**Chapter 1**

**ABSTRACT**

The task of finding objects belonging to classes of interest in images has long been a focus of Computer Vision research. The ability to localize objects is useful in many applications: from self-driving cars, where it allows the car to detect pedestrians, bicyclists, road signs, and other vehicles, to security, where intruding persons can be detected. Though a lot of progress has been made since the conception of the field of Computer Vision more than five decades ago, as always, there is scope for further improvement. This is especially true in the case of object detection where a myriad of factors including variation in object instances through pose and appearance, along with other environmental factors such as the degree of occlusion, and lighting tend to cause failures. In this project, we focus on improving object detection through the use of more representative features and better models. We have introduced a model named YOLO throughout our project. YOLO stands for you look only once which is a pre-trained model. We propose new features that are not only more powerful but also more robust and capture more information than currently popular features. Further, we have also demonstrated real-

**Chapter 2**

**Introduction to Object detection**

The programme started with scratch and in the first week of the training we were introduced with the Linux platform, learned the basic commands and the basics of python programming. In the second and third week of the training programme we worked upon the advanced concepts of python and were introduced with Machine learning. The training concluded with a project making session in which we were assisted with trainees and mentors.

Since their evolution, humans have been using many types of tools to accomplish various tasks. The creativity of the human brain led to the invention of different machines. These machines made the human life easy by enabling people to meet various life needs, including travelling, industries, constructions, and computing. Despite rapid developments in the machine industry, intelligence has remained the fundamental difference between humans and machines in performing their tasks. A human uses his or her senses to gather information from the surrounding atmosphere; the human brain works to analyse that information and takes suitable decisions accordingly. Machines, in contrast, are not intelligent by nature. A machine does not have the ability to analyse data and take decisions. For example, a machine is not expected to understand the story of Harry Potter, jump over a hole in the street, or interact with other machines through a common language.

Most of the IT products incorporate Artificial Intelligence (AI), Machine Learning (ML), Deep Leaning (DL) techniques and thus final products become smarter. Nowadays, IT industries frequently uses python programming language for application of AI, ML etc based techniques. Due to the advancement of technology where the machines are used for various tasks such as detecting, identifying and classifying objects in images, videos and real-time. Our project, “Object Detection” deals with detecting objects in images and videos.

This project of **Object Detection** is based on a pre-trained model You Look Only Once which is abbreviated as YOLO. It is a convolutional neural network-based model that detects objects in real time using the “You only Look Once” framework. It stands out from its competitors because, as the name indicates it only needs to see each image once. Currently there are 3 main implementations of YOLO, named as Darknet, AlexyAB/darknet, Darkflow. This project uses Darkflow implementation which is limited to classes contained in the datasets used to obtain this weight.

**Chapter – 3**

**Models**

**3.1 TensorFlow**

**3.2 YOLO (You Only Look Once)**

**3.3 OpenCV**

**3.4 NumPy**

**3.5 Matplotlib**

**CHAPTER-4**

**IMPLEMENTATION DETAILS**

**4.1 Hardware Requirements**

Device: Laptop, Smart Phones or Desktop Computer

Processor: Pentium IV (minimum) and above

RAM: 256MB (minimum) and above

Hard disk: 500MB (minimum) and above

**4.2 Software Requirements**

Operating System: Windows, Linux – Ubuntu

Platforms: Jupyter, Google Collab, Anaconda prompt, Virtual Box

Languages: Python

Web browsers: Chrome, Firefox

**Google Colab**

Google Colab is a free cloud service and now it supports free GPU! You can:

* Improve your Python programming language coding skills.
* Develop deep learning applications using popular libraries such as Keras,  
  TensorFlow, PyTorch, and OpenCV.

As the name suggests, Google Colabcomes with collaboration backed in the product. In fact, it is a Jupyter notebook that leverages Google Docs collaboration features. It also runs on Google servers and you don’t need to install anything. Moreover, the notebooks are savedto your Google Drive account.

**4.3** **Language**

**Python**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

**4.4 ARTIFICIAL INTELLIGENCE**

Machine learning is a subset of Artificial Intelligence (AI), and it is a broad term that aims at using data to offer solutions to existing problems. It is the science and engineering of replicating, even surpassing human level intelligence in machines. That means observe or read, learn, sense, and experience. The AI process loop is as follows:

Observe->plan->optimize->action->learn and adapt

The AI process loop discussed above can be achieved using intelligent agents. A robotic intelligent agent can be defined as a component that can perceive its environment through different kinds of sensors (camera, infrared, etc.), and will take actions within the environment through efforts. Here robotic agents are designed to reflect humans.

**4.5 MACHINE LEARNING**

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Algorithms are a sequence of instructions used to solve a problem. Algorithms, developed by programmers to instruct computers in new tasks, are the building blocks of the advanced digital world we see today. Computer algorithms organize enormous amounts of data into information and services, based on certain instructions and rules. It’s an important concept to understand, because in machine learning, learning algorithms – not computer programmers – create the rules.

