Object Detection in Camouflage Environment using Deep Learning

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

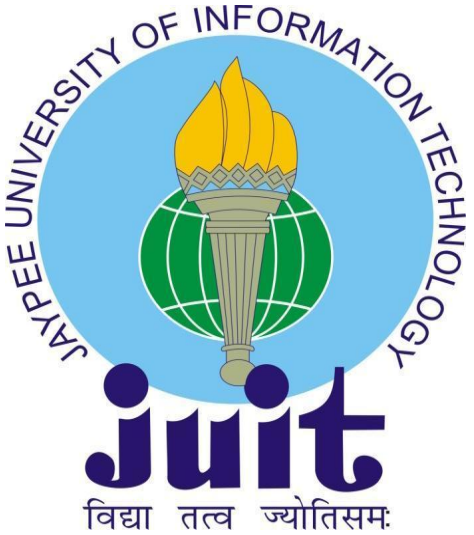
by

Akshat Tripathi [191392]

under the supervision of

Dr Vipul Kumar Sharma

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Certificate

Candidate's Declaration

We hereby declare that the work presented in this report entitled **Object Detection in Camouflage Environment using Deep Learning** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of **Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat** is an authentic record of our own work carried out over a period from **July 2022 to May 2023** under the supervision of **Dr. Vipul Kumar Sharma, Assistant Professor (Senior Grade)**.
We also authenticate that we have carried out the above-mentioned project work under the proficiency stream **Artificial Intelligence**.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Akshat Tripathi, 191392

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Vipul Kumar Sharma
Assistant Professor (Senior Grade)
Computer Science & Engineering and Information Technology

Dated:

Plagiarism Certificate

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 01-05-23

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: Akshat Tripathi Department: C.S.E Enrolment No 191392

Contact No. 6350673217 E-mail. akshattripathi215@gmail.com

Name of the Supervisor: Dr. Vipul Kumar Sharma

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): Object Detection in Camouflaged Enviornment using Deep Learning

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 66
- Total No. of Preliminary pages = 9
- Total No. of pages accommodate bibliography/references = 3

Akshat.T

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 9% (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

Vipul Sharma 01/05/2023

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Acknowledgement

Firstly, We express our heartiest thanks and gratefulness to almighty God for His divine blessing making it possible to complete the project work successfully.

We are really grateful and wish our profound indebtedness to Supervisor **Dr. Vipul Kumar Sharma, Assistant Professor (Senior Grade)**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Solan Deep knowledge & keen interest of my supervisor in the field of "**Deep Learning**" helped us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

Akshat Tripathi
[191392]

Table Of Contents

Table Of Figures

Table Of Graphs

Abstract

This project aims to accomplish a task popularly known as Camouflaged Object Detection also abbreviated as COD, in which we aim to identify objects that are present into similar surroundings. Due to native similarities between the target object and background make, Camouflaged Object Detection is far more difficult to accomplish than basic object detection.

The Project is based on the novel Camouflaged Object Detection framework. With the use of a framework, we build a model which will use a method that will accurately detect camouflaged objects by considering full-level information and a large receptive field. The experiments on a COD-Dataset show that our model achieves the desired performance we aim for.

Applications for moving object identification and tracking include surveillance, anomaly detection, vehicle navigation, etc. The body of knowledge on object identification and tracking is sufficiently extensive, and numerous important survey publications are available. However, because of its complexity, research on camouflage object identification and tracking is restricted. Existing research on this issue has relied either on biological traits of the hidden items or computer vision methods. The detection and tracking of camouflaged objects using computer vision algorithms is discussed in this article from a theoretical standpoint. This article also discusses a number of pertinent issues and the direction that future research in this area should take.

CHAPTER 1
INTRODUCTION

1.1 Introduction

The deep learning discipline is becoming more well-known and widespread as a result of the advancement in computers and the availability of GPUs (Graphics Processing Units) for mathematical computation. Deep learning object identification, a component of image processing, is crucial for computer vision and automated driving. Localization and categorization of objects are both included in object detection. In order to locate an item, a computer must first scan the image and provide the precise coordinates.

Computerised categorization of objects into several categories is called object classification. Fast/Faster R-CNN is the source of the conventional image object detection pipeline concept. The included object regions are generated by the region proposal network and entered into the classifier. The localisation of the item comes first, and its categorization comes afterwards. This pipeline function has an inefficient time cost.

The You Only Look Once (YOLO) network was created with the intention of solving this issue. With a real-time image processing speed of 45 frames per second for network prediction, YOLO is a single neural network end-to-end pipeline. The convolution neural networks, including the most advanced convolutional neural networks in recent years, are introduced in this thesis. The steps for implementing YOLO are described in detail. Since the YOLO network has a faster convergence rate during training, offers high accuracy, and has an end-to-end architecture, which makes networks simple to optimise and train, we use it for our applications.

1.2 Problem Statement

Create a model to recognise the image used as our data's input, and after iterating over the localization and classification issue, we find ourselves needing to identify and map items that belong to a class among many other classes. Finding and identifying a variety of things on an image is referred to as object detection.

With issues like categorization, the output of object detection is variable in length since the numerous sorts of objects identified may differ from picture to image. The operative word here is "varying."

Since we must take into account a variety of item appearances while constructing a model for it, such as the object's size and form, precise camouflaged object recognition is difficult. The model should be able to identify photos of various types involving characteristics like target size, colour contrast, and identification of effects like cropping, resizing, reshaping, etc. because there are several image data types involved in the object recognition process. There are several possible uses for DE camouflaging, including the requirement to differentiate between original and duplicate products during manufacturing, military applications, and autonomous systems where the ability to recognise objects in camouflaged pictures is crucial.

1.3 Objectives

We are attempting to build a model based on deep learning technology which will identify the object to identify and locate objects within an image which are difficult to perform even with the naked eye. With the help of our model, we can use COD in various fields like applying object detection techniques in biological, security, or military scenarios. It can be of supportive nature in the field of computer vision and extend the capability of Technologies like deep learning to produce our desired objectives.

1.4 Methodology

To make sure that each thing has distinctive traits is the main driving force behind this strategy. It may be easier to separate objects when they have certain qualities. Object detection approach classifies things based on these features. The same concept is applied in things like facial recognition and fingerprint recognition. It is possible to detect objects using both machine learning and deep learning approaches. Here, we'll use deep learning to enable us to complete the detection process without specifying the classification features in advance. Deep learning methods mainly rely on convolutional neural networks, in contrast to machine learning methods (CNNs). The techniques we'll employ are as follows: - Regional Suggestions (R-CNN, Fast R-CNN, Faster R-CNN) the layer of region proposal, which is a component of the region proposal network, generates bounding boxes around the image's objects using this method. This approach divides the image into a few superpixels, which are then combined close together. You don't look twice (YOLO). This real-time approach assists in the recognition of various objects in photographs. This programme employs regression to assist in determining class probabilities for the topic image. It divides the image into N grids of equal-sized SxS squares.

1.5 Organisation

This project report is divided into five distinct chapters, each of which is explained below:

Chapter 1: This gives a basic overview of our project, the issue statement that forms the basis of our project's aim, our main project goals, and the method or solution that was used to finish our assignment. to develop a classification model that can correctly find an object in an environment that has been camouflaged. to develop a classification model capable of correctly identifying an object in a covert environment.

Chapter 2: This chapter covers the history of object identification, classification, and camouflaged object detection, which starts with machine learning. The relevant information from standard books, journals, transactions, websites, etc. is used in the explanation of neural networks and in a quick introduction to a few distinct types of neural networks. This chapter goes into great detail about how problems with music genre classification models have been resolved using various neural network types, datasets, and accuracy measuring approaches by numerous researchers, all with varying degrees of success, as well as the method/approach that we decided to use.

Chapter 3: This chapter's main objective was to go over the step-by-step process for creating the environment for the study that was conducted for this report. In order to train the model for object recognition in camouflaged environments, we will first search for an object detection dataset that may be used. Next, we will create/visualise a Bounding Box or region of interest from the photos and preprocess the dataset. The main focus of the text is on the neural network that was employed, along with the features that were put into practise, such as the hardware and programming required to build the mode. Finally, it covers the many accuracy measures that were used in this study to evaluate the accuracy of our model.

Chapter 4: This chapter of the report gives us a comprehensive picture of the model's accuracy by covering the tests carried out and the related findings gathered at various stages of the project. When the test is over, the model builds on each stage to provide the most appropriate network model. In this chapter, the accuracy of various audio input formats is compared, the mislabeled network outputs are examined, and finally it is determined whether the input provided was correctly classified.

Chapter 5: This chapter covers the findings of our final project model as well as a summary and analysis of the research included in this report. It also points up potential areas for more or improved future study and development on the problem. It also includes the creative work, inventions, and fresh ideas that came from the analysis and its conclusions.

CHAPTER 2
LITERATURE SURVEY

The backdrop and the item can occasionally have colours that are similar, yet they have different texture patterns. The texture is thought to set the thing apart from its surroundings. Using a bottom-up aggregation architecture that blends the structural properties of texture elements with filter responses, Galun et al. created a method to identify camouflaged items. It recognises texture elements according to their form and describes them according to their size, aspect ratio, orientation, brightness, etc. To differentiate between different textures, various statistical measurements of these features are then taken into consideration. The aforementioned method is used on photos with diverse textures. This technique works best with images when the backgrounds and objects have different textures. However, if the item and backdrop have textures that are similar, this approach might not work well.

A method for multiple camouflage breaking employing a co-occurrence matrix and a Canny edge detector was put forth by Nagabhushan and Bhajantri in. The texture of the provided image is examined using the co-occurrence matrix, and edges are found using the Canny edge detector. The separation between objects with various textures is improved by combining the Canny edge detector with co-occurrence matrix. Despite producing strong Ajoy Mondal findings for artificial photos, this approach does not use real-world data.Prior to using this strategy, previous knowledge is also required. Neider and Zelinsky proposed finding concealed targets by sifting through distractions or inspecting background materials that resemble the target. Bhajantri and Nagabhushan suggested a method to find the camouflaged flaw in. Here, a tiny picture region's co-occurrence matrix-based texture characteristics are calculated. Watershed segmentation and cluster analysis are used to identify the problematic area. The texture feature determines how accurate this approach will be. Sequences with comparable textures for the backdrop and objects may not perform effectively. Sengottuvelan et al. created a method to recognise the object's camouflaged area and extract it from the surrounding area in a picture. Here, the concealed item is found using a texture feature and dendrogram based on the grey level co-occurrence matrix. Because the provided image is divided into multiple blocks or smaller parts, this process takes a long time. It is ineffective for photographs with shading effects and objects and backgrounds with similar textures.

To create and assess camouflage texture, Liming and Weidong suggested a method based on weighted structural similarity. Here, they created a camouflage image using weighted structural similarity and an original image feature. It can be used to remove camouflage. A number of background matching algorithms were developed by Owens et al. to make an object appear to be whatever is behind it.It is difficult to perfectly match the background from every conceivable angle. However, the suggested models must choose between several perceptual parameters, such as the prominence of occlusion borders and the degree of texture distortion. Li et al. presented a texture guided weighted voting approach to find foreground objects in camouflaged backgrounds in a similar vein . This technique divided the picture into frequency bands using the stationary wavelet transform. In some wavelet frequency bands, this approach was able to successfully capture tiny, hardly perceptible differences between the foreground and background in the picture domain. A weighted voting method based on the intensity and texture of all the wavelet bands is then used to identify the foreground.

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 Artificial Neural Networks

Artificial neural networks (ANNs) are adaptable artificial systems that model how the human brain functions. They are systems that have the ability to alter how they are set up in response to a practical objective. Because they can replicate the fuzzy rules defining the ideal solution for this sort of problem, they perform especially well when dealing with nonlinear challenges. The nodes, also known as processing elements (PE), and connections that make up an ANN are its fundamental components. Since each node has its own input and output ports via which it may connect with other nodes or the environment, each link is defined by these features. Connections that are either excitatory or inhibitory are shown by positive or negative values, respectively.

