semantic information about images and videos, enabling applications such as face recognition, self-driving cars, and advancements in medical healthcare.

However, object detection poses several challenges. Variations in illumination, different viewpoints, and poses make it difficult to accurately detect objects in diverse scenarios. Researchers are actively working on developing optimal solutions to address these challenges [13]. The ultimate goal is to improve the accuracy and robustness of object detection algorithms, enabling them to handle real-world complexities and achieve reliable results in various applications.

The main goal of object detection is to detect what type that object the class it belongs to as well as the location of the object in the localised image area.There are three main stages in which object detection is divided into -

Informative region selection

Detecting objects in an image poses challenges due to their potential presence anywhere in the image and the possibility of various orientations, resulting in different aspect ratios. The conventional approach to object detection involves scanning the entire image using boxes of different aspect ratios. However, this brute force approach can be computationally expensive.

Using fewer boxes may lead to the risk of missing some objects, as the limited coverage might not capture all potential locations and orientations of objects in the image [55]. Balancing the number of boxes used for detection is crucial to strike a balance between computational efficiency and ensuring comprehensive object coverage.

Feature extraction

To detect and recognize objects in an image, it is essential to extract relevant visual attributes and features. Two commonly used methods for feature extraction are Scale-Invariant Feature Transform (SIFT) [14] and Haar-like features [15].Despite the effectiveness of these feature extraction techniques, challenges still exist when dealing with variations in illumination and object orientation. Illumination changes can affect the appearance of objects, making it difficult to create optimal algorithms that accurately describe their rich features. Similarly, object orientation variations introduce additional complexities in feature extraction.

Classification

After detecting an object in an image, the next step is to determine the class or category to which the object belongs. This involves creating a classifier that can compare and predict the features of the object to assign it to a specific class [55].

The process of object detection typically involves two main approaches: classification-based and regression-based. In classification-based object detection, the goal is to classify the detected object into predefined classes or categories. This is achieved by training a classifier on labelled data, where the classifier learns to associate specific features or patterns with each class. Once the object is detected, the classifier analyses its features and assigns it to the most suitable class based on its learned knowledge.

On the other hand, regression-based object detection focuses on predicting the coordinates or bounding box parameters that enclose the detected object. Instead of classifying the object, the model directly regresses the spatial coordinates of the object within the image. This approach is commonly used in tasks such as object localization or object tracking.

2.Convolutional neural networks

2.1.What are convolutional neural networks?

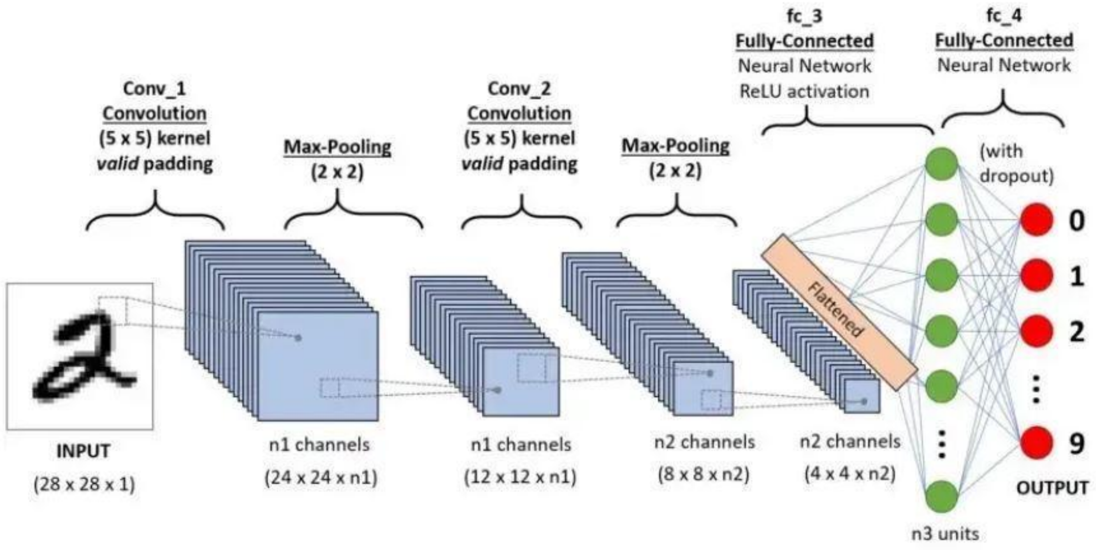


Figure 10.Convolution neural network sequence [16].

A convolutional neural network (CNN) is a deep learning algorithm designed specifically for image processing tasks. It operates by assigning weights to different parts of an input image, allowing it to differentiate between various aspects and features of the image. By leveraging these learned weights, a CNN can perform tasks such as image classification, object detection, and image segmentation.

The pre-processing step in a CNN involves resizing and resampling the input image, which helps ensure that the image is in a suitable format for further analysis. This step does not significantly impact the computational complexity of the network, as it primarily involves adjusting the dimensions of the image.

In Figure 10, we can observe the sequence of operations performed by a convolutional neural network, which typically includes convolution, pooling, and fully connected layers. Convolutional layers apply filters to the input image, capturing local patterns and features. Pooling layers reduce the spatial dimensions of the feature maps, helping to extract the most relevant information. Finally, fully connected layers perform classification or regression tasks based on the extracted features.

The architecture of a convolutional neural network is inspired by the pattern matching capabilities of neurons in the human brain. By emulating the structure and function of biological neurons, CNNs can effectively learn and recognize complex patterns and features in images, making them well-suited for a wide range of computer vision tasks [51].

2.2.Advantages of CNN

Indeed, representing an image as a flattened vector and feeding it into a multi-layer perceptron (MLP) is not effective in capturing the spatial relationships between pixels. MLPs are not designed to handle the inherent structure and locality of image data.

Convolutional neural networks (CNNs) address this limitation by leveraging the advantages of weighted kernels and pooling operations. The use of convolutional layers in CNNs allows for the extraction of local features by convolving a set of learnable filters (kernels) over the image. These filters detect patterns and edges at different spatial locations, capturing the interdependence of pixels and their relationships.

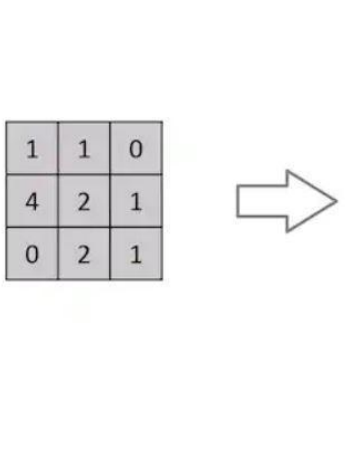


Figure 11.Flattening of 3x3 to 9x1 [16].

To define and asset the temporal and spatial dependencies in an image a convolutional neural network captures it better. We basically reduce the redundant parameters and extract the important features in an image [51].

2.3.Kernel

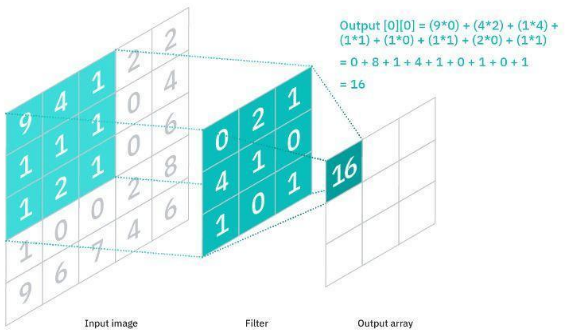

Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
16
Input image
Filter
Output array

Figure 12.Filter working of CNN [42].

In convolutional neural networks (CNNs), a kernel or filter is used to perform convolution operation on the input image. The kernel is typically a small matrix of weights, such as a 3x3 filter as mentioned in your example.