Instead of programming the computer every step of the way, this approach gives the computer instructions that allow it to learn from data without new step-by-step instructions by the programmer. This means computers can be used for new, complicated tasks that could not be manually programmed. Things like photo recognition applications for the visually impaired, or translating pictures into speech.

The basic process of machine learning is to give training data to a learning algorithm. The learning algorithm then generates a new set of rules, based on inferences from the data. This is in essence generating a new algorithm, formally referred to as the machine learning model. By using different training data, the same learning algorithm could be used to generate different models. For example, the same type of learning algorithm could be used to teach the computer how to translate languages or predict the stock market.

Inferring new instructions from data is the core strength of machine learning. It also highlights the critical role of data: the more data available to train the algorithm, the more it learns. In fact, many recent advances in AI have not been due to radical innovations in learning algorithms, but rather by the enormous amount of data enabled by the Internet.

**4.6 DEEP LEARNING**

Deep learning has been the buzzword in the machine learning world in recent times. The main objective of the deep learning algorithm so far has been to use machine learning to achieve Artificial General Intelligence (AGI), that is, replicate human-level intelligence in machines to solve any problems for a given area. Deep learning has shown promising outcomes in computer vision, audio processing, and text mining. The advancements in this area has led to a breakthrough such as self-driving cars.

There has been a number of powerful and popular open source libraries built in the last few years predominantly focused on deep learning. They are:

TensorFlow: As per the official documentation, it is a library for numerical computation using data flow graphs for scalable machine learning developed by Google researchers. It is currently being used by Google products for research and production. It was open sourced in 2015 and has gained wide popularity in the machine learning world.

Pylearn2: A Machine Learning library based on Theano, which means users can write new models/algorithms using mathematical expressions and Theano will optimize, stabilize, and compile those expressions.

Keras: It is known as a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It’s an interface rather than an end-end machine learning framework. It’s written in Python, simple to get started, highly module, and easy yet deep enough to expand to build/support complex models.

**4.6 ARTIFICAL NEURAL NETWORK**

Our brain is made up of a cluster of small connected units called neurons, which send electrical signals to one another. The long-term knowledge is represented by the strength of the connections between neurons. When we see objects, light travels through the retina and the visual information gets converted to electrical signals, and further on the electric signal passes through the hierarchy of connected neurons of different regions within the brain in a few milliseconds to decode signals/information.

Biological neurons have dendrites to receive signals, a cell body to process them, and an axon/axon terminal to transfer signals out to other neurons. Similarly an artificial neuron has multiple input channels to accept training samples represented as a vector, and a processing stage where the weights(w) are adjusted such that the output error (actual vs. predicted) is minimized. Then the result is fed into an activation function to produce output, for example, a classification label. The activation function for a classification problem is a threshold cutoff (standard is .5) above which class is 1 else 0.

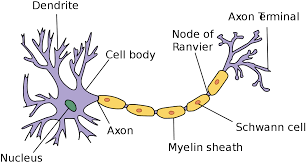
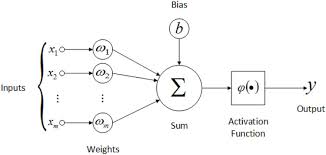
 

Figure 4.1 Biological vs Artificial neuron

The same approach of learning is followed in machines also.

In computers an image is represented as one large three-dimensional array of numbers. For example, consider Figure 6-2; it is the handwritten digit image of gray scale 28x28x1 (width x height x depth) size resulting in 784 data points. Each number in the array is an integer that ranges from 0 (black) to 255(white). In a typical classification problem the model has to turn this large matrix into a single label. For a color image additionally it will have three color channels: Red, Green, Blue (RGB) for each pixel, so the same image in color would be of size 28x28x3 = 2352 data points.

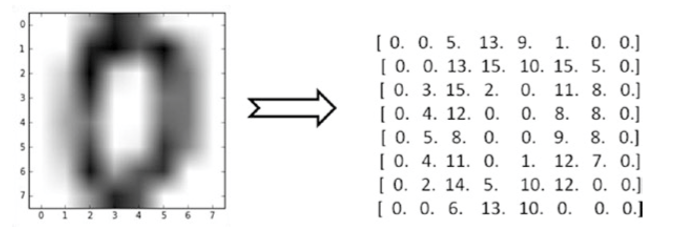


Figure 4.2 Handwritten digit(zero) image and corresponding array

Image classification can be challenging for a computer as there are a variety of challenges associated with representation of the images. A simple classification model might not be able to address most of these issues without a lot of feature engineering effort. Some of the key issues are:

* 1. View point variation: Same object can have different orientation.
  2. Scale and illumination variation: Variation in object’s size and the level of illumination on pixel level can vary.
  3. Deformation/twist and intra-class variation: Non-rigid bodies can be deformed in great ways and there can be different types of objects with varying appearance within a class. Blockage: Only small portion of object in interest can be visible.
  4. Background clutter: Objects can blend into their environment, which will make it hard to identify.