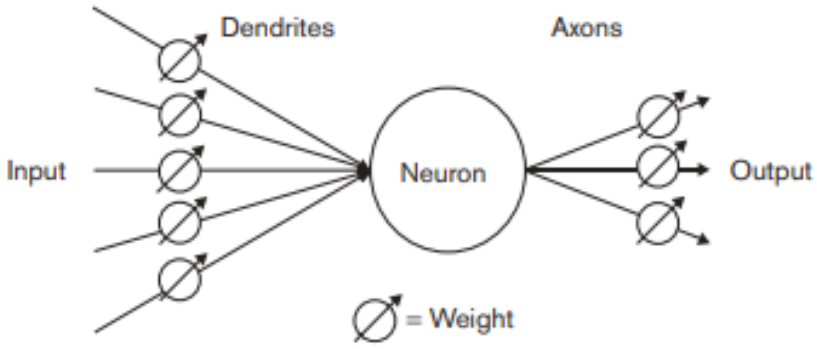


Figure 1: Artificial Neuron

Over time, the nodes' connections may evolve. This dynamic causes the whole ANN to begin to learn. The "Law of Learning" describes the process through which nodes adapt to their environment. An ANN's overall dynamic is time-dependent. Actually, for the ANN to change its connections, it has to communicate with the environment more frequently. Data constitute the ANN's surroundings. As a result, one of the fundamental qualities of ANNs, which are regarded as adaptive processing systems, is the learning process. By adjusting an ANN's connections to the environment's data structure, it is known as the Learning Process to "understand" the environment and the relationships that characterise it.

Neurons can be set up in any topological arrangement (e.g., one- or two-dimensional layers, three-dimensional blocks, or more-dimensional structures) depending on the nature and quantity of the input data. The feed-forward topology has been utilised to build the most prevalent ANNs. The number of PEs in an input layer will typically be determined by the number of input variables. One or more hidden layers operating within the ANN receive the information, which they subsequently send. The last component of this structure, the output layer, offers the outcome. Regardless of whether the outcome is a binary value or a single integer, the output layer only contains one PE. Backpropagation is the most popular architecture for neural networks, as shown in the image below.

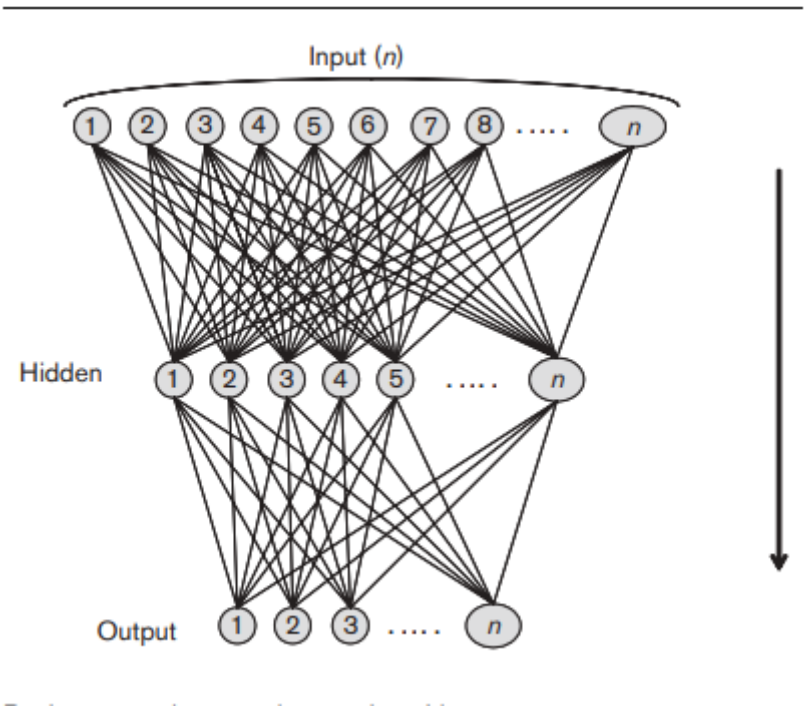


Figure 2: Back Forward Propagation

(Source: Introduction to artificial neural networks by Enzo Grossia and Massimo Buscemab)

Each PE inside the ANN is connected to every other PE nearby. These connections may be built by various neural network subtypes in various ways. Each of these links is given a "weight," which changes the input or output value.

The values of these connection weights are determined during the training process. These capabilities form the foundation of the ANN's learning ability. It is essential to understand that there are no categories as a result.

The most common problem that an ANN can deal with is as follows:
When there are N variables, for which data can be easily acquired, and M variables, which differ from the first and for which data collection is expensive and complex, ascertain if it is possible to forecast the values of the M variables based on the N variables.
If the M variables follow the N variables in time, the problem is referred to as a prediction problem; if the M variables depend on some sort of static or dynamic aspect, the problem is referred to as a recognition, discrimination, or extraction of fundamental characteristic problem. To effectively use an ANN to solve this kind of problem, a validation mechanism must be put into action. To start with a decent sample of examples, both the N variables (known) and the M variables (to be found) in each case must be known and reliable. The ANN must be trained on a sample of complete data before being evaluated for its predictive power.

While the validation process employs a subset of the sample to train the ANN (training set), the remaining instances are used to assess the ANN's prediction capabilities (testing set or validation set). This allows us to assess the ANN's reliability in resolving the problem before using it. There are many different kinds of protocols in the literature, each with advantages and disadvantages. One of the most popular uses is the so-called 5-2 cross-validation approach, which creates 10 elaborations for each sample. Five times the sample will be divided into two distinct subsamples, each with a similar mix of participants and controls.

As the technology is universally applicable and issues arise as quickly as queries that society, for whatever cause, raises, the potential range of problems that may be solved with ANNs is theoretically infinite. So let's review the requirements that must be met for the adoption of an ANN to be beneficial: The issue is complicated K Other approaches have not yielded sufficient outcomes K A rough answer to the issue is useful, not only a certain or ideal one. K A workable resolution to the issue results in significant cost and/or human savings K A significant body of case studies exists that illustrate the "odd" behaviour that the issue relates to.

3.1.1 The MNIST Dataset

A sizable collection of handwritten numbers may be found in the MNIST database (Modified National Institute of Standards and Technology database). There are 60,000 instances in the training set and 10,000 examples in the test set. It is a subset of two larger NIST Special Databases, Special Database 1 (which contains handwritten numbers written by high school students) and Special Database 3 (which contains handwritten digits produced by US Census Bureau personnel). The digits are centred in a fixed-size picture after being size-normalised. To fit in a 20x20 pixel box while maintaining their aspect ratio, the original black and white (bi-level) photographs from NIST were size normalised. The anti-aliasing approach employed by the normalisation algorithm results in the creation of grey levels in the final photographs. By calculating the centre of mass of the pixels and translating the image to place this point in the centre of the 28x28 field, the images were centred in a 28x28 image.

3.2 Convolution Neural Networks

Convolutional Neural Networks (CNN), a subtype of Artificial Neural Networks that has dominated a number of computer vision applications, are gaining traction in a variety of sectors, including radiology. CNN is designed to learn spatial feature hierarchies automatically and adaptively using a range of building blocks, including convolution layers, pooling layers, and fully connected layers. This review article provides insight into CNN's fundamental ideas and how they apply to various radiological tasks. It also discusses its difficulties and potential future applications in the field of radiology. This article will also discuss methods to minimise the problems of small dataset and overfitting when using CNN for radiological tasks.

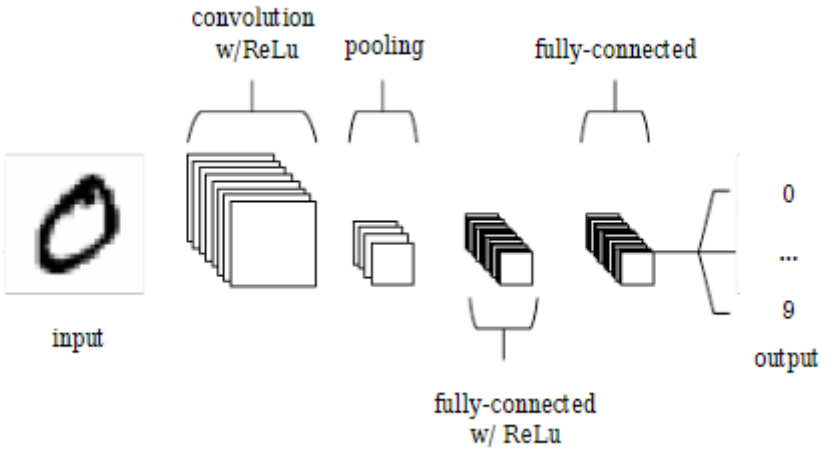


Figure 3: Image Classification using CNN

(Source: www.researchgate.net)

Three different types of layers make up a CNN. Convolutional, pooling, and fully-connected layers are what these are. A CNN architecture is created once these layers are stacked.

The basic functionality of the above example CNN above can be broken down into four key areas.

- The input layer will store the image's pixel values, as with other ANN variants.
- The convolutional layer will identify the output of neurons that are connected to particular local regions of the input by computing the scalar product between the weights of the neurons and the area connected to the input volume. The rectified linear unit, or ReLU, aims to apply a "element-wise" activation function, such as the sigmoid, to the output of the preceding layer's activation.
- To further reduce the number of parameters in that activation, the pooling layer will then simply downsample along the spatial dimensionality of the input.
- The fully-connected layers will then perform the same tasks as in conventional ANNs, attempting to extract class scores from the activations, which can then be utilised for classification. In order to enhance performance, it is also advised that ReLu be utilised between these layers.

These kernels are typically thinly distributed throughout the whole depth of the input, despite their low spatial dimensionality. Each filter is convolved across the spatial dimensions of the input by the convolutional layer as the data reaches it, creating a 2D activation map.

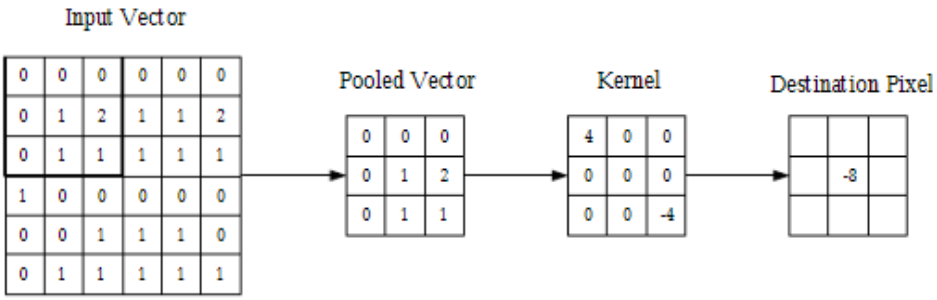


Figure 4: Pooling in CNN

(Source: Source: Introduction to artificial neural networks by Enzo Grossia and Massimo Buscemab)

The activation maps for each kernel will be layered along the depth dimension to create the convolutional layer's whole output volume. As we hinted at before, using inputs like photos to train ANNs yields models that are too large to train efficiently. This is due to the fully-connected nature of conventional ANN neurons, hence every neuron in a convolutional layer is only linked to a tiny portion of the input volume to counteract this. The size of the neuron's receptive field is a term used to describe the region's dimensionality. Nearly always, the depth of the input is equal to the magnitude of the connectivity through the depth.

Through the optimization of their output, convolutional layers are also able to significantly reduce the model's complexity. The three hyperparameters of depth, stride, and setting zero-padding are used to optimise these.

By manually adjusting the number of neurons in each convolutional layer to the same area of the input, the depth of the output volume that the layers create may be controlled. This is demonstrated by various types of ANNs, where every neuron in the hidden layer is directly coupled to every other neuron in the prior layer. While lowering this hyperparameter can drastically lower the network's overall neuron count, it can also significantly lower the model's capacity for pattern recognition. In order to position the receptive field, we may also define the stride in which we establish the depth around the spatial dimensionality of the input.

For instance, if we set the stride to 1, the receptive field would be greatly overlapped and produce tremendously massive activations. Alternately, increasing the stride will lessen the amount of overlapping and result in a result with smaller spatial dimensions. Zero-padding is a straightforward procedure that involves padding the input's border. It works well to provide further control over the output volumes' dimensionality. It's critical to realise that by utilising these strategies, we will change the spatial dimensionality of the convolutional layers' output. You may use the following formula to determine this:

$$\frac{(V - R) + 2Z}{S + 1}$$

Where Z is the amount of zero padding applied, V stands for the input volume size (height, breadth, and depth), R for the receptive field size, and S stands for the stride. Since the neurons won't fit neatly across the input if the estimated answer from this equation is not a whole integer, the stride has been adjusted improperly.

Pooling layer:

Pooling layers work to gradually lower the representation's dimensionality, which in turn lowers the model's computational complexity and parameter count. The "MAX" function is used by the pooling layer to scale the dimensionality of each activation map in the input. These typically take the shape of max-pooling layers with 2 2 dimensional kernels applied with a 2 stride along the input's spatial dimensions.The depth volume is kept at its original size, while the activation map is reduced to 25% of its original size. There are only two commonly noted max-pooling methods because the pooling layer is destructive. The pooling layers' stride and filters are typically both set to 2 2, which enables the layer to extend throughout the input's full range of spatial dimensions.