The convolution operation involves sliding the kernel over the entire image and computing a weighted sum of the pixel values within the receptive field covered by the kernel. This computation is performed element-wise, multiplying the corresponding kernel weights with the corresponding pixel values and summing them up. This process is repeated for each position of the kernel as it slides across the image.

By using the kernel to convolve with the image, the CNN extracts local features and spatial patterns. The output of this convolution operation is a feature map, where each element represents the result of convolving the kernel with a local region of the input image.

The dimension of the output feature map depends on the size of the input image, the size of the kernel, and the stride used during the convolution operation. In the example above, using a 3x3 filter, the dimension of the output feature map will be reduced compared to the input image.

The use of kernels in CNNs allows for localised feature extraction, enabling the network to learn and capture important patterns and structures in the image. The weighted averaging of the kernel values with the input image pixels helps to highlight relevant features and reduce the dimensionality of the output, which can help in simplifying and accelerating subsequent computations.

Overall, the convolutional operation with kernels is a fundamental component of CNNs and plays a crucial role in extracting meaningful features from the input image [51].

2.3.Striding and padding

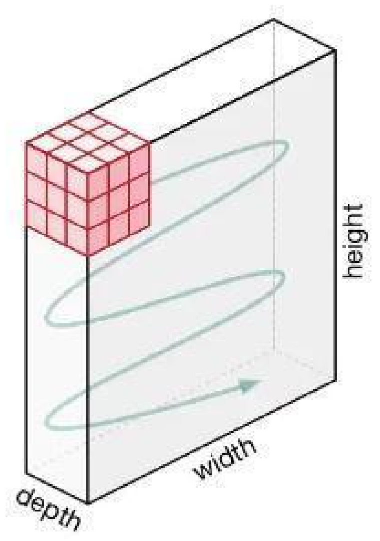


Figure 13.Movement of kernel [16].

Striding refers to the step size or the number of pixels the kernel moves horizontally and vertically as it slides over the input image. A stride of 1 means the kernel moves one pixel at a time, scanning the image column by column. A higher stride value, such as 2, would cause the kernel to skip pixels and move faster, resulting in a lower dimension of the output feature map. Striding affects the spatial resolution of the output feature map, as it determines how much overlapping or skipping occurs during the convolution operation [16].

Padding, on the other hand, involves adding extra border pixels to the input image. It is often used to control the spatial dimensions of the output feature map. Padding can be applied to ensure that the output feature map has the same spatial dimensions as the input image, or it can be used to increase the spatial dimensions. Padding with zeros (zero-padding) is a common approach, where zeros are added around the borders of the image. This helps in preserving the spatial information at the boundaries and preventing a reduction in dimensionality during convolution [16].

2.4.Pooling

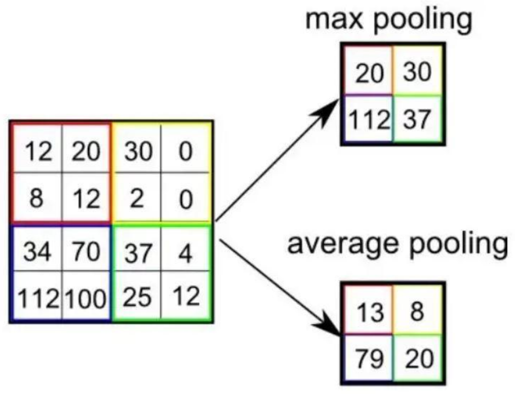


Figure 14.Types of pooling [16].

Max pooling and average pooling are the two commonly used types of pooling operations.

In max pooling, a pooling filter (usually with a small size, such as 2x2) slides over the input feature map, and for each location, it selects the maximum value within the filter window. The selected maximum value is then assigned to the corresponding position in the output feature map. Max pooling helps in capturing the most prominent or dominant features within the filter window and discarding less important or less significant information. It is particularly useful for achieving translation invariance and reducing the impact of small spatial translations in the input [16].

On the other hand, average pooling computes the average value within the filter window and assigns it to the corresponding position in the output feature map. Average pooling can help in reducing the impact of noise in the input and providing a smoother representation of the features. However, it may not preserve detailed information as effectively as max pooling [16].

Both max pooling and average pooling are used to downsample the feature maps, reducing their spatial dimensions. By reducing the spatial dimensions, pooling helps in reducing the computational complexity of the network and extracting higher-level, more abstract features.

Overall, max pooling is commonly preferred in convolutional neural networks due to its ability to capture dominant features and its effectiveness in reducing noise. However, the choice between max pooling and average pooling can depend on the specific task and the characteristics of the data being processed [16].

2.5.Applications of CNN

As artificial intelligence is the future , convolution neural network demand is getting higher.Big companies such as Tesla,NASA are continuously researching in the field.Some major applications of CNN are -

Face recognition

Here, we recognise and store rich features on the face, including the eyes, nose, mouth, and ears.These features are then compared against the database's features to determine whether they match [49].

Document Analysis

Text recognition from digital images is basically what happens when text is turned to an image.This means that we can use it for a wide range of activities, including signature verification, handwriting comparison, and file verification [49].

Climate analysis

Understanding climate is a difficult task due to the presence of global warming and heating issues.By feeding the image to a network,we can find patterns to why this is happening and what is the relation between them [49].

Object detection

Due to global warming and other heating-related problems, understanding the climate is a challenging undertaking.We can discover patterns to explain why this is happening and what the relationship between them is by feeding the image to a network [49].

Collecting Historic elements

We can discover the connections between each of them by gathering natural historical elements such as evolution, natural biodiversity, and habitat loss.primarily how that particular biodiversity led these individuals to that solution [50].

Understanding Grey areas

Similar to machine learning, probability can do better than humans in areas where they determine the final product.In cases involving fuzzy reasoning, convolutional neural networks can produce better results [50].

The convolutional neural network uses these properties, which are present, to reduce the size of the image and extract rich features from it.As a result, less computing power will be needed, which will increase the accuracy of object prediction in photos.There are currently primarily three forms of object detection that are in use.Now, we'll describe the various types and demonstrate how they operate.

3.Types of object detection pipelines

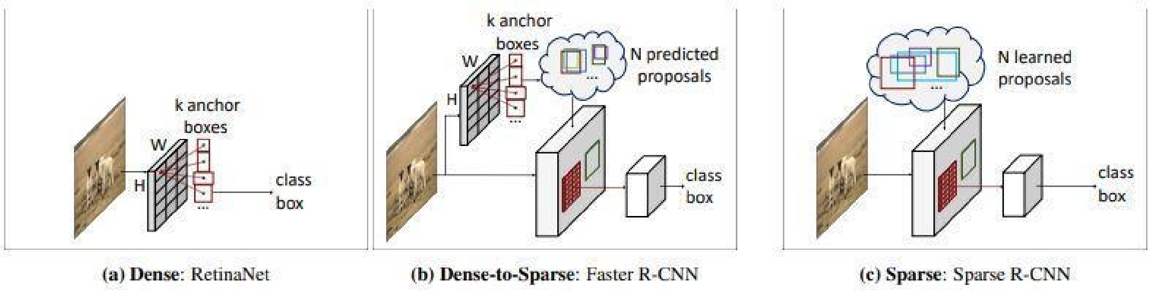


Figure 15.Types of object detection pipelines [1].

Object detection is a challenging task that involves locating and categorising objects within an image and determining their respective classes. There are several pipelines or methods used in object detection, and Figure 6 provides an overview of some of these methods.

To create an optimal algorithm instead of scanning the whole image with various aspect ratios to search a particular object is a huge task.Let us see what types of object detection pipelines are currently present.We can see the various working in figure 6 of the different types of method present in object detection.