To address the drawback of single perceptrons, multilayer perceptrons were proposed; also, commonly known as a feedforward neural network, it is a composition of multiple perceptrons connected in different ways and operating on distinctive activation functions to enable improved learning mechanisms. The training sample propagates forward through the network and the output error is back propagated and the error is minimized using the gradient descent method, which will calculate a loss function for all the weights in the network.

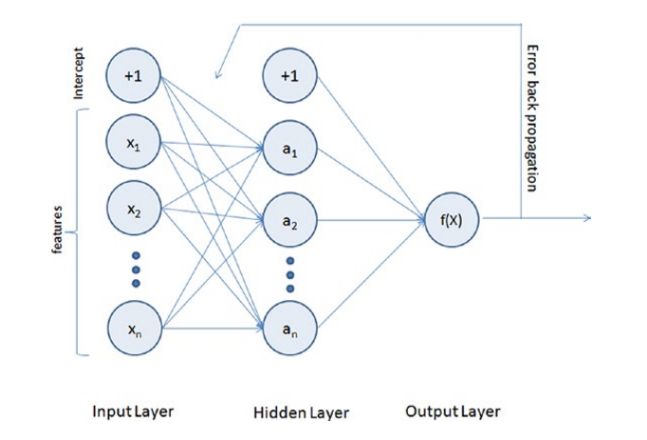


Figure 4.3 Multilayer perceptron representation

The activation function for a simple one-level hidden layer of a multilayer perceptron can be given by:

formula

where xi is the input and Wji (1) is the input layer

weights and Wkj (2) is the weight of hidden layer. A multi-layered neural network can have many hidden layers, where the network holds its internal abstract representation of the training sample. The upper layers will be building new abstractions on top of the previous layers. So, having more hidden layers for a complex dataset will help the neural network to learn better.

From Figure 4.3, the MLP architecture has a minimum of three layers, that is, input, hidden, and output layers. The input layer’s neuron count will be equal to the total number of features and in some libraries an additional neuron for intercept/bias. These neurons are represented as nodes. The output layers will have a single neuron for regression models and binary classifier; otherwise it will be equal to the total number of class labels for multiclass classification models.

Using too few neurons for a complex dataset can result in an under-fitted model due to the fact that it might fail to learn the patterns in complex data. However, using too many neurons can result in an over-fitted model as it has capacity to capture patterns that might be noise or specific for the given training dataset. So, to build an efficient multi-layered neural network, A widely accepted rule of thumb is that you can start with one hidden layer, as there is a theory that one hidden layer is sufficient for the majority of problems. Then, gradually increase the layers on a trial-and-error basis to see if there is any improvement in accuracy. The number of neurons in the hidden layer can ideally be the mean of the neurons in the input and output layers.

**4.7 TYPES OF NEURAL NETWORKS**

Different types of neural networks use different principles in determining their own rules. There are many types of artificial neural networks, each with their unique strengths. Also, they operate in different ways to achieve different outcomes.

1. **Feedforward Neural Network**

This neural network is one of the simplest forms of ANN, where the data or the input travels in one direction. The data passes through the input nodes and exit on the output nodes. This neural network may or may not have the hidden layers. In simple words, it has a front propagated wave and no back propagation by using a classifying activation function usually. In a feedforward neural network, the sum of the products of the inputs and their weights are calculated.

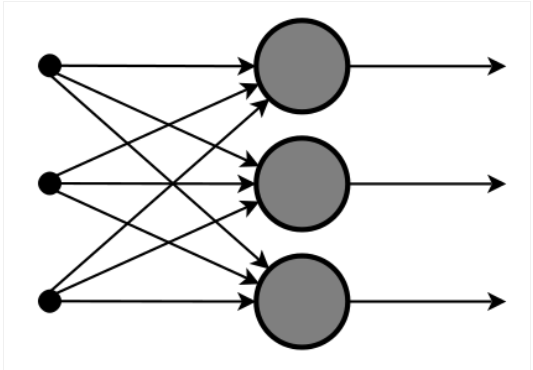


Figure 4.4 Feedforward Neural Network-Artificial Neuron

Here, the sum of the products of inputs and weights are calculated and fed to the output. The output is considered if it is above a certain value i.e. threshold (usually 0) and the neuron fires with an activated output (usually 1) and if it does not fire, the deactivated value is emitted (usually -1).

Feedforward neural networks are used in technologies like face recognition and computer vision. This is because the target classes in these applications are hard to classify.

A simple feedforward neural network is equipped to deal with data which contains a lot of noise. Feedforward neural networks are also relatively simple to maintain.

**2. Radial Basis Function Neural Network**

A radial basis function considers the distance of any point relative to the centre. Such neural networks have two layers. In the inner layer, the features are combined with the radial basis function. Then the output of these features is taken into account when calculating the same output in the next time-step.