Additionally, with a stride of 2 and a kernel size of 3, overlapping pooling can be used. Because of the destructive nature of pooling, a kernel size greater than 3 will often cause the model's performance to deteriorate severely.. It is also crucial to realise that CNN topologies may include general pooling in addition to max-pooling. L1/L2-normalisation, average pooling, and other common operations may all be carried out by the pooling neurons found in general pooling layers. However, the usage of max-pooling will be the main emphasis of this course.

3.2.1 The CIFAR-10 Dataset

The CIFAR-10 dataset was created by researchers at the Canadian Institute for Advanced Research (CIFAR), commonly known as CIFAR. The dataset contains 60,000 colour photos (32 × 32 pixel) of things from 10 different categories, such as frogs, birds, cats, ships, and so on.

The class names and the corresponding ordinal integer values are mentioned below.

[0: Airplane, 1: Car, 2: Bird, 3: Cat, 4: Deer, 5: Dog, 6: Frog, 7: Horse, 8: Ship, 9: Truck]

These exceedingly small pictures — considerably smaller than a standard snapshot — were created for the dataset with computer vision research in mind. It is common practice in the field of machine learning to compare computer vision algorithms using the well-known dataset CIFAR-10. Obtaining 80% classification accuracy is not a difficult techniqueDeep learning convolutional neural networks, with a classification accuracy of 90% on the test dataset, provide the most effective solution.

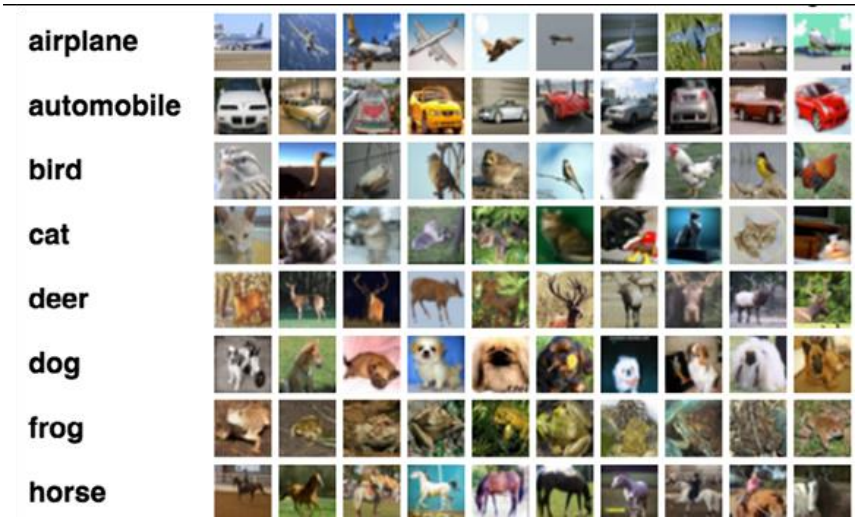


Figure 5: CIFAR-10 Dataset (Source: packtub.com)

3.3 Deep Neural Networks and Deep Learning

Deeper neural networks necessitate more labour to train. It proposes a residual learning architecture to help train networks that are far deeper than those previously used. It explicitly

reformulates the layers as learning residual functions with reference to the layer inputs rather than learning unreferenced functions. It contains a large amount of empirical evidence suggesting that these residual networks are easier to adjust and can benefit from significantly greater depth. We examine residual networks with up to 152 levels of depth on the ImageNet dataset—eight layers deeper than VGG nets but with less complexity. An ensemble of these residual nets achieves a 3.57% inaccuracy on the ImageNet test set. At the ILSVRC 2015, this work won first place in the classification problem. For many visual recognition tasks, the depth of representations is critical. We achieve a 28% relative improvement on the COCO object detection dataset solely because of our extremely deep representations.

When deeper networks start to converge, a degradation problem arises: as network depth increases, accuracy gets saturated (which is relatively expected) and rapidly degrades. Surprisingly, such worsening is not the result of overfitting, because adding more layers to a sufficiently deep model increases training error, as stated in and fully supported by our studies. A sample example is shown below.

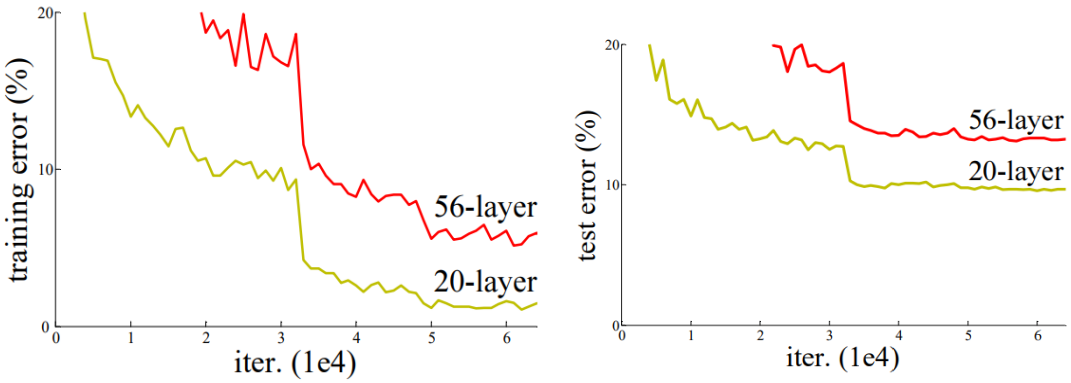


Figure 6: Training error vs iterations (Source: packtub.com)

We expressly allow these layers to fit a residual mapping while a Deep Neural Network fits a specified underlying mapping.Formally, we allow the stacked nonlinear layers to match another F(x):= H(x)x mapping, designating the intended underlying mapping as H(x). F(x)+x is the transformation of the original mapping. We feel that optimising the residual mapping is easier than optimising the unreferenced initial mapping. If identity mapping were optimal,

reducing the residual to zero would be faster than fitting an identity mapping with a stack of nonlinear layers.

Feedforward neural networks with "shortcut connections" can implement F(x) +x. Shortcut connections are those that allow you to skip one or more tiers. In our situation, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Identity shortcut links introduce no new parameters or computational complexity. SGD with backpropagation may still be used to train the complete network end-to-end, and it can be readily implemented using popular libraries without altering the solvers. We give extensive ImageNet tests to demonstrate the degradation problem and assess our solution.

In Deep Neural Network we observed the following:

1. Although it is easy to optimise our extremely deep residual nets, their "plain" counterparts (which do nothing more than stack layers) have higher training error as depth increases.
2. Increasing depth may be easily benefited by our deep residual nets, producing results that are far superior than those of prior networks. Similar instances are seen on the CIFAR-10 dataset, proving that our strategy's optimization problems are not specific to that dataset. On this dataset, we show models with over 100 layers and models with over 1000 layers that were successfully trained.
3. Using incredibly deep residual nets on the ImageNet classification dataset, we achieve good results. A 152-layer residual net, the deepest network ever reported on ImageNet, is simpler than VGG.

The extremely deep representations also perform well in terms of generalisation. This compelling evidence suggests that the residual learning principle is generalizable, and we anticipate that it will be applicable to other vision and non-vision problems.

3.4 Object Recognition

It is a technique for locating the things that may be seen in pictures and movies. It is one of the most important applications of machine learning and deep learning. The goal of this discipline

is to teach robots how to instinctively understand an image's information much like humans do. When we look at a photograph or watch a video, we can quickly tell who is in the scene and what is being shown. Autonomous vehicles can discern between a lamppost and a person or a stop sign thanks to object recognition technology. It also provides advantages in a number of applications, such as robotic vision, industrial inspection, and disease detection in bioimaging.

Any object recognition has the following steps involved:

1. Predict the class of an object in an image using image classification.
2. **Object Localization:** Determine the presence of objects in a picture and mark their location with a bounding box.
3. **Object Detection:** Use a bounding box to locate the presence of items and detect the classes of the objects found in these boxes.
4. **Object Segmentation**: In order to maximise storage and resource consumption for large items, object segmentation is the act of dividing an object into a group of smaller fixed-size objects.

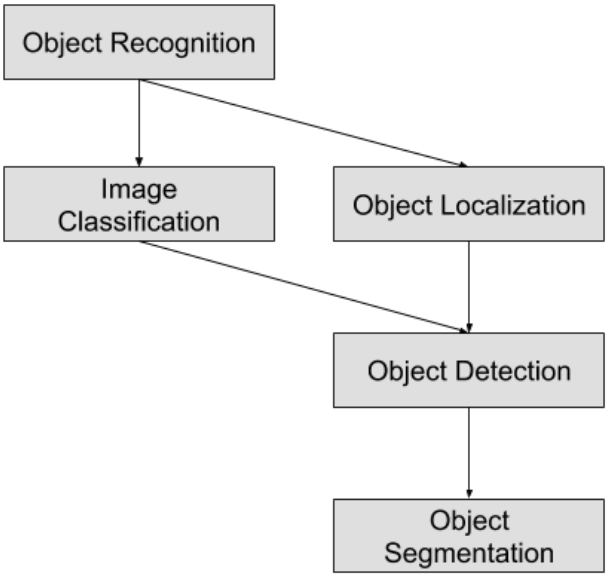


Figure 7: Object Recognition Flow (Source:www.cloudfront.net)

3.4.1 Object Recognition using Deep Learning

The method for object recognition that is most often used is convolution neural network. This CNN network returns the probability of each class based on an image as its input. The chance of production for the item is high if it is visible in the image; otherwise, the likelihood of production for the other classes is either negligible or low. Deep learning has the advantage over machine learning in that it does not require feature extraction from the data.

Challenges of object recognition:

- This approach will not work if the image contains multiple class labels because the CNN model only recognises a single class label.
- A different approach is required if we want to specify if an object is present within the bounding box.

3.4.2 Image Classification

It is the process of labelling or categorising the pictures once information has been extracted from them. It accepts an image as an input and returns a classification label for the picture along with metrics like likelihood, loss, accuracy, etc.

The following steps are required for classifying images:
Step 1:Import the required libraries. Choose an interesting dataset or design your own picture dataset to solve your own image classification challenge.
Step 2: Load the data and perform some basic EDA (Exploratory Data Analysis). Assigning routes, generating categories (labels), and resizing our images will all be part of preparing our dataset for training.
Step 3: Divide the datasets into training and testing. Training is an array that contains picture pixel values as well as the index of the image in the categories list.
Step 4: Build the model.
Step 5: Compile and train the model.
Step 6: Test the accuracy of the model, create a confusion matrix etc, using the testing data.

There are two types of classification:

1. **Binary classification**:

In this type of classification our output is in binary value either 0 or 1, which represents whether the object to be found is present in the image or not. For the 'Not found' label we use 0, and for the 'Found' label we use 1. As an example, suppose users are creating a classification model that identifies whether or not a picture contains a koala. In order to extract different information from an image, multiple filters are used. In this case, a filter may pick up on the koala's eyes and fangs, while another might pick up on its ears, etc. To extract the information, we make use of these filters.

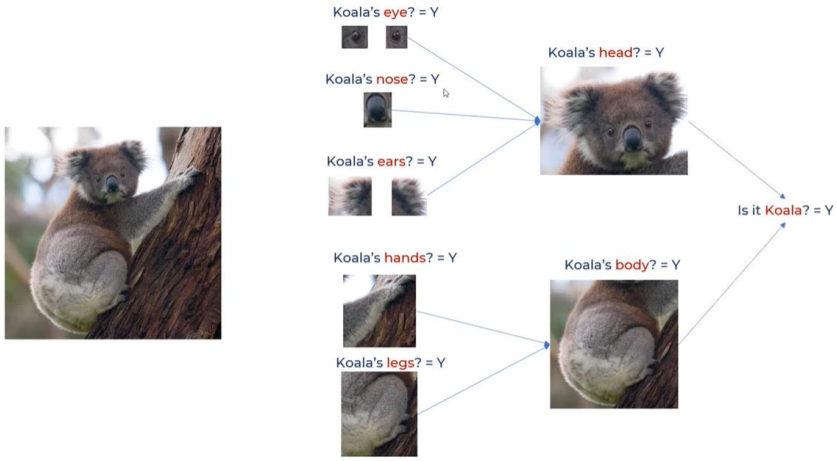


Figure 8: Feature Extraction of Koala Bear (Source:www.analyticalvidya.com)

Let's examine in more detail with some graphics how the convolution filter is applied in our image.