3.1.Dense method in object detection

The sliding window algorithm is commonly employed in object detection pipelines to detect objects. However, the performance of this method has reached a plateau despite utilising deep convolutional neural networks to extract rich image features [17]. In the dense method, one-stage detectors such as YOLO, RetinaNet, and SSD are used. These detectors employ anchor boxes of various sizes and slide them across the image grid to capture objects with different orientations, sizes, and aspect ratios [18] [19] [20]. Recent advancements have introduced anchorless algorithms that select specific anchor boxes in restricted region areas, reducing computation requirements.

To evaluate object detection accuracy, dense methods use intersection over union (IoU) to measure the overlap between predicted and ground truth bounding boxes. Additionally, non-maximum suppression techniques [21] are applied to remove redundant predictions and optimise computation.

Despite these advancements, object detection remains challenging due to factors such as object appearance variation, viewpoint changes, occlusion, and environmental conditions. Researchers continue to explore new techniques and algorithms to enhance the accuracy, efficiency, and robustness of object detection systems.

3.1.1.Single stage pipeline

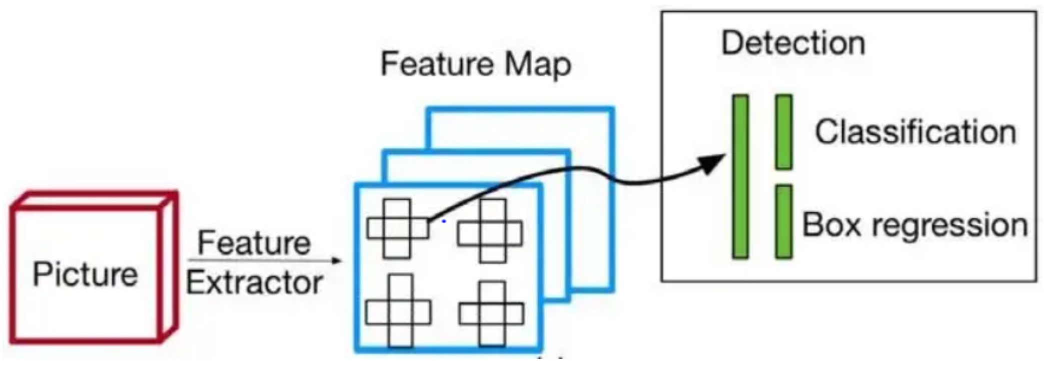


Figure 16.Single stage object detector [31].

Dense object detection employs a single-stage pipeline that directly extracts features from the entire image. This approach does not involve region of interest (ROI) algorithms to determine potential object areas. Instead, anchor boxes of various aspect ratios are applied across the image to capture rich features [31]. As a result, multiple anchor boxes may overlap with each other after object detection.

One-stage object detectors like YOLO and SSD are notable examples of dense object detection methods [18] [19]. These detectors use anchor boxes and employ a unified framework to simultaneously predict object classes and their bounding box coordinates. By leveraging deep convolutional neural networks, these models achieve real-time performance and high accuracy in object detection tasks.

3.1.2.YOLO

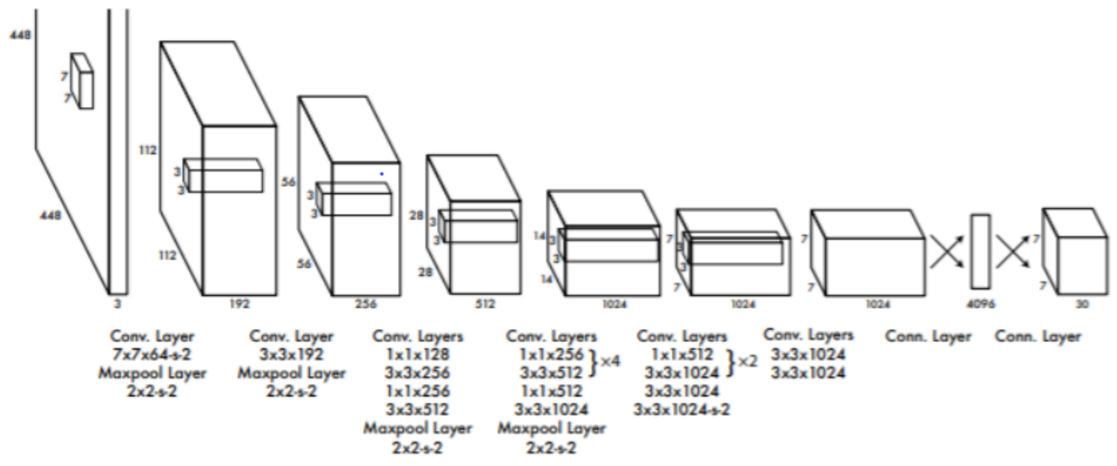


Figure 17.YOLO Architecture [31].

In YOLO, a single-stage pipeline is employed for object detection. The input image is processed by a neural network that divides it into multiple regions, each assigned with a bounding box. These bounding boxes are weighted to provide a confidence score indicating the presence of an object within the box [31].

The YOLO system consists of 24 convolutional layers and two fully connected layers, each serving a specific purpose [31].

-The first 20 layers are convolutional layers, followed by a pooling layer and a fully connected layer.

-The network is pre trained using images with a resolution of 224 x 224 x 3.

-For object detection, there are four additional convolutional layers and two fully connected layers.

-The final layer is responsible for generating bounding boxes and confidence scores.

-The activation function used in YOLO is ReLU (Rectified Linear Unit). It introduces non-linearity into the network and helps improve its ability to learn and extract features from the input data.

3.2.Dense to sparse method in object detection

In two-stage object detection pipelines, a different approach is taken compared to the single-stage pipelines. These pipelines have gained prominence in the field of object detection. They build upon the dense method by introducing a two-step process:

Region Proposal: In this step, the algorithm generates region proposals, which are areas in the image where objects are likely to be present. These proposals serve as potential regions of interest for further analysis. To refine the proposals and remove redundant ones, non-maximum suppression is applied. Examples of region proposal algorithms include Selective Search [22] and Region Proposal Networks [23].

Object Detection: After obtaining the region proposals, a second stage is employed to classify and localise objects within these proposals. This stage involves further processing and analysis to identify the objects accurately. The output is the detection of objects with their corresponding bounding boxes and class labels.

One notable advancement in two-stage object detection is the DETR (Detection Transformer) approach, which is capable of detecting objects without relying on manually created parameters. It achieves impressive results and represents a significant development within the dense-to-sparse method.