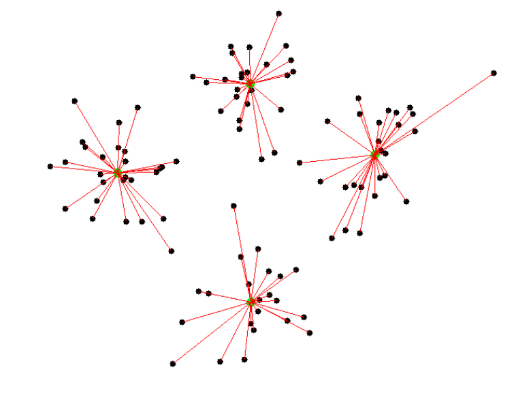


Figure 4.5 Radial Basis Function Neural Network

The above figure represents the distance calculating from the centre to a point in the plane similar to a radius of the circle. Here, the distance measure used in Euclidean, other distance measures can also be used. The model depends on the maximum reach or the radius of the circle in classifying the points into different categories. If the point is in or around the radius, the likelihood of the new point begins classified into that class is high. There can be a transition while changing from one region to another and this can be controlled by the beta function.

The radial basis function neural network is applied extensively in power restoration systems. In recent decades, power systems have become bigger and more complex. This increases the risk of a blackout. This neural network is used in the power restoration systems in order to restore power in the shortest possible time.

**3.** **Multilayer Perceptron**

A multilayer perceptron has three or more layers. It is used to classify data that cannot be separated linearly. It is a type of artificial neural network that is fully connected. This is because every single node in a layer is connected to each node in the following layer.

A multilayer perceptron uses a nonlinear activation function (mainly hyperbolic tangent or logistic function).

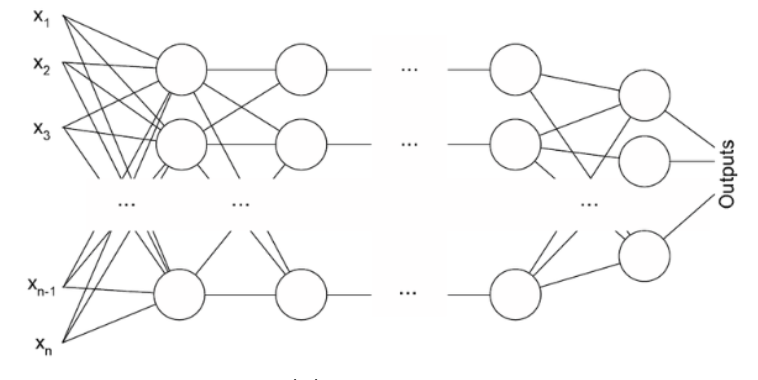


Figure 4.6 Multilayer Perceptron

4.**Convolutional Neural Network**

A convolutional neural network(CNN) uses a variation of the multilayer perceptrons. A CNN contains one or more than one convolutional layers. These layers can either be completely interconnected or pooled. Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters. Due to this ability, convolutional neural networks show very effective results in image and video recognition, natural language processing, and recommender systems. Convolutional neural networks also show great results in semantic parsing and paraphrase detection. They are also applied in signal processing and image classification. CNNs are also being used in image analysis and recognition in agriculture where weather features are extracted from satellites like LSAT to predict the growth and yield of a piece of land.

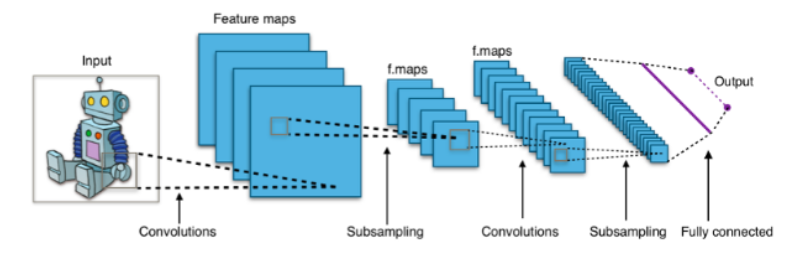


Figure 4.7 Convolutional Neural Network

**5.Recurrent Neural Network (RNN) – Long Short-Term Memory**

A Recurrent Neural Network is a type of artificial neural network in which the output of a particular layer is saved and fed back to the input. This helps predict the outcome of the layer.

The first layer is formed in the same way as it is in the feedforward network. That is, with the product of the sum of the weights and features. However, in subsequent layers, the recurrent neural network process begins.

From each time-step to the next, each node will remember some information that it had in the previous time-step. In other words, each node acts as a memory cell while computing and carrying out operations. The neural network begins with the front propagation as usual but remembers the information it may need to use later.

If the prediction is wrong, the system self-learns and works towards making the right prediction during the backpropagation. This type of neural network is very effective in text-to-speech conversion technology.