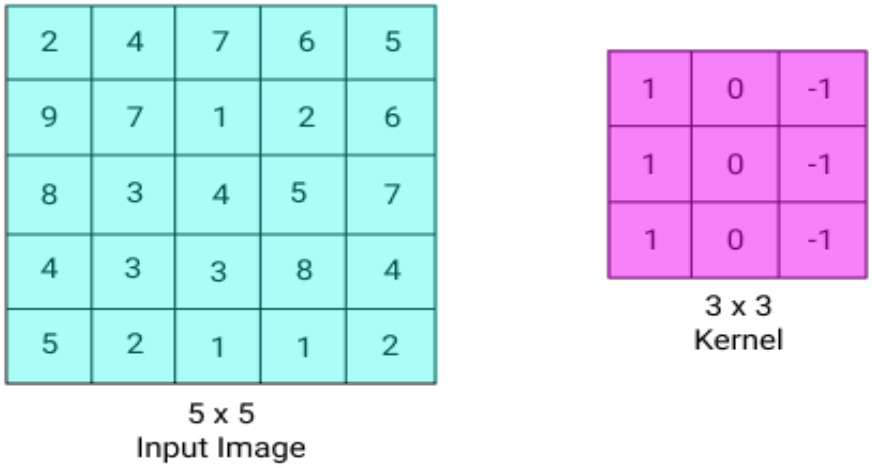
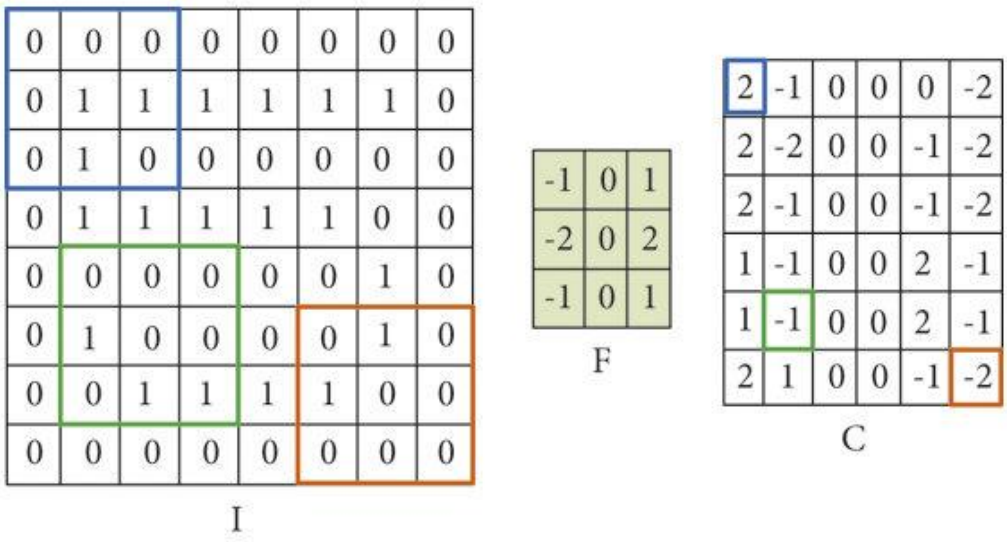


Figure 9: Convolution Operation on Sample Image using Kernel Matrix

(Source: www.medium.com)

Given: Using our three-by-three learnable filters after taking a grayscale image, we do the convolution procedure.



Now, instead of using the predetermined number of channels, we must apply the same rules for RGB scale or colourful images.

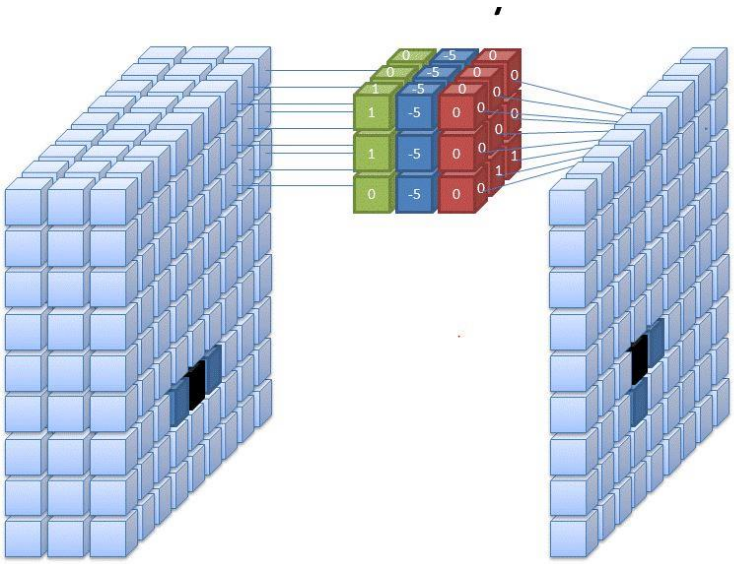


Figure 10: Convolution for RGB Image (Source:wikipedia.com)

Multi-class classification:

In this type of classification our output will be multi-class, instead of two.

3.5 Object Localization

An instance of a certain item class may be found in an image by specifying a closely cropped bounding box that is centred on the instance. The specification of an object localization is a human-labelled "ground-truth" bounding box for the provided item, which is deemed to be a proper localization if it adequately overlaps it. The "Object Localization" task's objective is to locate a single instance of a certain item type.

Figure 11: Sample Data for Object Localization (Source: www.fritz,ai)

In order to generate a bounding box around an item of interest, the object localization technique predicts a collection of four continuous integers, namely the x coordinate, y coordinate, height,

and breadth. Convolutional neural network layers are frequently used as the first few layers in CNN-based classifiers. The number of layers used might range from a few to 100 depending on the application, the amount of input, and the available computer capability (for example, ResNet 101).

The number of layers is an important subject of research. Following the CNN layers are a pooling layer and one to two entirely linked layers. The output layer, which is the final layer, determines whether or not an item appears in an image. Consider the following scenario: an algorithm recognises 100 distinct things in a photograph. The last layer then returns an array with a length of 100 and values ranging from 0 to 1, indicating the chance of an object appearing in a picture.

Everything in an object localization algorithm is the same except for the output layer. The final layer of classification algorithms produces a probability value between 0 and 1. Because it is a regression problem, localization approaches produce results in four actual numbers.

3.5.1 The Loss Function L(y,y`) for Object Localization

To forecast values as closely as feasible to the real values is the aim of any machine learning algorithm. A loss function is a necessary component of every supervised machine learning technique used to do this, and algorithms learn by minimising loss through the optimization of weights or parameters.

Since object localization is a regression issue, any regression loss function that is suitable for an N-dimensional array can be used. Huber loss, L1 and L2 distances, for instance, and so on. L2 distance loss is commonly used by both the corporate and scientific communities, as was previously mentioned.

3.5.2 Euclidean Distance (L2 Norm):

L2 distance is sometimes referred to as Euclidean Distance. By employing their cartesian coordinates, the Pythagorean theorem is used to calculate the distance between two points in N-dimensional space. With this approach, any N-dimensional space may be employed.

To further understand L2, let's consider two three-dimensional points, P and Q. Points P and Q are represented in the cartesian coordinate systems by (p1, p2, p3) and (q1, q2, q3), respectively.

$$\text{Distance}(P, Q) = \text{sqrt}((p1-q1) + p2-q2) + p3-q3)).$$

The optimisation procedure seeks to minimise the Euclidean distance between the actual and predicted values; the lower the L2 distance between the prediction and the actual value, the better the technique.

3.6 Object Detection

Object detection is a computer vision approach that assists in the recognition and localization of items in an image or video. Object detection properly defines bounding boxes around the items it has identified, allowing us to determine their location inside (or movement across) a scene.

It's important to clarify the differences between object detection and picture identification before moving on since they can occasionally be misunderstood.

Image recognition assigns a label to a picture. A photograph of a dog is referred to as a "dog." Even today, an image of two dogs is described as a "dog." The model predicts the position of each object and the corresponding label, whereas object detection surrounds each dog with a box labelled "dog".

While the image classification challenge focuses on categorising the photos, in object recognition our objective is to identify all of them set in the most relevant boxes. In one image there may be more than one class we are searching for. So, compared to object identification, object detection provides more information about a picture.

Deep learning-based and machine learning-based object detection methods can generally be distinguished from one another.

In more traditional ML-based techniques, clusters of pixels that might constitute an object are found by applying computer vision algorithms to examine various features of a picture, such as the colour histogram or edges. These attributes are then used as input into a regression model to predict the object's position and label.

Deep learning-based systems use convolutional neural networks (CNNs) to do end-to-end, unsupervised object identification since deep learning techniques are currently cutting-edge technologies for object recognition. This eliminates the requirement for separate feature definition and extraction.

3.6.1 Basic Layout of Object Detection:

Deep learning-based object recognition models typically include two components. An encoder takes an image as input and processes it via a series of layers and blocks that educate them to extract statistical features that may be used to identify and locate objects. The encoder's outputs are received by a decoder, which calculates the bounding boxes and labels for each item.

The simplest decoder is a pure regressor. By connecting to the encoder's output, the regressor directly predicts the position and size of each bounding box. The model's output is the X, Y coordinate pair for the object and its area in the image. While being basic, this paradigm has limits. You must specify the number of boxes ahead of time. If your image has two canines but your model can only recognise one of them, one will be kept unidentified. Pure regression-based models, on the other hand, may be a viable option if you know ahead of time how many things you need to forecast in each image.

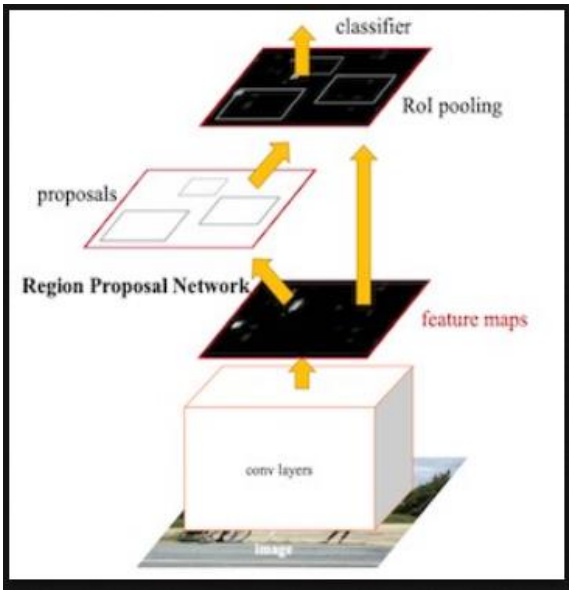


Figure 12: Basic Structure (Source:www.medium.com)

Single Shot Detectors (SSDs) try to find a happy medium. SSDs depend on a set of specified regions rather than a subnetwork to suggest areas. The input image is covered with a grid of anchor points, and regions are placed at each anchor point as boxes in a variety of sizes and shapes. The model provides a prediction of whether or not an object exists within the region for each box at each anchor point, as well as adjustments to the box's location and size to better match the object. Since there are several boxes at each anchor point and anchor points might be close together, SSDs can result in a lot of overlapping detections. In order to exclude the majority of these predictions and choose the best one, post-processing must be done to SSD results. Non-maximum suppression is the most well-known post-processing method.

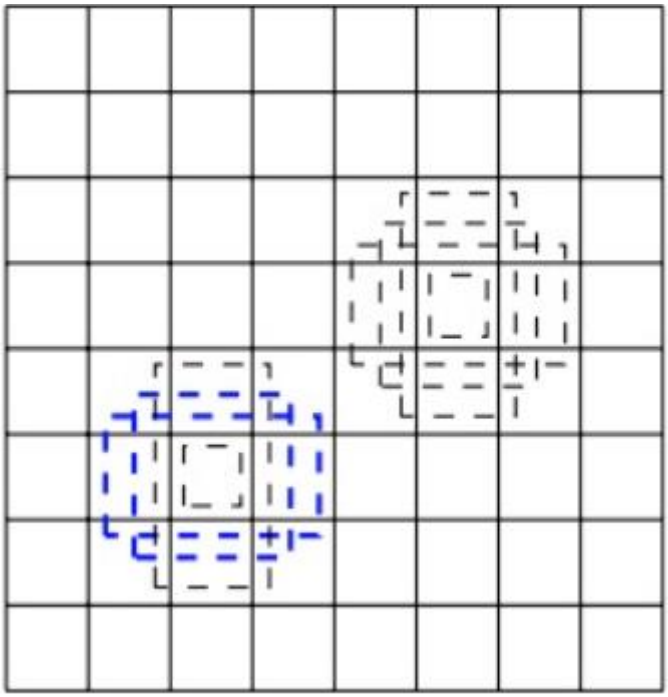


Figure 13: An Illustration of Single shot detectors (Source: www.medium.com)

A region proposal network is a regressor technique expansion. This decoder's model indicates areas of an image where it believes an object may be present. The pixels from these regions are then sent into a classification subnetwork, which generates a label (or rejects the proposal). The pixels that include such regions are then sent via a classification network. This method provides a more exact, adaptive model that may recommend any number of places that may include a bounding box. However, the decreased computational efficiency comes at the tradeoff of higher precision.

3.6.2 Model Architecture Overview

Object Recognition Neural Network Architectures created until now are divided into 2 main groups, i.e.

- Single-stage Detectors and,
- Multi-stage Detectors.