3.1.1.Two stage pipeline

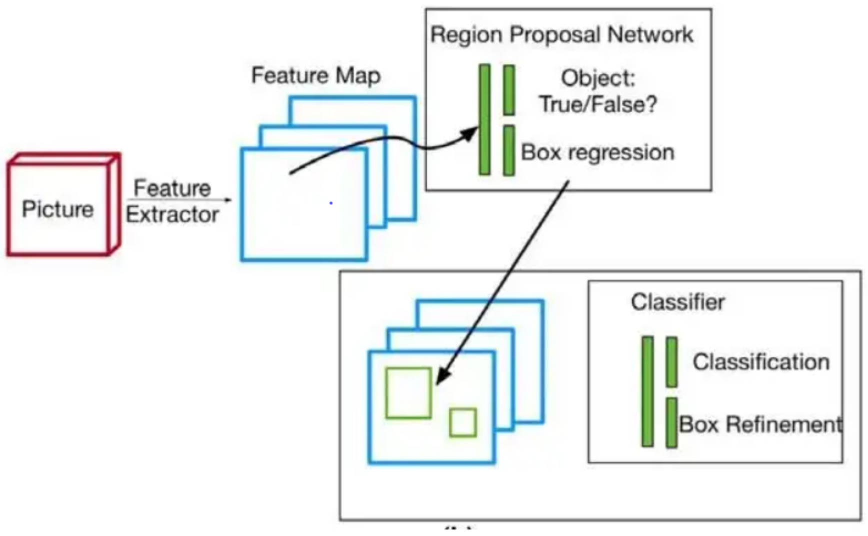


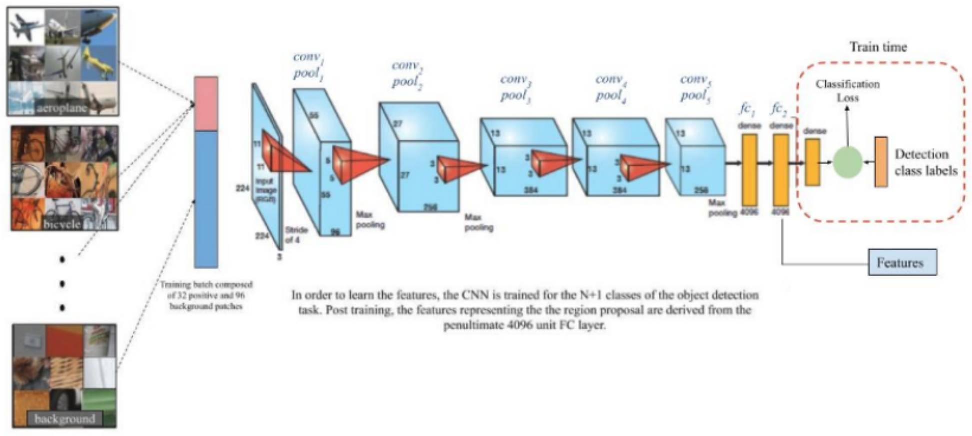
Figure 18.Two stage pipeline [31].

In the dense-to-sparse object detection using a two-stage pipeline, the focus is on obtaining regions of interest (ROIs) in the initial stage. These ROIs are specific areas within the image where objects are expected to be located. The goal is to reduce the computational load by narrowing down the search space.

Once the ROIs are identified, the second stage of the pipeline comes into play. This stage involves processing the ROIs and applying anchor boxes of various aspect ratios to detect and classify objects within these regions. Non-maximum suppression is then utilised to eliminate redundant object detections and refine the final results.

An example of a two-stage pipeline for object detection is Fast R-CNN [22]. It has gained popularity for its ability to efficiently detect objects by combining region proposal and classification stages, providing accurate results with improved computational efficiency.

3.1.2.Region CNN



In order to learn the features, the CNN is trained for the N+1 classes of the object detection task. Post training, the features representing the the region proposal are derived from the penultimate 4096 unit FC layer.

Figure 19.R-CNN architecture [41].

The object detection system employs a two-stage pipeline. In the first stage, the system identifies and localises regions of interest using various techniques. In the subsequent stage, object detection is performed within these regions. The network is trained using stochastic gradient descent (SGD) with 32 sample windows during pre-training. The final output includes approximately 2000 region proposals for object classification. Support Vector Machines (SVM) are utilised for classifying the detected objects into specific categories. [33]

3.3.Sparse method in object detection

Although the dense method techniques are removed by the sparse method, the accuracy of these approaches is typically lower than that of the dense to sparse or dense methods. This kind of technique is used by G-CNN [25], which updates boxes in a grid iteratively to forecast the objects.But it hasn't succeeded in getting very accurate. Sparse R-CNN [1] finds a solution to that issue and outperforms its competitors in terms of accuracy.

We will now examine the Sparse R-CNN's working approach, as well as its components, technology, and architecture.

4.Methodology

The primary concept behind Sparse R-CNN is the elimination of numerous object detection-related parameters.We use a small number of proposal boxes rather than enormous quantities of anchor boxes or region proposal boxes.Let's examine the Sparse R-CNN pipeline's architecture.

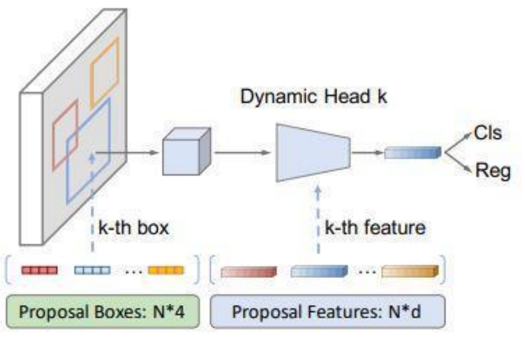


Figure 20.Sparse R-CNN pipeline [1].

The pipeline of Sparse R-CNN [1] comprises three essential components: the backbone network, the dynamic head, and the final output.

The backbone network is responsible for extracting image features, which are vital for object detection. It processes the input image to obtain high-level representations using convolutional operations and other relevant techniques.

The dynamic head interacts with the extracted features and performs continuous refinement and adaptation. It integrates information from different feature levels and dynamically adjusts the network parameters to improve detection accuracy.

The final output of the pipeline includes both classification and regression information. The classification output identifies the object classes present in the image, while the regression output provides refined bounding box coordinates. The input to the pipeline consists of the image itself, as well as proposal features and proposal boxes that can be trained and updated to enhance the detection performance [1].

4.1.Backbone of the network

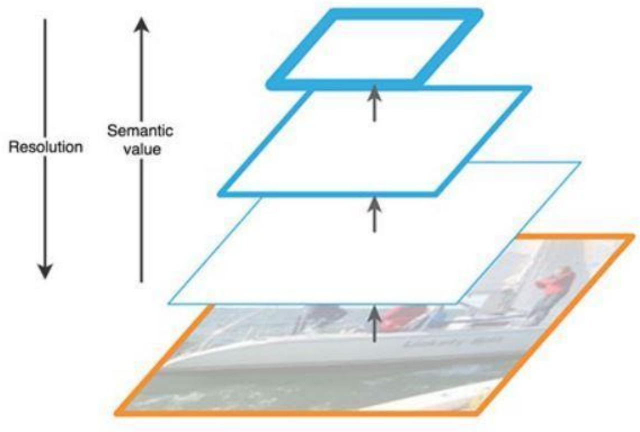


Figure 21.Feature extraction in FPN [45].

Sparse R-CNN leverages a feature pyramid network (FPN) based on the ResNet architecture [26] to generate multi-scale feature maps of the input images. The FPN operates by resizing the image into different resolutions and extracting rich features at each scale. These multi-scale feature maps are then utilised in the pipeline for object detection, enabling the model to capture objects of varying sizes and scales.

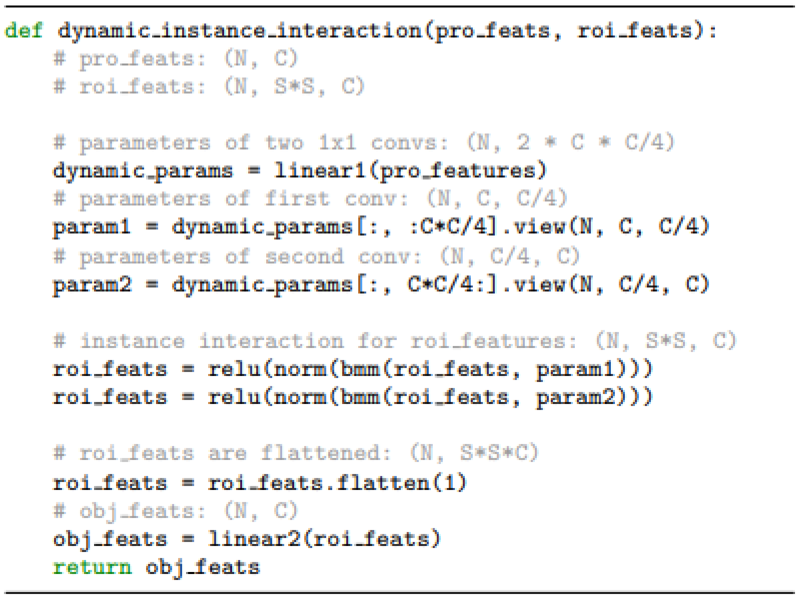
4.2.Learnable proposal box

In the Sparse R-CNN pipeline, a limited number of boxes are initialised as region proposals, unlike the region proposal networks that generate a large number of proposals. These boxes are represented as 4-dimensional vectors, including normalised centre coordinates, height, and width. Unlike region proposal networks, the values of these boxes are learnable and updated through the backpropagation algorithm during training [1]. This eliminates the need for manual parameter specification and allows the model to predict object locations in the image. Additionally, these boxes in Sparse R-CNN are designed to be less correlated with each other compared to region proposal networks.