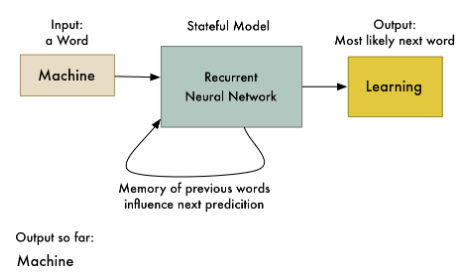


Figure 4.8 Recurrent Neural Network- Long Short Term Memory

**6. Modular Neural Network**

Modular Neural Networks have a collection of different networks working independently and contributing towards the output. Each neural network has a set of inputs which are unique compared to other networks constructing and performing sub-tasks. These networks do not interact or signal each other in accomplishing the tasks. The advantage of a modular neural network is that it breakdowns a large computational process into smaller components decreasing the complexity. This breakdown will help in decreasing the number of connections and negates the interaction of these network with each other, which in turn will increase the computation speed. However, the processing time will depend on the number of neurons and their involvement in computing the results.

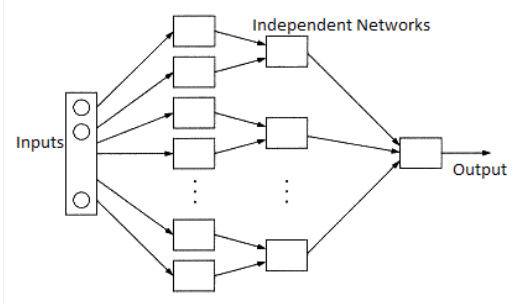


Figure 4.9 Modular Neural Network

**7.Sequence-To-Sequence Models**

A sequence to sequence model consists of two recurrent neural networks. There’s an encoder that processes the input and a decoder that processes the output. The encoder and decoder can either use the same or different parameters. This model is particularly applicable in those cases where the length of the input data is not the same as the length of the output data.

Sequence-to-sequence models are applied mainly in chatbots, machine translation, and question answering systems.

**4.8 Convolutional Neural Network**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

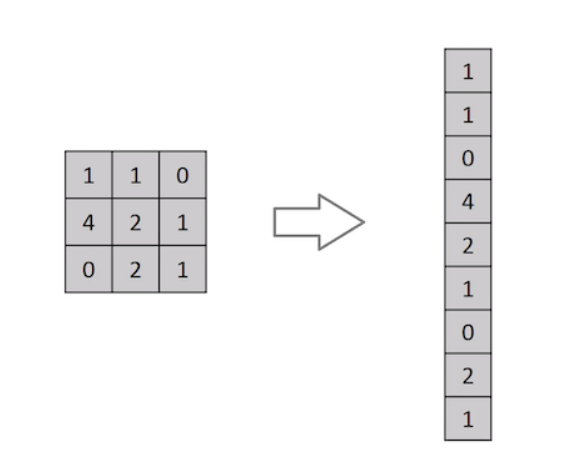


Figure 4.10 Flattening of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values so we can flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

**4.8.1 Input Image**

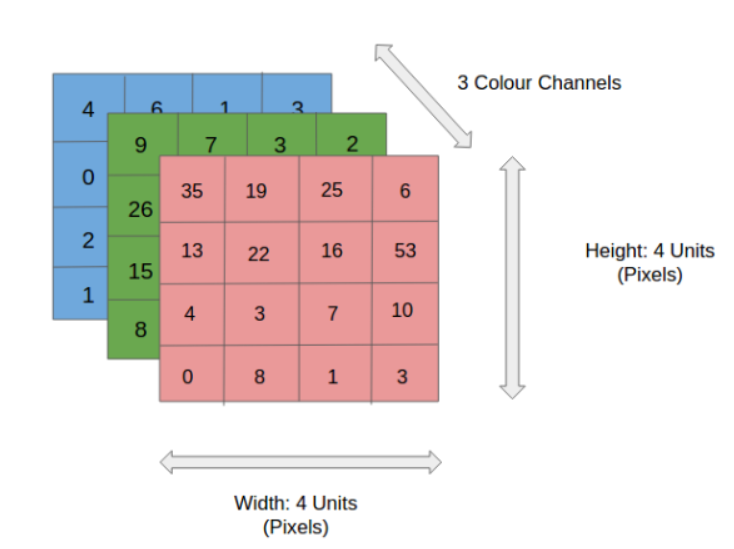


Figure 4.11 4x4x3 RGB Image

In the figure, we have an RGB image which has been separated by its three colour planes — Red, Green, and Blue. There are a number of such colour spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