Single-stage Detectors:

Without the use of pre-generated area suggestions (candidate object bounding boxes), object classification and bounding-box regression are performed directly.

Multi-stage Detectors:

Region proposals are created, for example, by selective search as in R-CNN and Fast R-CNN or through a Region Proposal Network (RPN) like in Faster R-CNN. Additionally, additional actions can be taken, such as binary-mask prediction and bounding-box regression for improving region suggestions. Although slower than one-stage detectors, two-stage detectors often achieve higher levels of accuracy.

Bounding Box – ROI (Region Of Interest):

We must satisfy a new definition for object recognition. Bounding boxes are used in our attempts to recognise items so that they may be discovered. Later, we shall discover how to get near boxes to the thing that was discovered. As we can see, identifying objects in photographs is a bit more difficult than trying to localise and identify them using image categorization.

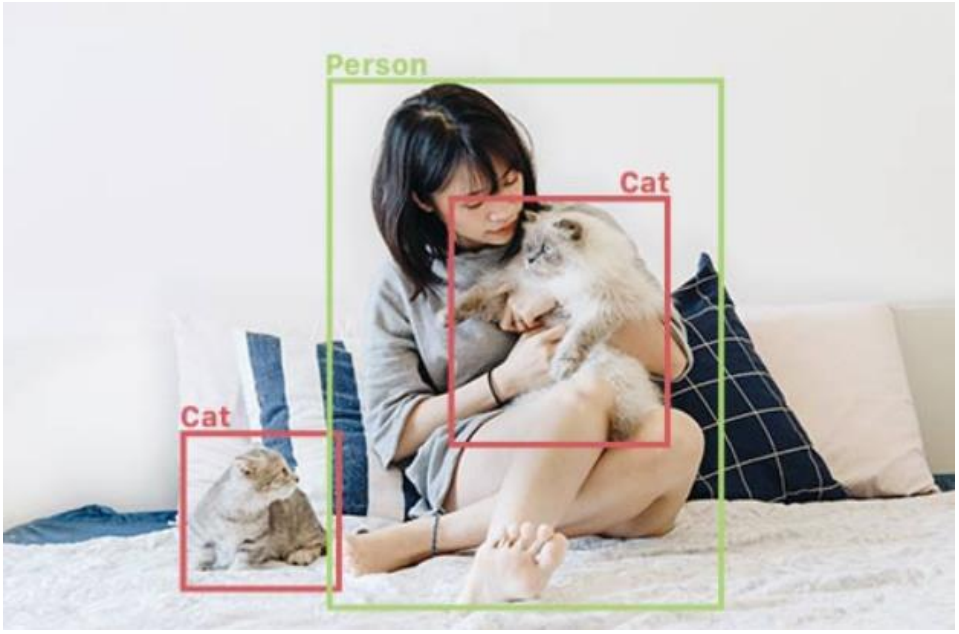


Figure 14: Bounding Box (Source: www.medium.com)

3.6.3 UIEB Dataset

Due to its importance in marine engineering and aquatic robotics, underwater image enhancement has been garnering a lot of interest. Over the past few years, many methods for improving underwater images have been presented. However, these algorithms are mostly tested on either artificial datasets or a small number of carefully chosen real-world photos. Because of this, it is difficult to predict how these algorithms would perform on photographs taken in the wild and how we can evaluate our success in the field. We provide the first thorough perceptual investigation and analysis of underwater picture enhancement utilising massive real-world photographs to close this gap. In this study, we build an Underwater Image Enhancement Benchmark (UIEB) from 950 underwater photographs taken in the field, 890 of which include equivalent reference photographs.

Figure 15: A sample image from the Constructed UIEB Dataset
(Source:www.analyticalvidya.com)

3.6.4 MS COCO Dataset:

The MS COCO (Microsoft Common Objects in Context) dataset is a large object recognition, segmentation, key-point detection, and captioning dataset. The collection has 328K photographs.

Splits: The MS COCO dataset was first released in 2014. There are 164K photographs in all, with 83K training shots, 41K validation photos, and 41K test photos. The 81K picture additional test set issued in 2015 comprised all previous test shots as well as 40K new images. Based on community feedback, the training/validation split was changed from 83K/41K to 118K/5K in 2017. In the new split, the same photographs and notes are used. The 2017 test set

includes a subset of the 41K pictures from the 2015 test set. The 2017 edition also includes a fresh 123K picture unannotated dataset.

Figure 16: A collection of samples in the MS COCO dataset (Souce: paperswithcode.com)

3.6.5 VOC: Visual Object Classes (2007) Dataset

The provided training data is a collection of images, each with an annotation file including a bounding box and an object class label for each of the twenty classes represented in the image. Keep in mind that a single image may include several objects from various classes. A few samples of photographs are available to online readers. All annotators were given instructions on how to annotate, which they all followed. These regulations may be found here. The data will be released in two stages, the first of which will involve the release of a developer kit that contains training and validation data as well as evaluation tools (written in MATLAB). The validation set's goal is to demonstrate the operation of the evaluation software prior to competition submission. In the second stage, the test set will be made accessible for the actual tournament. Similarly to the VOC2006 challenge, the ground truth of the test results will not be made public until the challenge is completed. Half of the data will be utilised for training/validation, while the other half will be used for testing. The distributions of pictures and objects per class in the training/validation and test sets are similar. There are a total of 9,963 photos with 24,640 identified items.

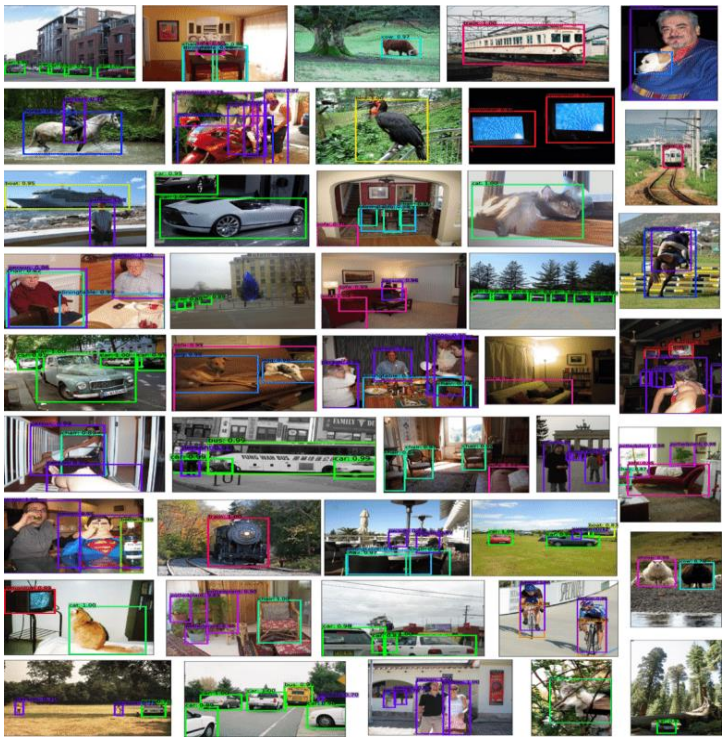


Figure 17: Sample images of VOC (2007) (Source: withpascalcode.com)

3.7 Object Localisation

The main goal of this strategy is to make sure that each object has distinctive characteristics. These qualities can help in object separation, classification, and detection. The same idea is used in things like facial recognition, fingerprint recognition, etc. Both machine learning and deep learning methods may be used to detect objects. Using a deep learning approach, we can complete the detection process in this case without having to define the classification features in advance. Convolutional neural networks are frequently utilised in the deep learning technique in contrast to machine learning (CNNs).

The following actions must be taken as part of the object detection process:

- Utilising visual input provided via a video or image.
- Regionize or divide the input visual into pieces.
- Work on each section separately and as a single image.
- Submitting these images to our convolutional neural network (CNN) for probable class classification.

- After classifying the images, we can integrate them all to create the original input image along with the objects that were found and their labels.

The following are the strategies we'll use:

- Regional Advice (R-CNN, Fast R-CNN, Faster R-CNN): This method uses the region proposal layer, a part of the region proposal network, to create bounding boxes around the objects in the picture. With this technique, the image is divided into a few superpixels and combined close by.
- You never look twice (YOLO): This real-time technique helps with object detection in photos of various types. This programme uses regression to obtain the class probabilities for the topic image. It works by splitting the picture into N grids of SxS squares of equal size.

Family of Region-based Convolutional Neural Networks:

The R-CNN Family includes various object detection models. The region proposal structures serve as the foundation for these detection models. These qualities are advancing greatly over time, becoming more accurate and effective.

- **R-CNN:**

  The R-CNN approach uses a technique known as selective search to extract the elements from the image. This approach generates a large number of areas and works on them all at the same time. If there are any items in these locations, they are checked to determine if they exist. The accuracy with which objects are classified influences how effectively this technique works.

- **Fast R-CNN:**

  The Slow-RCNN technique uses the SPP-net (Spatial Pyramid Pooling) and R-CNN structure to accelerate the Slow-RCNN model. The Fast-RCNN uses the SPP-net to calculate the CNN representation for the full picture just once. This is then used to determine the CNN representation for each patch created by the R-selective CNN search technique. The fast-RCNN trains the entire process from beginning to end. Along with the training procedure, the Fast-RCNN model also contains bounding box regression. This combines the localization and classification procedures into one, hastening the process.

- **Faster R-CNN:**

The Faster-RCNN technique is even faster than the fast-RCNN approach. Although Fast-RCNN was fast, RPN (Region Proposal Network) was utilised to replace the selective search technique in Faster-RCNN. The RPN speeds up the selection process by employing a small neural network that generates regions of interest. To cope with the varying aspect ratios and sizes of objects, this technique employs Anchor Boxes in addition to RPN.Faster-RCNN is one of the most precise and successful object identification systems.

- **YOLO (You Only Look Once):**

In contrast to the YOLO framework, which focuses on the entire image and predicts the bounding boxes before calculating its class probabilities to label the boxes, the R-CNN approach described above focuses on the division of a visual into parts and focuses on the parts with a higher probability of containing an object. The YOLO framework family is made up of highly fast object detectors. The following is a discussion of the various YOLO models:

- **YOLOv1**

Because it combines the object detection and classification models into a single detection network, this model is also known as the YOLO unified. This was the first attempt to create a network that detects real-time objects very fast. YOLO only predicts a limited number of bounding boxes to achieve this goal.

- **YOLO v2 and v3**

YOLOv2 and YOLOv3 are upgraded versions of the YOLOv1 framework. YOLOv2 is also known as YOLO9000. YOLOv2 addresses the localization concerns that plagued the YOLOv1 framework by focusing on recall and localization. The following techniques are used: anchor boxes, high-resolution classifiers, fine-grained features, multi-level classifiers, bDarknet19, YOLOv2, and batch normalisation. All of these things make V2 better to V1. In the Darknet19 feature extractor, items in the image are classified using a softmax layer, five max-pooling layers, and 19 convolutional layers. YOLOv3 is the quickest and most accurate approach for object identification. It reliably identifies the objects using logistic classifiers, as opposed to YOLOv2's softmax technique. Now we can make categories with several labels.In order to advance accuracy, The YOLOv3 also leverages Darknet53, a feature extractor

with 53 convolutional layers, an improvement over the v2's usage of Darknet19. It also utilises a tiny object detector to detect all of the little items in the picture.

3.8 Object Segmentation

Image segmentation is a function that creates an output from input images. In the output, each pixel's object class or instance is specified by a mask or matrix with numerous elements.

Figure 18: Object Segmentation (Source: www.medium.com)

Different neural network implementations and designs are appropriate for picture segmentation. They typically share the same fundamental elements:

An **encoder** is a set of layers that uses deeper, more focused filters to extract visual features. If the encoder has prior experience with a comparable task (like image recognition), it may be able to use that experience to complete segmentation tasks.

A **decoder** is a collection of layers that gradually transforms the output of the encoder into a segmentation mask according to the pixel resolution of the input image.

Skip-Connections are multiple long-range neural network connections called "skip connections" that enable the model to recognise features.

Some common image segmentation techniques are:

1. **Edge-Based Segmentation**

A common method for processing images that recognisesEdge-based segmentation refers to the process of separating the edges of several objects in a photograph. It aids in finding features of linked things in the image by using information from the edges. Edge-based segmentation algorithms detect edges based on changes in contrast, texture, colour, and saturation. Edge detection aids in the removal of unnecessary information from images, reducing their size and allowing for analysis. Edge chains, which are made up of individual edges, can be used to precisely portray the borders of objects in an image.