4.3.Learnable proposal features

The proposal boxes in the Sparse R-CNN pipeline are 4-dimensional representations that capture certain values, but they may not fully capture important features such as the shape and orientation of the object. To address this limitation, higher-dimensional learnable proposal features are introduced. These features store richer information about the object and encompass all its characteristics, allowing for more comprehensive object representation and detection [1].

4.4.Dynamic instance interactive head



```
def dynamic_instance_interaction(pro_feats, roi_feats):
    # pro_feats: (N, C)
    # roi_feats: (N, S*S, C)

    # parameters of two 1x1 convs: (N, 2 * C * C/4)
    dynamic_params = linear1(pro_features)
    # parameters of first conv: (N, C, C/4)
    param1 = dynamic_params[:, :C*C/4].view(N, C, C/4)
    # parameters of second conv: (N, C/4, C)
    param2 = dynamic_params[:, C*C/4:].view(N, C/4, C)

    # instance interaction for roi_features: (N, S*S, C)
    roi_feats = relu(norm(bmm(roi_feats, param1)))
    roi_feats = relu(norm(bmm(roi_feats, param2)))

    # roi_feats are flattened: (N, S*S*C)
    roi_feats = roi_feats.flatten(1)
    # obj_feats: (N, C)
    obj_feats = linear2(roi_feats)
    return obj_feats
```

Figure 22.Pseudo code for dynamic head how the kth proposal feature makes dynamic parameters for the kth region of interest.Here BMM means batch matrix multiplication while linear is linear projection [1].

The Sparse R-CNN pipeline operates on N proposal boxes, where each box undergoes the RoIAlign operation to extract its corresponding feature. These proposal features are then utilised in the dynamic head for final output prediction. Each region of interest has its own head for processing. With N proposal boxes, there are N proposal features, and each proposal feature is further filtered using an RoI to eliminate irrelevant information and generate the final object feature.

The pipeline involves an iterative process where the proposal boxes generated in the first stage serve as inputs for the second stage. Additionally, a self-attention module is incorporated to capture the relationships between different objects and enhance the overall detection performance [1].

4.5.Set prediction loss

The loss function used in the Sparse R-CNN pipeline is defined as:

Loss = λ(Focal Loss Coefficient) · L(Focal Loss) + λ(L1 Loss Coefficient) · L(L1 Loss) + λ(Intersection over union Coefficient) · L(Intersection over union loss)

Here, each term in the loss function is defined as follows:

Focal Loss: It is a loss function introduced in the RetinaNet model [27] that addresses the problem of class imbalance in object detection. It assigns higher weights to hard examples (misclassified examples) and reduces the influence of easy examples.

L1 Loss: It is a regression loss that measures the absolute difference between predicted box coordinates and ground truth box coordinates. It is used to penalise the discrepancies between predicted and target box positions.

Intersection over Union (IoU) Loss: It measures the similarity between predicted and ground truth bounding boxes by calculating the intersection over union ratio. It penalises the differences in box overlap and helps in accurate localization.

Each term in the loss function is multiplied by a corresponding coefficient (λ) that determines its contribution to the overall loss. These coefficients allow adjusting the relative importance of each loss term during training based on the specific requirements of the detection task [1].

4.5.1.L1 loss

This loss is used for minimising or reducing the value between predicted and actual value [57].It is not affected by outliers also.

L1 = sum of | predicted - actual |

$$L1LossFunction = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

Figure 23.L1 loss equation [47].

4.5.2.Intersection over Union

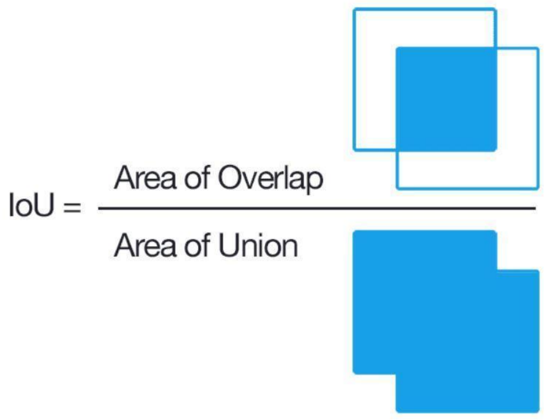


Figure 24.Intersection over union [48].

The loss function in object detection, as defined by the Intersection over Union (IoU), measures the overlap between the predicted and ground truth bounding boxes. By calculating the ratio of the intersection area to the union area, the IoU reflects the accuracy of object detection. A higher IoU value corresponds to a higher level of accuracy. In practical applications, the IoU is commonly employed as a threshold to eliminate overlapping or redundant bounding boxes in order to refine the object detection results [28].

4.5.3.Focal loss

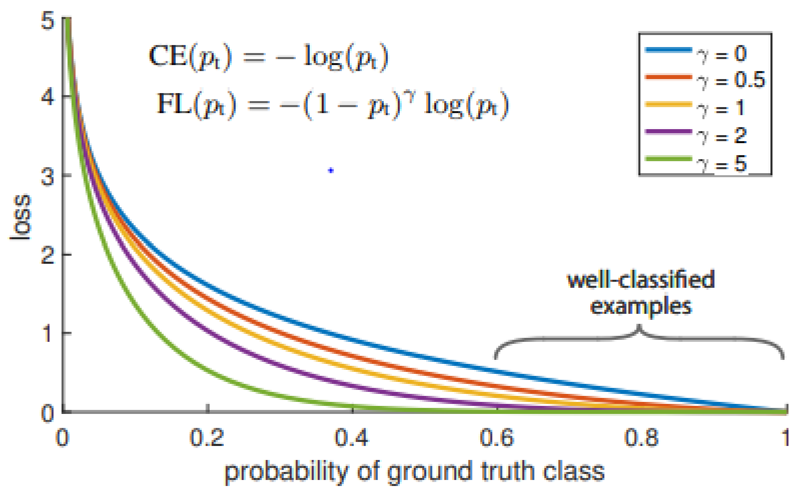


Figure 25.Comparison with loss and probability of ground truth class using focal loss [27].

Focal loss is an enhanced version of cross-entropy loss that addresses the issue of class imbalance in object detection. It assigns different weights to hard misclassified examples (e.g., noisy or blurred images) and easy misclassified examples (e.g., background) [27]. In cross-entropy loss, the imbalance between foreground and background objects can lead to suboptimal performance in single-stage object detectors.

The formula for cross-entropy loss is given as:

Cross entropy = -log(pt),

where p represents the probability of the class label, ranging from 0 to 1. However, this traditional approach fails to handle the class imbalance effectively [27]. Focal loss introduces a tunable focusing factor, gamma, and modifies the loss function as:

Focal loss = -(1-p)^gamma * log(pt).

The gamma value, greater than zero, adjusts the contribution of each example to the loss calculation [27]. Figure 7 illustrates the impact of different gamma values on the focal loss.

These are the key parameters and features of the Sparse R-CNN network. Now, let's explore the results achieved by this model.

5.Technologies Used

5.1.Python

It is a high-level programming language that currently rules the fields of artificial intelligence and machine learning. It is employed in numerous situations, including scripting, testing, and the programming of various applications.