**4.8.2 Convolution Layer — The Kernel**

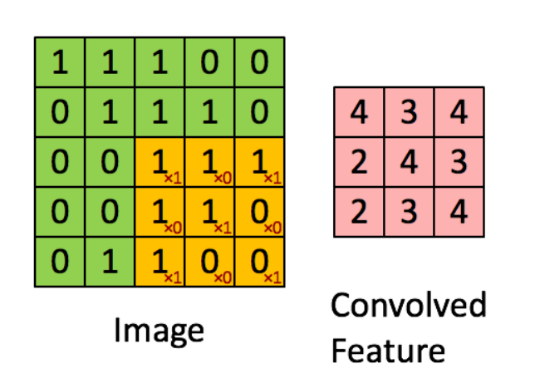


Figure 4.12 Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

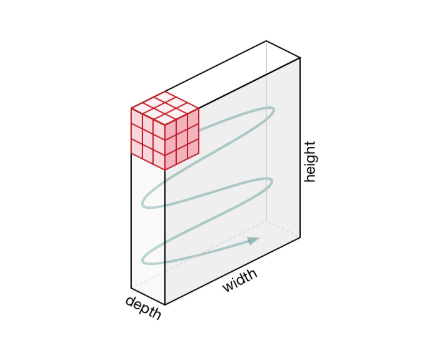


Figure 4.13 Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

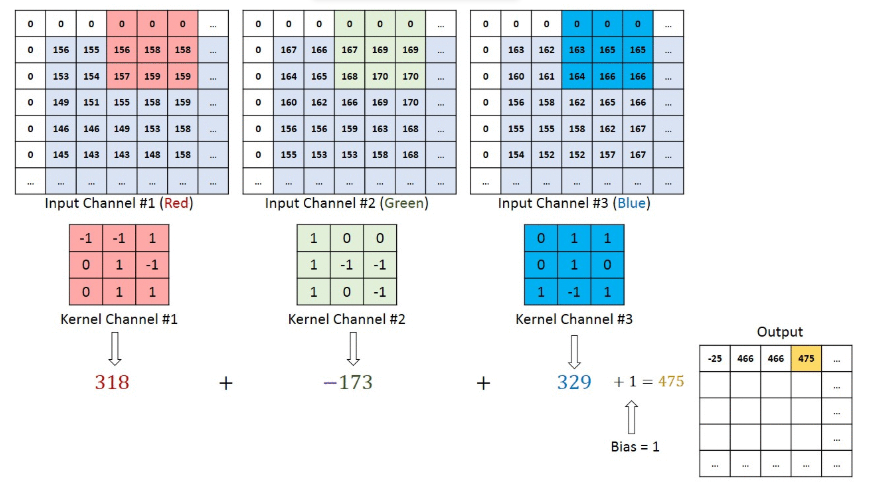


Figure 4.14 Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted

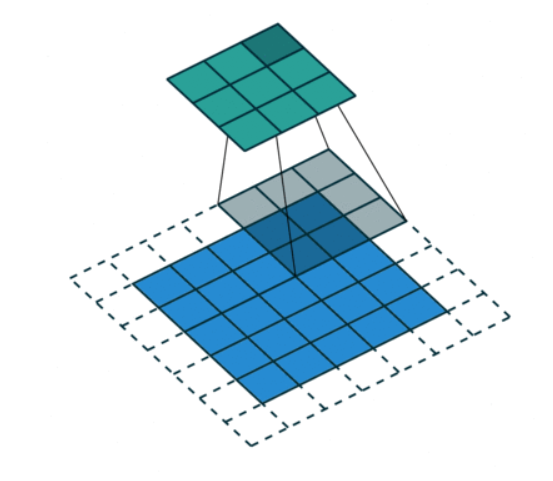


Figure 4.15 Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

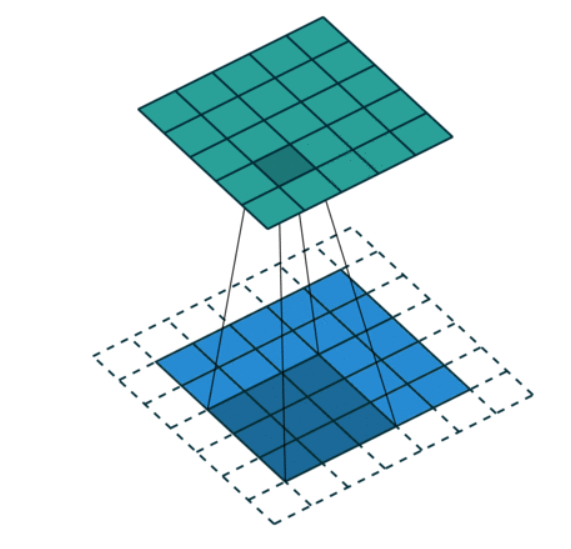


Figure 4.16 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — Same Padding.

On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — Valid Padding.