2. **Threshold-Based Segmentation**

The most basic picture segmentation approach is thresholding, which divides pixels depending on their intensity relative to a predetermined value or threshold. It is appropriate for segmenting things that are more intense than other objects or backgrounds. In low-noise photos, the threshold value T can function as a constant. In some circumstances, dynamic thresholds can be used. Thresholding separates a grayscale image into two segments based on their relationship to T, resulting in a binary image.

3. **Region-based Segmentation**

It entails splitting a picture into sections with comparable properties. Each area is a collection of pixels that the algorithm finds using a seed point. After locating the seed points, the algorithm can expand areas by adding more pixels or decreasing and combining them with other points.

4. **Cluster-Based Segmentation**

Unsupervised classification methods like clustering algorithms make it easier to find inconspicuous details in pictures. By distinguishing between clusters, shades, and structures, they improve human vision. The method divides data into discrete pieces and groups similar components into clusters to divide images into groups of pixels with similar characteristics.

5. **Watershed Segmentation**

Watersheds are alterations to grayscale images. Similar to how topographic maps analyse images, watershed segmentation algorithms do the same, with pixel brightness indicating elevation (height). This method identifies the boundaries between the watershed lines and the ridge and basin lines. It divides images into sections depending on the height of the pixels, putting together pixels with the same grey value. There are several uses for the watershed technique, including medical image processing. For instance, it can help distinguish between lighter and darker regions in an MRI scan, which can help with diagnosis.

3.9 Camouflage Object Detection:

Images that have been camouflaged may be broadly categorised into two categories: those with natural camouflage and those with artificial camouflage. Animals (such as insects and cephalopods) employ natural camouflage as a survival tactic to avoid being seen by a predator. Contrarily, man-made camouflage typically appears in items as manufacturing flaws or is employed in video games and other forms of art to conceal information.

3.9.1 COD Formulation

COD is a class-independent task as opposed to class-dependent tasks like semantic segmentation. As a result, COD is a straightforward concept to formulate and describe. A strategy for detecting camouflaged objects must be used to assign each pixel in a picture a confidence score, pi, between 0 and 1, where pi stands for the pixel's probability score. Pixels that are not totally allocated to the disguised objects receive a score of 0, while those that are given a value of 1 are considered to be fully assigned. Instance-level COD will be the subject of future study since this research concentrates on the object-level COD problem.

3.9.2 Evaluation Metrics

In SOD operations, mean absolute error (MAE) is commonly employed. We employ the MAE (M) measure, as Perazzi et al. used, to compare the pixel-level accuracy of a forecast map (C) with a ground-truth (G). While valuable for assessing the frequency and magnitude of errors,

the MAE metric cannot pinpoint their precise location. Fan et al. recently presented an E-measure (E) based on human visual perception that analyses both image-level statistics and pixel-level matching. This measure is ideal for evaluating the outcomes of disguised object recognition in terms of global and local accuracy. COD requires a measurement that can evaluate structural similarities since disguised things usually have complicated geometries. As an alternative metric, we use the S-measure (S).

We investigate this metric in the COD sector since current research suggests that the weighted F-measure F(w) provides more trustworthy evaluation findings than the regular F. The majority of camouflaged images are from Flickr and have been used for academic purposes using the keywords "camouflaged animal," "unnoticeable animal," "camouflaged fish," "camouflaged butterfly," "hidden wolf spider," "walking stick," "dead-leaf mantis," "bird," "sea horse," "cat," "pygmy seahorses," and others. The remaining 200 camouflaged pictures are from sources such as Visual Hunt, Pixabay, Unsplash, Free-images, and others that provide royalty-free, public-domain stock photography.

To remove selection bias, we also chose 3,000 powerful images from Flickr. 1,934 non-camouflaged images from the Internet, featuring backdrops like forests, grasslands, ocean, snow, and the sky, were chosen to augment the negative samples. Additional details on the picture selection procedure are provided by Zhou et al.

Proposed model:

We demonstrate the proposed method for object detection and recognition using a CNN-based architecture in this section. In order to identify the important portions of the picture based on the suggested model, we first assess the object's saliency computation. Convolution layers are then utilised to construct the activation maps in a CNN model. The next step is to compute the hierarchical feature mappings and create a salient feature, or uniform feature space, using the Max Pooling and deconvolution procedures. An ROI pooling model is then created to provide mapped image features based on these processes. With the use of this feature map image, we create suggestions using the region proposal generation technique; nevertheless, the initial region proposals that are created are noisy because of background region response and edge responses. The ROI (region of interest) for feature extraction is also determined using saliency-based segmentation. This ROI offers the corresponding class and resolution details to guarantee detection quality. In order to enhance the proposal generating process, regions are refined, and

an object detection module is employed to finish the detection process. The recommended technique's whole flow is shown in the diagram below.

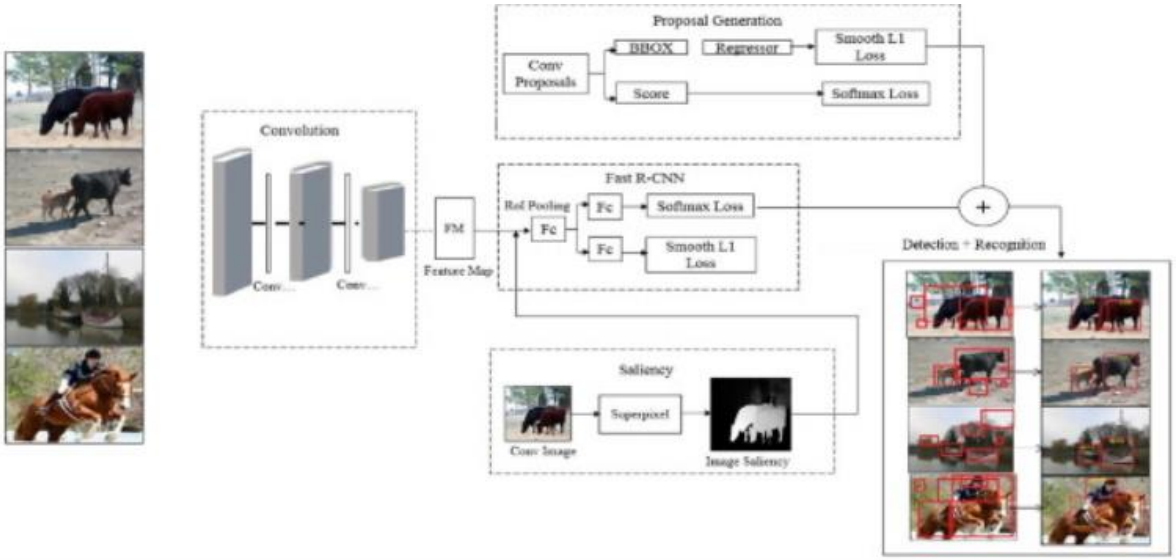


Figure 19: Model architecture (Source: www.sciencedirect.com)

The detection process may be separated into two stages: segmentation and detection, which offer the semantic map and bounding boxes on the image, which are useful for recognition later on. These results are then processed to generate preliminary estimates of the bounding boxes and the box region that is likely to be considered as the segmentation region. These bounding boxes are then processed as original predictions and sent into CNN, which provides image recommendations and statistics on the total accessible elements in the image. Following object recognition, the bounding box regression model is utilised to decrease bounding box errors before employing the identification approach.

Saliency detection:

This section describes saliency detection modelling, which is used to extract semantic data about prominent locations. Because the principal objects in certain cases are hidden, we focused on retrieving saliency data even from camouflaged photos in this study. The proposed saliency model therefore aids in improving detection precision even when camouflaged items are present in the picture. Saliency detection in this study assesses texture data, which may then be utilised to detect salient elements, as seen below:

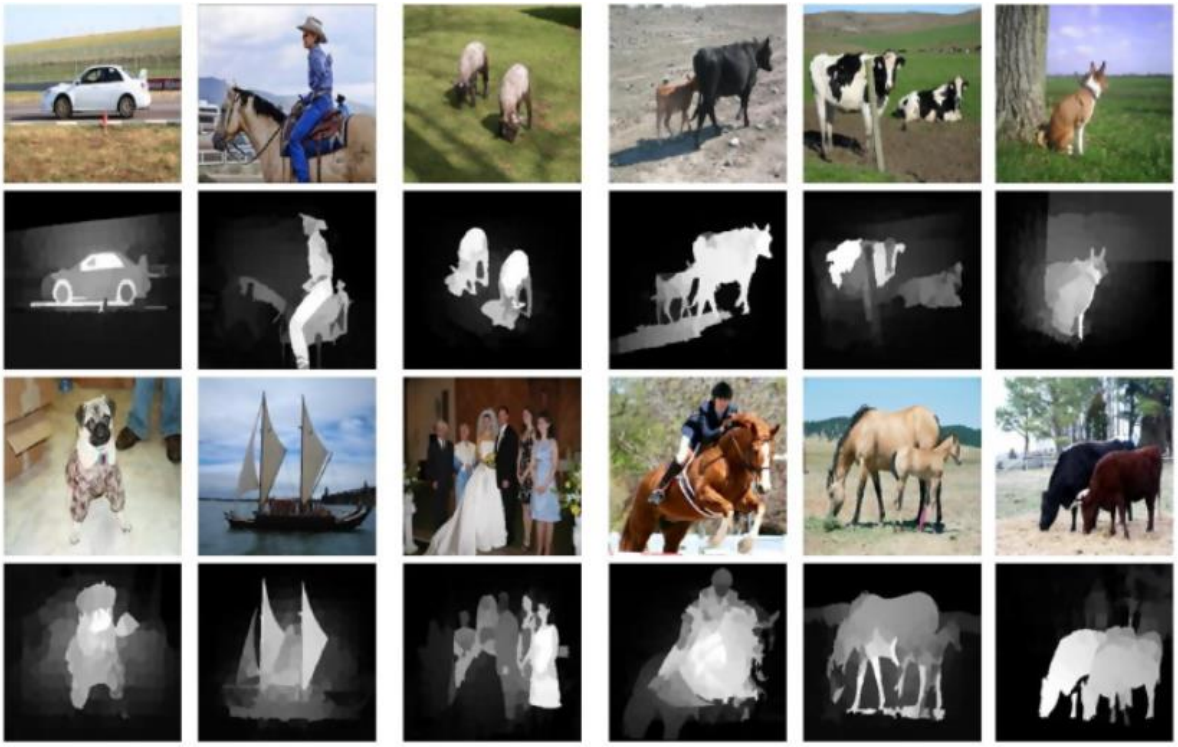


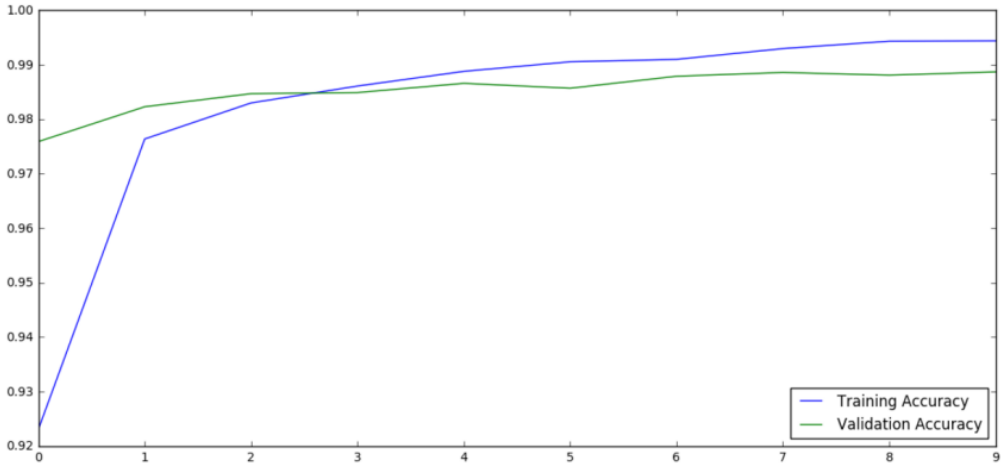
Figure 20: Dataset Sample (Source: www.sciencedirect.com)

According to the proposed strategy, graph theory-based hypergraph modelling is used to determine saliency. Superpixel information is provided by hypergraph modelling to provide comprehensive data on the consistency of inner and external pixels.