5.2.Detectron2

The Facebook research team created Detectron2 [29], which focuses on employing object detection. It includes a number of libraries and techniques for keypoint identification, segmentation, and object detection.

5.3.OpenCV

A library called OpenCV [56] is used for a variety of picture representation, conversion, and recognition tasks.It provides a vast selection of image processing algorithms.

6.Average precision

In order to assess the model's performance in object detection, accuracy alone is not sufficient.The area under the precision and recall curves is used to calculate average precision.Let's first look at how to compute these values [30].

True positive (TP) - when a proper output is detected by the model and the actual output is positive.

True negative(TN) -when a valid output is detected by the model but the actual outcome is negative.

False negative(FN) - when the result that the model recognises as being wrong is actually positive.

False positive(FP) - when the result that the model recognises as being wrong is actually negative.

These four values define a confusion matrix [48].

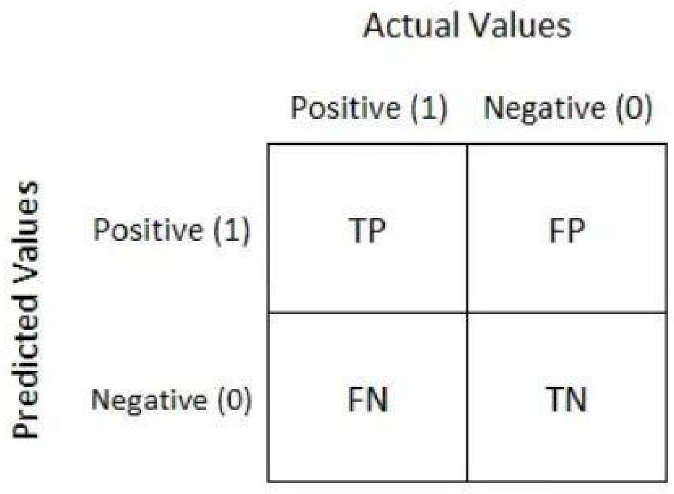


Figure 26.Confusion matrix [48].

Now precision and recall are calculated by the following formula.

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

After evaluating the precision and recall metrics, we can determine the performance of a model by examining the area under the curve (AUC). Precision measures the frequency of correct predictions made by the model, while recall assesses the model's ability to correctly identify positive instances.

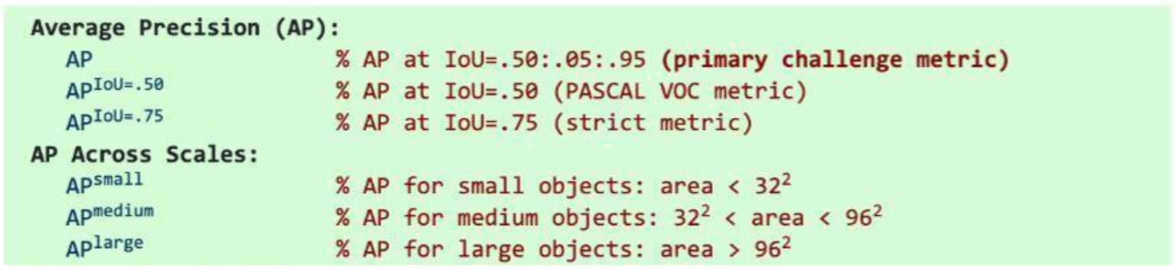
A higher precision with lower recall suggests that the model has a high level of accuracy in predicting positive outputs but may miss some instances. Conversely, a higher recall with lower precision indicates that the model has a greater ability to identify positive instances but may also generate more false positives.

By analysing the AUC, we can gain a comprehensive understanding of the model's overall performance, taking into account both precision and recall [30].

$$\mathrm{AP} = \int_0^1 p(r)dr$$

Figure 27.Average precision [48].

Thus to calculate the AP we need to find the area under the precision and recall curve.Now in the COCO dataset we use the following metrics for various APs.



Average Precision (AP):	
AP	% AP at IoU=.50:.05:.95 **(primary challenge metric)**
$AP^{IoU=.50}$	% AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$	% AP at IoU=.75 (strict metric)
AP Across Scales:	
AP^{small}	% AP for small objects: area < 32^2
AP^{medium}	% AP for medium objects: 32^2 < area < 96^2
AP^{large}	% AP for large objects: area > 96^2

Figure 28.COCO metrics [43].

The above COCO metrics are used for calculating various metrics which are used to compare different object detection models [43].

6.1.Improving average precision

To improve the average precision we need to improve the accuracy of our CNN model.Let us see the many ways on how we can improve the accuracy of the CNN model.

1.Tune parameters of the network

We can adjust a lot of factors, such as the learning rate, the number of epochs, or the optimisers in our algorithm.To determine the ideal number of epochs to employ for training our model, we must do numerous trials.

2. Image data augmentation

Because of the variety, we are aware that the more data we supply, the higher its accuracy.To produce more image samples, we can rotate and vary the direction of the image.

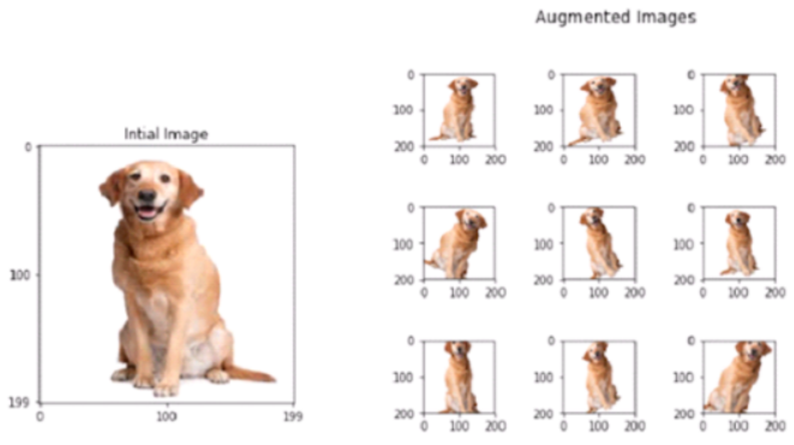


Figure 29.Augmentation of an image [44].

The accompanying illustration illustrates how we might enhance the image to broaden the size of our input sample.We may incorporate the rotated version of these images to boost accuracy because CNN is also rotation invariant.

3. Deeper network topology

Instead of expanding the network's width, we can increase its depth and increase accuracy.The depth of the network will combine all the crucial elements, whilst the width of the network determines what input we must provide to the network [44].However, excessive width and depth might lead to costly computation issues.Finding the proper depth is crucial, thus.

4. Underfitting and overfitting

Overfitting occurs when the model that was trained by the model is trained too well and provides too much lower accuracy for predicting unknown data [60].When tested on the training set of data, the model underfits poorly.

Underfitting can be fixed by either adding more layers to our network or fine-tuning the parameters to maximise accuracy.

For overfitting we do the following methods

-Train with more data
Larger the data set ,lesser the possibility of overfitting due to the variety of the data present in the input sample.

-Early stopping

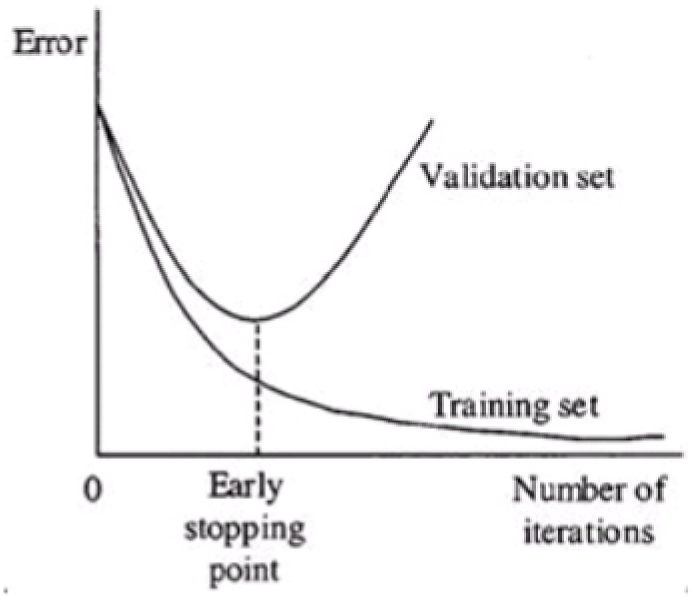


Figure 30.Comparison with number of iterations and error [44].