**4.8.3 Pooling Layer**

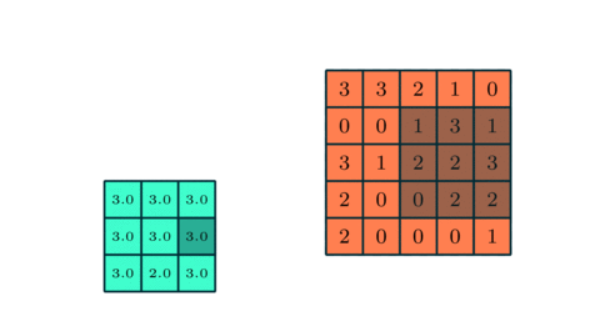


Figure 4.17 3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

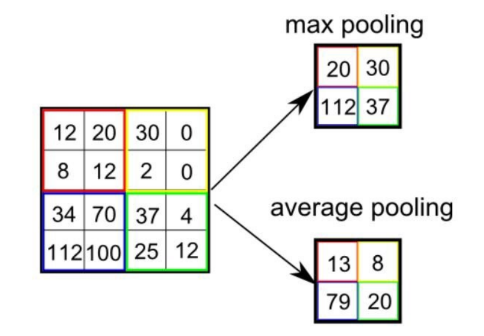


Figure 4.18 Types of pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

**4.8.4 Fully Connected Layer**

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

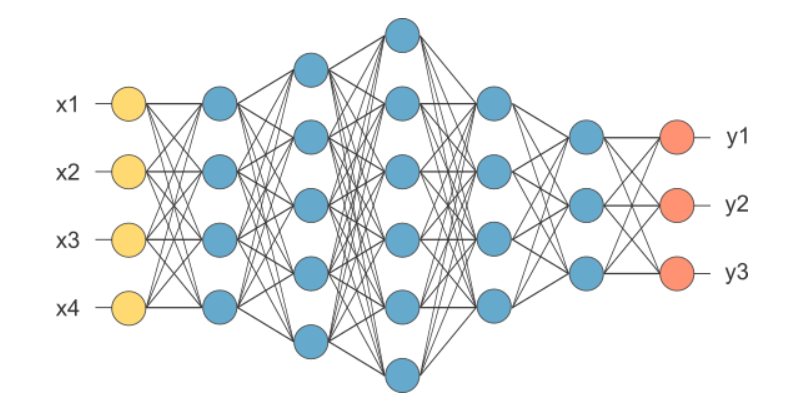


Figure 4.19 After pooling layer, flattened as FC layer

In the above diagram, feature map matrix will be converted as vector (x1, x2, x3, …). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as SoftMax or sigmoid to classify the outputs as cat, dog, car, truck etc.,

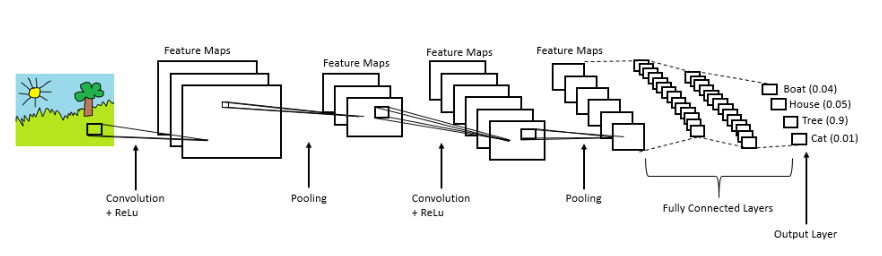


Figure 4.20 Complete CNN architecture

**4.9 YOLO-You Look Only Once**

Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in a image, Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. This the architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since we frame detection as a regression problem we don’t need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means we can process streaming video in real-time with less than 25 milliseconds of latency.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it can’t see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

Our network uses feature from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision.

Our system divides the input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) ∗ IOU . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. Each grid cell also predicts C conditional class probabilities, Pr(Classi |Object). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

Pr(Classi|Object)∗Pr(Object)∗IOU = Pr(Classi)∗IOU

, which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

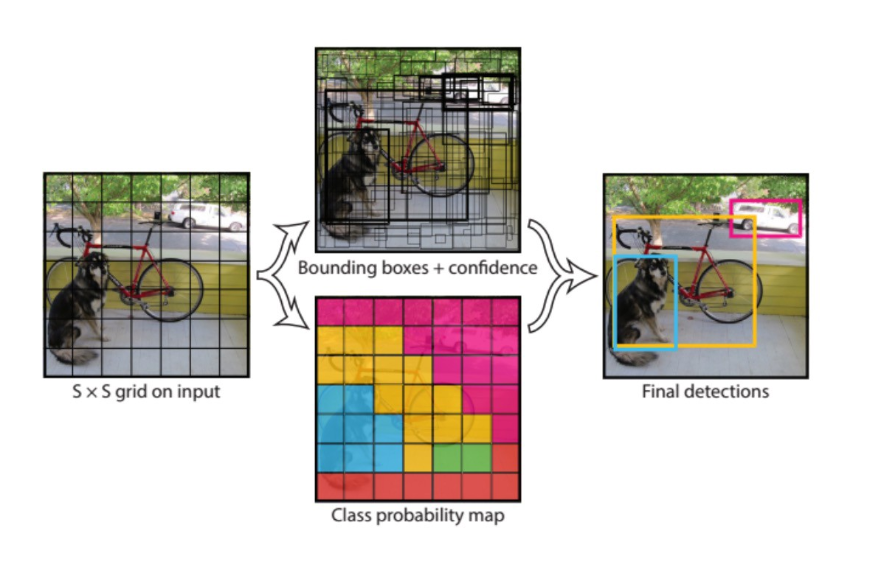


Figure 4.21 The Model. Our system model detection as a regression problem. It divides the image into an S x S grid and for each grid cells predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S x S x ( B \* 5 + C ).