CHAPTER 4
PERFORMANCE ANALYSIS

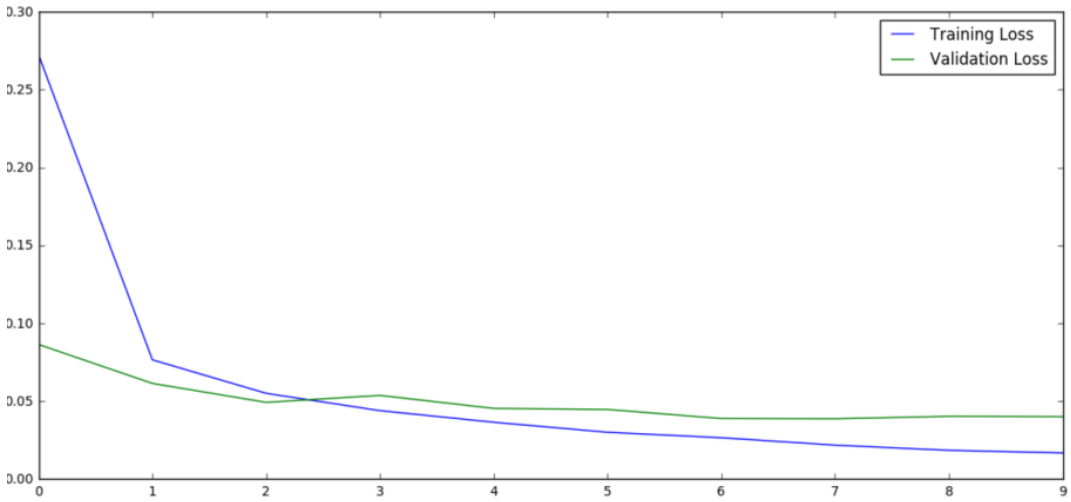
4.1 Performance Analysis of MNIST Dataset

After 10 iterations, the training set accuracy is 0.9952 and the validation set accuracy is 0.9894, which is a very good result on both the training and validation sets. As predicted, the loss value is 0.0169, and the validation loss is 0.0380 in the end. When we apply the model to the test set, we receive a promising result when we use Keras' evaluation function. We reached a precision of 0.99030000627. This is predicted to be slightly lower than the training set's accuracy, but still extremely good. We got 0.0287198445676 for the loss rate. In the Keras model, these two numbers are the only outputs of the evaluation function. Loss is a scalar value that must be minimised during model training. The lesser the loss, the more accurate our projections are. The loss option in the compile function is set to "categorical cross-entropy." "It is a Categorical cross-entropy between an output tensor and a target tensor" by its definition in the Keras manual, which is preferable to be chosen for multi-class problems. Basically, the accuracy is a fraction of properly predicted cases, thus the fraction of misclassified cases is equal to 1 - Accuracy, which is the error (rate). Fortunately, in comparison to the results on the MNIST website, we have a relatively good performance here, because, as shown in the ranking table, the Convolutional solutions have an error rate ranging from 1.7 to 0.2 per cent due to different code implementations and also different implementation of the specific packages and libraries that they have used. LeCun et al. (1998).



Graph 1: Result Accuracy Rate

(Source-Convolutional Neural Network solution for MNISTDataset)



Graph 2: Loss Rate of Training Set

(Source-Convolutional Neural Network solution for MNISTDataset)

4.2 Performance Analysis of CIFAR-10 Dataset

Using the CIFAR-10 dataset, which consists of ten classes of ten thousand testing pictures and fifty thousand training images. We give tests that were taught and graded on the training set. We shall use basic designs like the ones below on purpose because our focus is on the behaviour of extremely deep networks rather than on cutting-edge outcomes.

The form is present in the simple/residual structures. 3232 images that have had the per-pixel mean removed are the network inputs. There are 33 convolutions in the top layer. Then, we utilise a stack of 6n layers with 33 convolutions, with 2n layers for each of the 32, 16, and 8 feature map sizes.. There are accordingly 16 filters, 32 filters, and 64 filters. Stride of two convolutions is used for subsampling. Softmax, a 10-way fully linked layer, and global average pooling round out the network. There are 6n+2 stacked weighted layers in total. The table below provides a summary of the architecture:

output map size	32×32	16×16	8×8
# layers	1+2n	2n	2n
# filters	16	32	64

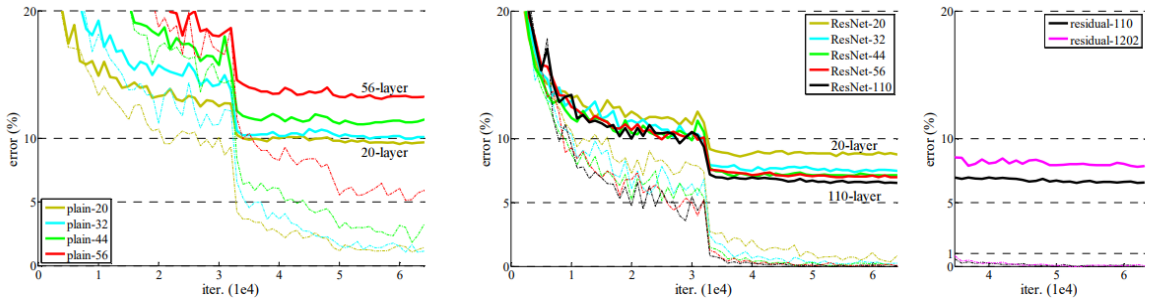
When utilised, shortcut connections are connected to the 33 layer pairs (totally 3n shortcuts). We always use identity reductions on this dataset, and the method error (%) is determined. Data augmentation is a component of all strategies. We do five runs of ResNet-110 and then present "best (mean std)" as in. As a result, our residual models have the same depth, breadth, and number of parameters as their simpler counterparts. We employ weight initialization in and BN with no dropout, 0.0001 weight decay, and 0.9 momentum. Two GPUs and a 128-piece minibatch are used to train these models. After 32k, 48k, and 64k iterations, we split the initial learning rate of 0.1 by 10 and cease training. It is based on a 45k/5k train/val split. We employ the straightforward data augmentation for training: A 3232 cut is created at random from the padded image or its horizontal flip after 4 pixels are added to either side of the image. During testing, we simply examine a single angle of the original 3232 image. We compare networks with 20, 32, 44, and 56 layers for n = 3, 5, 7, and 9. As depth grows, deep plain nets suffer from increasing depth and more training error.

method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Graph 3: error analysis of resnet and other Networks

The standard deviations (std) of the layer responses are shown in the Analysis of Layer Responses. After BN and before additional nonlinearity (ReLU/addition), the answers are the outputs of each of the 33 layers. This research demonstrates the reaction strength of the residual functions in ResNets. The graph below illustrates that ResNets have smaller replies than their simple equivalents. These findings corroborate our core hypothesis that residual functions are often closer to zero than non-residual functions.

We also discover that the deeper ResNet has smaller response magnitudes, as seen by comparisons of ResNet-20, 56, and 110 in. When there are more layers of ResNets, each layer alters the signal less.



Graph 4: Performance Graph

(Source-Convolutional Neural Network solution for MNISTDataset by Majid Vafadar)

4.3 Performance analysis of VOC 2007

In PASCAL VOC 2007, we first compare YOLO to several real-time detection algorithms. In order to understand the differences between YOLO and R-CNN variations, we evaluate the mistakes caused by YOLO and Fast R-CNN, one of the top performing R-CNN variants. Based on the various error profiles, we show that YOLO may be utilised to rescore Fast R-CNN detections and decrease mistakes from background false positives, resulting in a considerable performance gain. We also present VOC 2012 findings and compare mAP to current state-of-the-art techniques. Finally, we show that YOLO generalises to new domains better than other detectors using two artwork datasets.

Their GPU DPM solution, which operates at 30Hz or 100Hz, is contrasted with YOLO.

We examine the accuracy-performance tradeoffs in object identification systems by examining their relative mAP and speed, whereas the other attempts do not achieve real-time. As far as we know, Fast YOLO is the world's fastest object detector and the quickest object detection method on PASCAL. It is more than twice as accurate as prior real-time detection efforts, with a mAP of 52.7%. In addition to maintaining real-time performance, YOLO increases mAP to 63.4%. We also use VGG-16 to teach YOLO. Although it is slower than YOLO, this model is

more accurate. Although it is slower than real-time, it is still useful when compared to other VGG-16-based detection systems.

Even while the fastest DPM may be used without significantly reducing mAP, it still lags below real-time performance by a factor of two. Its low detection accuracy in comparison to neural network systems is another constraint. Selective Search is replaced by static bounding box suggestions in R-CNN without R. Despite being substantially quicker than R-CNN, it is still not real-time and suffers greatly from the lack of suitable ideas. Fast R-CNN accelerates the classification phase of R-CNN, although it still depends on selective search, which may yield bounding box suggestions after around two seconds per image. So, despite having a large mAP, the 0.5 fps frame rate is still far from realtime. Similar to Szegedy et al., the new Faster R-CNN proposes bounding boxes by replacing selective search with a neural network. In our experiments, their most accurate model reaches 7 fps while a smaller, less accurate one runs at 18 fps. R-CNN VGG-16, which is faster, is 6 times slower than YOLO but has 10 mAP more. Even though it is just 2.5 times slower than the ZeilerFergus Faster R-CNN.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Graph 5: Performance Analysis of Different techniques

VOC 2007 Error Analysis:

We study a deep breakdown of VOC 2007 data to further investigate the discrepancies between YOLO and cutting-edge detectors. We compare YOLO to Fast R-CNN since it is one of PASCAL's top detectors and makes its detections public. Hoiem et al.'s methodology and resources are used.

At test time, for each category, we look at the top N guesses for that category. Each prediction is either right or incorrect, depending on the sort of error:

• Correct: IOU > .5

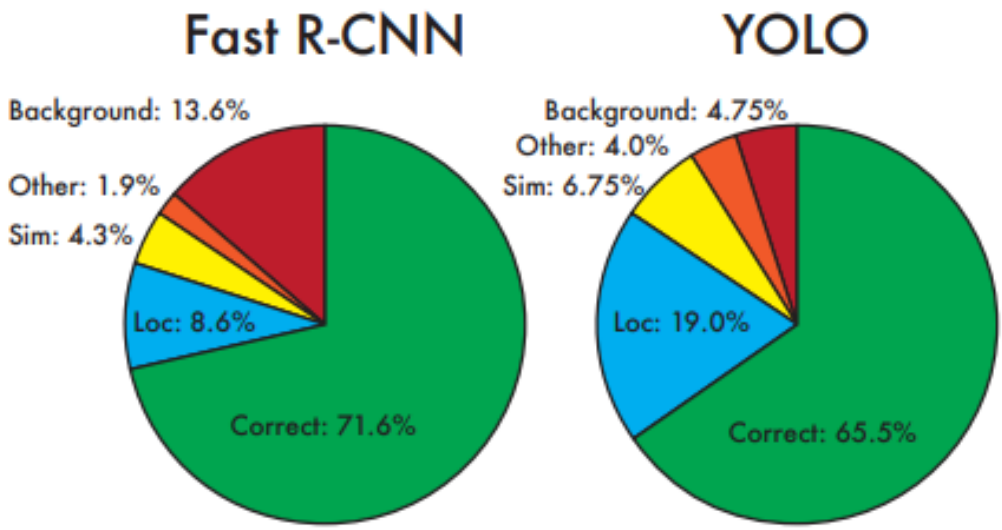
• Localization: 1 < IOU < .5

• Similar: IOU > .1

The proportion of localization and background errors in the top N detections for each category (N = number of items in that category) is shown in the graphs below.

• Other: class is wrong, IOU > .1

• Background: IOU < .1 for any object.



Graph 6: Error analysis

	VOC 2007 AP
YOLO	**59.2**
R-CNN	54.2
DPM	43.2
Poselets [2]	36.5
D&T [4]	-

Graph 7: Quantitative results on the VOC 2007

CHAPTER 5
CONCLUSIONS

5.1 Conclusion

In this work, we focused on object recognition and identification using Yolo-based models. Because traditional detection algorithms produce subpar results, we included the saliency detection model, which aids in identifying the image's main region, to increase detection precision. The bounding box regression model is used to reduce detection error in the following stage, which also takes use of the proposal generating framework, and the RoI pooling loss function is developed to train the recognition parameters. The planned plan's efficacy is compared to existing practises.. The comparative analysis demonstrates a considerable improvement in the identification and detection effectiveness. Furthermore, we examined the proposed methodology's capacity to detect disguised objects against state-of-the-art technologies and discovered that it performed well.

We recommended improving the underwater picture. CNN was trained on the generated dataset to encourage the development of deep learning-based underwater image enhancement systems. The experimental findings show that the suggested CNN model outperforms state-of-the-art approaches and that the created dataset may be used to train CNNs.

5.2 Future Work

In future investigations, we will broaden the dataset to include more difficult underwater images and videos. Furthermore, we will attempt to construct a range map estimate network. The 1100 underwater photographs with range maps supplied in] might be used to train the range map estimate network. To increase the performance of the underwater picture enhancement network, we will use such critical prior knowledge in conjunction with anticipated range maps.

In addition, we discuss some intriguing research avenues in the field of Camouflaged object detection, such as Multi-modal Camouflaged object detection, Co-Camouflage detection, and

Video Camouflaged object detection. The specifics of each are covered in the following sections.