Numbers of iteration do improve the model,but after some time the model tries to overfit on the training dataset,therefore stopping before can help in removing overfitting.

-Cross validation

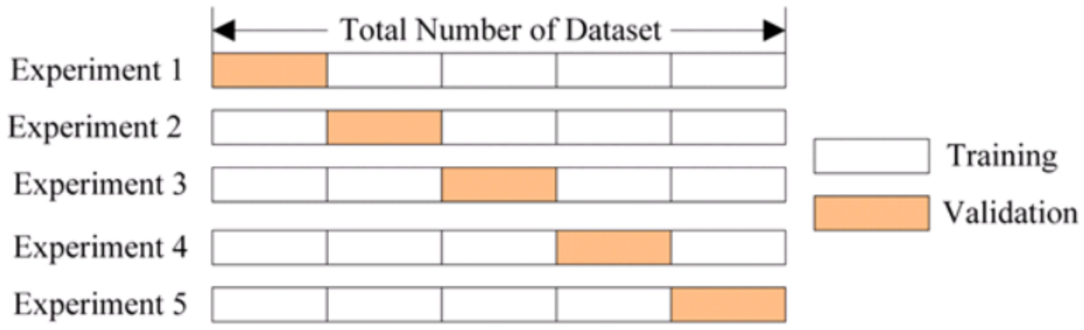


Figure 31.Cross validation [44].

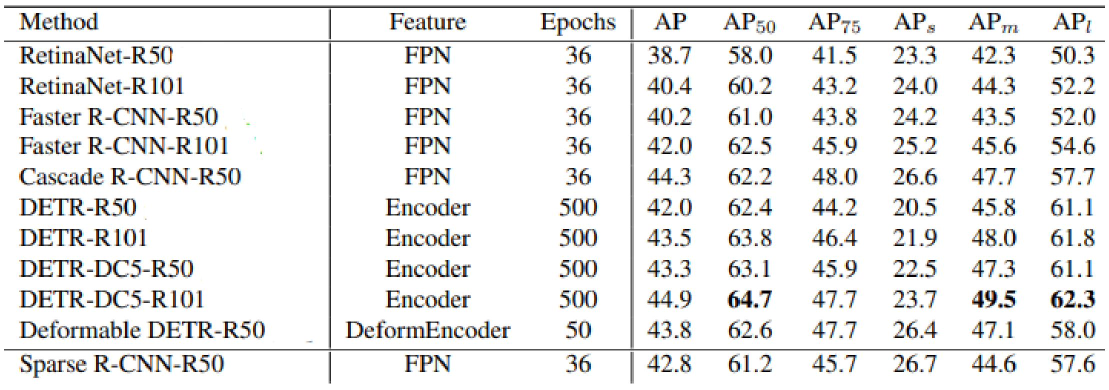
We divide our data set into k partitions and for each partition we do the training.Then we average out the accuracy for each iteration.In the k partitions k-1 are used for training while the remaining are used for validation [44].

7.Result

The average precision measurement in object detection determines the outcome. The outcome is displayed on the COCO dataset and contrasted with other datasets.Let's check the average precision first.

7.1.Comparison of various models

The comparison between various models was taken from their before output and papers [28] and we are currently comparing with the sparse R-CNN.



Method	Feature	Epochs	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
RetinaNet-R50	FPN	36	38.7	58.0	41.5	23.3	42.3	50.3
RetinaNet-R101	FPN	36	40.4	60.2	43.2	24.0	44.3	52.2
Faster R-CNN-R50	FPN	36	40.2	61.0	43.8	24.2	43.5	52.0
Faster R-CNN-R101	FPN	36	42.0	62.5	45.9	25.2	45.6	54.6
Cascade R-CNN-R50	FPN	36	44.3	62.2	48.0	26.6	47.7	57.7
DETR-R50	Encoder	500	42.0	62.4	44.2	20.5	45.8	61.1
DETR-R101	Encoder	500	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R50	Encoder	500	43.3	63.1	45.9	22.5	47.3	61.1
DETR-DC5-R101	Encoder	500	44.9	**64.7**	47.7	23.7	**49.5**	**62.3**
Deformable DETR-R50	DeformEncoder	50	43.8	62.6	47.7	26.4	47.1	58.0
Sparse R-CNN-R50	FPN	36	42.8	61.2	45.7	26.7	44.6	57.6

Figure 32.Comparison of sparse R-CNN with various detectors [1].

We can readily observe that Sparse R-CNN outperforms a number of its rivals.Its average accuracy is rather comparable to DETR.As a result, sparse R-CNN outperforms other detectors in the object detection field in terms of performance with less parameters. The highlighted portion shows which of them has the highest average precision.Small images are denoted by APs, whereas medium and big images are denoted by APm and APl, respectively. While the percentage of intersection over union is denoted by the AP50, AP75.

7.2.Output of custom images

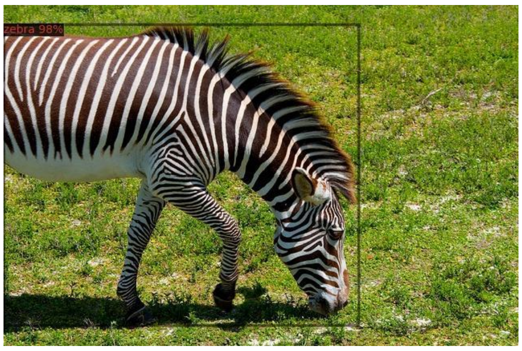


Figure 33.The model showing the category as well as the confidence level.

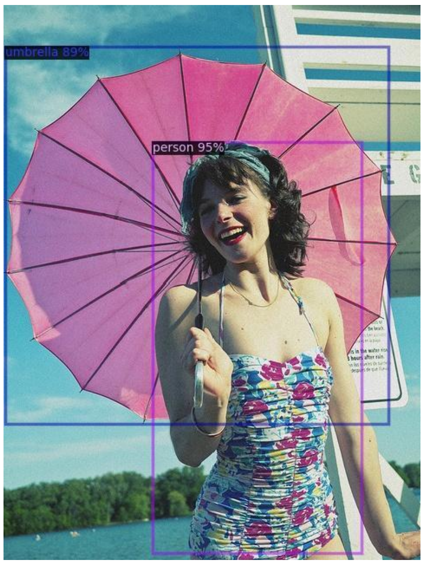


Figure 34.The model showing the category as well as the confidence level.

The output can be seen in the custom images with the confidence score as well.We can see it is able to categorise and show how accurate the result is.

7.3.Comparison with different number of proposal boxes

Proposals	AP	AP50	AP75
100	42.3	61.2	45.7
300	43.9	62.3	47.4

500	44.6	63.2	48.5

Table 1 Change in number of proposal boxes.

We can see that average precision increases with the number of proposal boxes but the training time as well as increases.

AP	AP50	AP75	APs	APm	APl
45.7	65.0	49.8	32.4	49.9	61.6

Table 2 Evaluation of model for 250 COCO images.

7.4.Conclusion

We may observe that the sparse R-CNN outperforms its competitors in terms of accuracy without requiring a sizable number of parameters. With a high degree of accuracy, it is possible to extract and detect objects by employing the proposal boxes as well as proposal features. Additionally, the non-maximum suppression in the photographs is not used. As a result, Sparse R-CNN can be applied to object detection and enhanced.

7.5.Future Work

Additionally, we can create additional algorithms based on the sparse R-CNN model after hyper-tuning the parameters to get an ideal method. Other object detectors can be combined and assembled to further raise the average precision. We can improve the model's average precision by adjusting the proposal box values as well as the network's depth.

8.References

[1] Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., & Luo, P. (2021). Sparse R-CNN: End-to-end object detection with learnable proposals. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[2] W. S. Mcculloch, and W. H. Pitts, “A logical Calculus of Ideas Immanent in Nervous Activity,” The Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, 1942.

[3] F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,” Psychological Review, pp. 368-408, 1958.

[4] C. V. D. Malsburg, “Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.” .

[5] Davd. Rumhar, Geoffrey. Hinton, and RonadJ. Wams, “Learning representations by back-propagating errors.”

[6] A. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano, and K. J. Lang, “Phoneme recognition using time-delay neural networks,” IEEE Transactions on Acoustics Speech & Signal Processing, vol. 37, no. 3, pp. 328-339, 1989.

[7] W. Zhang, “Shift-invariant pattern recognition neural network and its optical architecture,” in Proceedings of annual conference of the Japan Society of Applied Physics, 1988.

[8] Specht, and D.F., “A general regression neural network,” IEEE Transactions on Neural Networks, vol. 2, no. 6, pp. 568-576.

[9] K. Aihara, T. Takabe, and M. Toyoda, “Chaotic neural networks,” Physics Letters A, vol. 144, no. 6-7, pp. 333-340.

[10] B. L. Lecun Y , Bengio Y , et al., “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Advances in neural information processing systems, vol. 25, no. 2, 2012.

[12] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, p. 1627, 2010.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in CVPR, 2014.

[14] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. J. of Comput. Vision, vol. 60, no. 2, pp. 91–110, 2004.

[15] R. Lienhart and J. Maydt, “An extended set of haar-like features for rapid object detection,” in ICIP, 2002.

[16] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.Sumit Saha, 2018

[17] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005

[18] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In CVPR, 2017.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In ECCV, 2016.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In ICCV, 2017.

[21] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In NIPS, 2020.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NeurIPS, 2015

[23] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. IJCV, 104(2):154–171, 2013.

[24] Nicolas Carrion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-toEnd object detection with transformers. In ECCV, 2020.

[25] Mahyar Najibi, Mohammad Rastegari, and Larry S Davis. G-cnn: an iterative grid based object detector. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2369–2377, 2016.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In ICCV, 2017.

[28] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalised intersection over union: A metric and a loss for bounding box regression. In CVPR, 2019.

[29] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github. com/facebook research/detectron2, 2019.

[30] Beitzel, S.M., Jensen, E.C., Frieder, O. (2009). MAP. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_492

[31] Ndonhong, Vanessa & Bao, Anqi & Germain, Olivier. (2019). Wellbore Schematics to Structured Data Using Artificial Intelligence Tools. 10.4043/29490-MS.

[32] Redmon, Joseph & Divvala, Santosh & Girshick, Ross & Farhadi, Ali. (2016). You Only Look Once: Unified, Real-Time Object Detection. 779-788. 10.1109/CVPR.2016.91.

[33] Haripriya, P., & Porkodi, R. (2020). Deep Learning Pre-Trained Architecture Of Alex Net And Googlenet For DICOM Image Classification.

[34] Main Types of Neural Networks and its Applications — Tutorial. (2021). Retrieved 15 January 2021, from https://towardsai.net/p/machine-learning/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e

[35] Instance Segmentation of Point Clouds using Deep Learning, Gerardo Francisco Perez Layedra, UPC, (2021). Retrieved 15 January 2021, from https://upcommons.upc.edu/bitstream/handle/2117/117737/131440.pdf?sequence=1&isAllowed=y

[36] A biological neuron. | BruceBlaus, CC BY 3.0, via Wikimedia Commons

[37] Introduction to Image Processing — Part 1: from Fundamentals,https://perez-aids.medium.com/introduction-to-image-processing-part-1-fundamentals-579cc414cebe

[38]Image Processing with Fiji/ImageJ,Jan 1 ,from https://learning.rc.virginia.edu/notes/fiji-intro/

[39] What is Deep Learning? ,Saniya Parveez, Roberto Iriondo from https://pub.towardsai.net/what-is-deep-learning-34767bb10366

[40] 2D Image Digital Representation from https://edtech.engineering.utoronto.ca/object/2d-image-digital-representation

[41] Haripriya, Ponnapalli and R. Porkodi. “Deep Learning Pre-Trained Architecture Of Alex Net And Googlenet For DICOM Image Classification.” (2020).

[42] Convolutional Neural Networks By: IBM Cloud Education,20 October 2020

[43] Coco metrics from https://cocodataset.org/#detection-eval

[44] Improving Performance of Convolutional Neural Network,Dipti Pawar,Aug 14, 2018

[45] Object Detection with Deep Learning: A Review Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE

[46] Intersection over union ,https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[47] Loss Functions in Deep Learning,Sanober Ibrahim,2021,https://insideaiml.com/blog/LossFunctions-in-Deep-Learning-1025

[48]Padilla, Rafael & Netto, Sergio & da Silva, Eduardo. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. 10.1109/IWSSIP48289.2020.

[49]Wang, Xiaoming & Li, Jianping & Liu, Yifei. (2018). Application of Convolutional Neural Network (Cnn)in Microblog Text Classification. 127-130. 10.1109/ICCWAMTIP.2018.8632583.

[50]Lu, H. & Zhang, Q.. (2016). Applications of deep convolutional neural networks in computer vision. 31. 1-17. 10.16337/j.1004-9037.2016.01.001.

[51]Chauhan, Rahul & Ghanshala, Kamal & Joshi, R.. (2018). Convolutional Neural Network (CNN) for Image Detection and Recognition. 278-282. 10.1109/ICSCCC.2018.8703316.

[52]Grossi, Enzo & Buscema, Massimo. (2008). Introduction to artificial neural networks. European journal of gastroenterology & hepatology. 19. 1046-54. 10.1097/MEG.0b013e3282f198a0.

[53]Oleiwi, Zahraa. (2019). Digital Image processing: sampling and quantization (Bit resolution).

[54]Mohan, Vaka & Durga, B. & Devathi, Swathi & Raju, Srujan. (2016). Image Processing Representation Using Binary Image; Grayscale, Color Image, and Histogram. 10.1007/978-81-322-2526-3_37.

[55]Sharma, Kartik & Thakur, Nileshsingh. (2017). A review and an approach for object detection in images. International Journal of Computational Vision and Robotics. 7. 196. 10.1504/IJCVR.2017.081234.

[56]Mahamkali, Naveenkumar & Ayyasamy, Vadivel. (2015). OpenCV for Computer Vision Applications.

[57]Kopciak, Peter. (2015). Introduction to Machine Learning with Feed-Forward Artificial Neural Networks and Evolving with Genetic Algorithms. 10.13140/RG.2.1.2951.3449.

[58]Haslwanter, Thomas. (2016). Python. 10.1007/978-3-319-28316-6_2.

[59]Gilanie, Ghulam. (2022). Image Representation and Description.

[60]Zhu, Mu. (2004). Recall, precision and average precision.

PAPER NAME

Final report Sparse R-CNN Object Detecti on Using Proposal Boxes.docx

WORD COUNT	CHARACTER COUNT
10090 Words	**57688 Characters**
PAGE COUNT	FILE SIZE
55 Pages	**3.3MB**
SUBMISSION DATE	REPORT DATE
May 24, 2023 6:06 PM GMT+5:30	**May 24, 2023 6:08 PM GMT+5:30**

● 11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 4% Internet database
- 0% Publications database
- Crossref database
- Crossref Posted Content database
- 11% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 8 words)