- Multi-Modal COD:

  According to the type of object in the image, different chromaticity values can be seen in thermal maps. As a result, moving the temperature map to the COD makes it easier to tell the target apart from its surroundings. The thermal map-based camouflage object detection may be utilised to realistically separate disguised people from complex backgrounds. Additionally, there is no data in the RGB-T dataset that is open to the public. The RGB-DT COD field has to be filled with additional datasets as a consequence in order to increase detection accuracy even more.

- Co-Camouflage Detection:

  Given how challenging it is to accurately identify the target from a single picture, camouflaged object identification is challenging. On the other hand, many images of the same category of objects in various settings may provide crucial feature information for differentiating objects in a complex environment. Greater colour contrast in the image with strong illumination makes it easier to spot the item. Contrarily, the image in low-light conditions has less shadows, indicating less interference; as a result, the two factors together may help the photographer avoid impediments and highlight the intended area. Co-Camouflage detection may be viewed as a fresh research direction that improves detection precision as a result.

- Video COD:

  So far, camouflaged object recognition based on single images has shown promising results. There has, however, been no model described that can distinguish targets from continually input camouflage pictures (i.e., camouflage films). Furthermore, in the camouflage video, the movement difference between inter-frame images assists in the deconstruction of camouflage and boosts detection efficacy. Furthermore, Video COD may be utilised as a pre-process for a number of visual tasks such as video surveillance, tracking of disguised objects, and so on. It is worth noting that there are currently no publicly accessible databases for the aforementioned jobs, so creating the matching dataset is also a research trend.

References

[1].A. Tankus and Y. Yeshurun, "Detection of regions of interest and camouflage breaking by direct convexity estimation", Proc. Workshop Vis. Surveill., pp. 42-48, 1998.

[2].J. Zhang, Y. Lv, M. Xiang, A. Li, Y. Dai and Y. Zhong, "Depth-guided camouflaged object detection" in arXiv:2106.13217, 2021.

[3].R. Shigematsu, D. Feng, S. You and N. Barnes, "Learning RGB-D salient object detection using background enclosure depth contrast and top-down features", Proc. IEEE Int. Conf. Comput. Vis., pp. 2749-2757, Oct. 2017.

[4].D.-P. Fan, M.-M. Cheng, J.-J. Liu, S.-H. Gao, Q. Hou and A. Borji, "Salient objects in clutter: Bringing salient object detection to the foreground", Proc. Eur. Conf. Comput. Vis. (ECCV), pp. 186-202, 2018.

[5].Y. Piao, W. Ji, J. Li, M. Zhang and H. Lu, "Depth-induced multi-scale recurrent attention network for saliency detection", Proc. Int. Conf. Comput. Vis. (ICCV), pp. 7254-7263, 2019.

[6].J.-X. Zhao, J.-J. Liu, D.-P. Fan, Y. Cao, J. Yang and M.-M. Cheng, "EGNet: Edge guidance network for salient object detection", Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 8779-8788, Oct. 2019.

[7].M. Zhuge et al., "Salient object detection via integrity learning" in arXiv:2101.07663, 2021.

[8].J.-X. Zhao, Y. Cao, D.-P. Fan, M.-M. Cheng, X.-Y. Li and L. Zhang, "Contrast prior and fluid pyramid integration for RGBD salient object detection", Proc. Comput. Vis. Pattern Recognit. (CVPR), pp. 3927-3936, Jun. 2019.

[9].D.-P. Fan, Y. Zhai, A. Borji, J. Yang and L. Shao, "BBS-Net: RGB-D salient object detection with a bifurcated backbone strategy network", Proc. Eur. Conf. Comput. Vis. (ECCV), pp. 275-292, 2020.

[10].K. Fu, D.-P. Fan, G.-P. Ji and Q. Zhao, "JL-DCF: Joint learning and densely-cooperative fusion framework for RGB-D salient object detection", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 3052-3062, Jun. 2020.

[11].Y. Zhai et al., "Bifurcated backbone strategy for RGB-D salient object detection" in arXiv:2007.02713, 2020.

[12].K. Fu, D.-P. Fan, G.-P. Ji, Q. Zhao, J. Shen and C. Zhu, "Siamese network for RGB-D salient object detection and beyond", IEEE Trans. Pattern Anal. Mach. Intell., Apr. 2021.

[13].Y. Ge, C. Zhang, K. Wang, Z. Liu and H. Bi, "WGI-Net: A weighted group integration network for RGB-D salient object detection", Comput. Vis. Media, vol. 7, no. 1, pp. 115-125, Mar. 2021.

[14].J. Wang, K. Song, Y. Bao, L. Huang and Y. Yan, "CGFNet: Cross-guided fusion network for RGB-T salient object detection", IEEE Trans. Circuits Syst. Video Technol., Jul. 2021.

15.W. Zhou, Q. Guo, J. Lei, L. Yu and J.-N. Hwang, "ECFFNet: Effective and consistent feature fusion network for RGB-T salient object detection", IEEE Trans. Circuits Syst. Video Technol., Mar. 2021.

[16].F. Huo, X. Zhu, L. Zhang, Q. Liu and Y. Shu, "Efficient context-guided stacked refinement network for RGB-T salient object detection", IEEE Trans. Circuits Syst. Video Technol., May 2021.

[17].D. Fan, W. Wang, M. Cheng and J. Shen, "Shifting more attention to video salient object detection", Poc. Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 8554-8564, Jun. 2019.

[18].H. Bi, D. Lu, N. Li, L. Yang and H. Guan, "Multi-level model for video saliency detection", Proc. IEEE Int. Conf. Image Process. (ICIP), pp. 4654-4658, Sep. 2019.

[19].H. Fu, X. Cao and Z. Tu, "Cluster-based co-saliency detection", IEEE Trans. Image Process., vol. 22, no. 10, pp. 3766-3778, Oct. 2013.

[20].K. Zhang, T. Li, S. Shen, B. Liu, J. Chen and Q. Liu, "Adaptive graph convolutional network with attention graph clustering for co-saliency detection", Proc. Comput. Vis. Pattern Recognit. (CVPR), pp. 9050-9059, Jun. 2020.

[21].D.-P. Fan, Z. Lin, G.-P. Ji, D. Zhang, H. Fu and M.-M. Cheng, "Taking a deeper look at co-salient object detection", Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 2919-2929, Jun. 2020.

[22].D.-P. Fan et al., "Re-thinking co-salient object detection", IEEE Trans. Pattern Anal. Mach. Intell., Feb. 2021.

[23].Q. Fan, D.-P. Fan, H. Fu, C.-K. Tang, L. Shao and Y.-W. Tai, "Group collaborative learning for co-salient object detection", Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 12288-12298, Jun. 2021.

[24].K. Fu, Y. Jiang, G.-P. Ji, T. Zhou, Q. Zhao and D.-P. Fan, "Light field salient object detection: A review and benchmark" in arXiv:2010.04968, 2020.

[25].C. P. Rao, A. G. Reddy and C. R. Rao, "Camouflaged object detection for machine vision applications", Int. J. Speech Technol., vol. 4, pp. 1-9, Mar. 2020.

[26].D.-P. Fan, G.-P. Ji, G. Sun, M.-M. Cheng, J. Shen and L. Shao, "Camouflaged object detection", Proc. Comput. Vis. Pattern Recognit. (CVPR), pp. 2777-2787, 2020.

Appendix

MNIST Dataset:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow import keras
import seaborn as sns
```

```
# loading dataset
dataset = tf.keras.datasets.mnist

# splitting dataset into training and testing sets
(X_train, y_train), (X_test, y_test) = dataset.load_data()
```

```
X_train_flattened = X_train.reshape(len(X_train),28*28)
X_train_flattened.shape
```

```
# scaling training data for better accuracy
X_train = X_train / 255
# scaling testing data for better accuracy
X_test = X_test / 255
```

```
# creating model object
model = Sequential()
# adding only the output layer with sigmoid activation
model.add(Dense(120, activation='relu'))
model.add(Dense(190, activation='relu'))
model.add(Dense(10, activation='sigmoid'))
# defining optimizers, loss functions and metrics
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics='accuracy')
# testing model accuracy
model.fit(X_train_flattened, y_train, epochs=10,validation_split=0.2)
model.save('majorproject.h5')
```

```
y_predicted_labels = [np.argmax(i) for i in y_predicted]
y_predicted_labels[:10]
```

```
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_predicted_labels)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

CIFAR-10 Dataset:

```python
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
```

Python

```python
(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
```

Python

```python
X_train.shape
```

Python

```
(50000, 32, 32, 3)
```

```python
X_test.shape
```

Python

```
(10000, 32, 32, 3)
```

```python
y_train[:5]
```

```
array([[6],
       [9],
       [9],
       [4],
       [1]], dtype=uint8)
```

```python
y_train = y_train.reshape(-1,) # -1 to keep 10000 as it is, and blank to flatten one-sized list
y_train[:5]
```

```
array([6, 9, 9, 4, 1], dtype=uint8)
```

```python
y_test = y_test.reshape(-1,) # -1 to keep 10000 as it is, and blank to flatten one-sized list
y_test[:5]
```

```
array([3, 8, 8, 0, 6], dtype=uint8)
```

```python
classes = ['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']
```

```python
plot_sample(X_train,y_train,0)
plot_sample(X_train,y_train,1)
plot_sample(X_train,y_train,2)
```

```
X_train = X_train / 255
X_test = X_test / 255
```

```
ann = models.Sequential([
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(3000,activation='relu'),
    layers.Dense(1000,activation='relu'),
    layers.Dense(10,activation='sigmoid')
])

ann.compile(optimizer='SGD',
loss='sparse_categorical_crossentropy',
metrics='accuracy')

ann.fit(X_train, y_train, epochs=5)
```

```
cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu',input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu',input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Flatten(),
    layers.Dense(50,activation='relu'),
    layers.Dense(10,activation='softmax')
])
```

```
cnn.compile(optimizer='SGD',
loss='sparse_categorical_crossentropy',
metrics='accuracy')
```

```
cnn.fit(X_train, y_train, epochs=5)
```

```
y_predicted = cnn.predict(X_test)
y_predicted_classes = [np.argmax(i) for i in y_predicted]

print('classification Report: \n', classification_report(y_test,y_predicted_classes))
```

```
classification Report:
              precision    recall  f1-score   support

           0       0.59      0.62      0.60      1000
           1       0.59      0.75      0.66      1000
           2       0.52      0.29      0.37      1000
           3       0.45      0.25      0.32      1000
           4       0.44      0.50      0.47      1000
           5       0.59      0.33      0.42      1000
           6       0.46      0.81      0.59      1000
           7       0.74      0.53      0.62      1000
           8       0.50      0.81      0.62      1000
           9       0.61      0.47      0.53      1000

    accuracy                           0.54     10000
   macro avg       0.55      0.54      0.52     10000
weighted avg       0.55      0.54      0.52     10000
```

```
    for i in range(10):
        print('Actual:',classes[y_test[i]],' Predicted:',classes[y_predicted_classes[i]])
```

```
Actual: cat  Predicted: cat
Actual: ship  Predicted: ship
Actual: ship  Predicted: ship
Actual: airplane  Predicted: airplane
Actual: frog  Predicted: frog
Actual: frog  Predicted: frog
Actual: automobile  Predicted: automobile
Actual: frog  Predicted: frog
Actual: cat  Predicted: deer
Actual: automobile  Predicted: automobile
```

Model:

```
from imageai.Detection import ObjectDetection

dataset="./dataset_collection/"

recognizer = ObjectDetection()

path_model = "./Models/yolo-tiny.h5"

path_input = "./Input/images.jpg"

path_output = "./Output/newimage.jpg"

recognizer.setModelTypeAsTinyYOLOv3()
```

```
recognizer.setModelPath(path_model)

recognizer.loadModel()

recognition = recognizer.detectObjectsFromImage(

    input_image = path_input,

    output_image_path = path_output

    )

for eachItem in recognition:

    print(eachItem["name"] , " : ", eachItem["percentage_probability"])
```

Outputs:

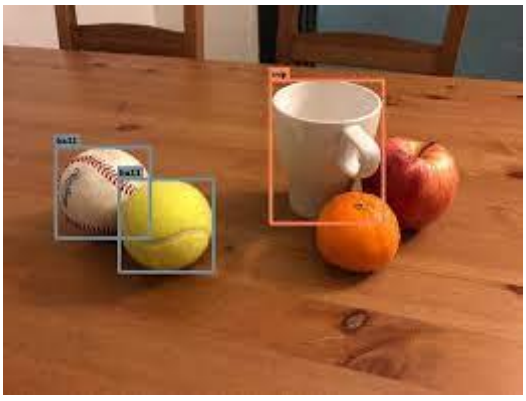


Figure: Multiple Object Detecting