**Chapter -5**

**IMPLEMENTED CODE**

import time

import cv2

from darkflow.net.build import TFNet

from matplotlib import pyplot as plt

import numpy as np

options={

'model': 'cfg/yolo.cfg',

'load': 'bin/yolo.weights',

'threshold':0.3,

'gpu':1.0

}

tfnet = TFNet(options)

colors = [tuple(255 \* np.random.rand(3)) for \_ in range(10)]

capture = cv2.VideoCapture(0)

capture.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 1920)

capture.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 1080)

while True:

stime = time.time()

ret, frame = capture.read()

if ret:

results = tfnet.return\_predict(frame)

for color, result in zip(colors, results):

tl = (result['topleft']['x'], result['topleft']['y'])

br = (result['bottomright']['x'], result['bottomright']['y'])

label = result['label']

confidence = result['confidence']

text = '{}: {:.0f}%'.format(label, confidence \* 100)

frame = cv2.rectangle(frame, tl, br, color, 5)

frame = cv2.putText(

frame, text, tl, cv2.FONT\_HERSHEY\_COMPLEX, 1, (0, 0, 0), 2)

cv2.imshow('frame', frame)

print('FPS {:.1f}'.format(1 / (time.time() - stime)))

if cv2.waitKey(1) & 0xFF == ord('q'):

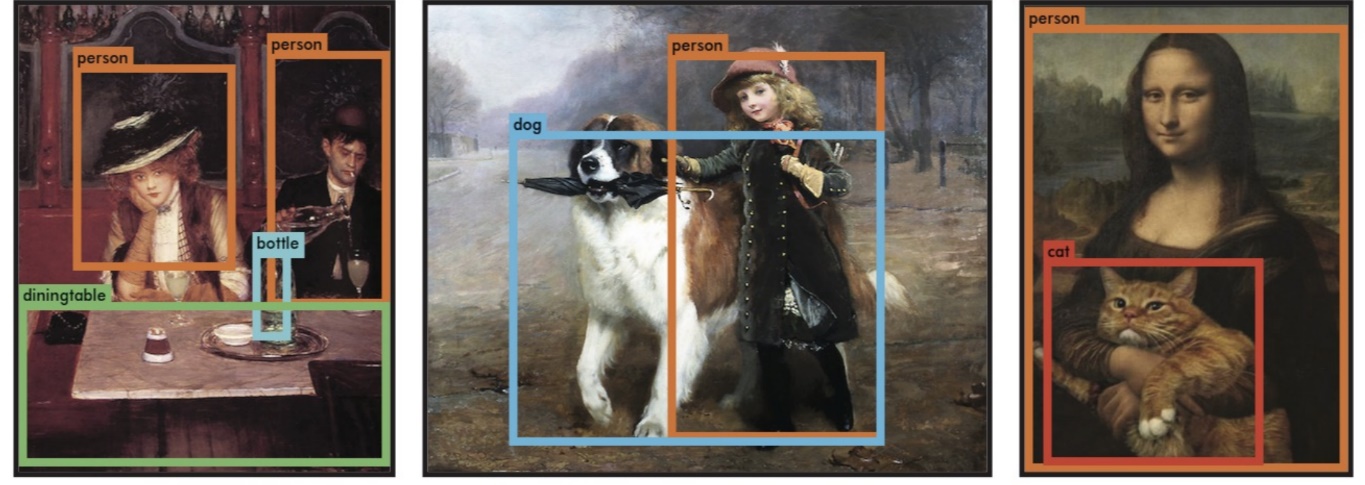
break

capture.release()

cv2.destroyAllWindows()

**CHAPTER 6**

**SCREENSHOTS**



**CHAPTER 6: CONCLUSION**

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. Taking its application into consideration we have created our project titled object detection using YOLO. YOLO (you only look once) is an object detection algorithm that utilizes bounding box regression heads and classification methods. It is a Pre-trained model. The YOLO architecture in simple terms consists of an [math]S×S[/math] grid cells of classifiers and regressors. We have used YOLO V2 in this project as it has greater accuracy and speed. To conclude this section, The Execution speed of the proposed approach is sufficient enough to be used for real-time applications.

**CHAPTER 7: FUTURE SCOPE**

Object detection is breaking into a wide range of industries, with use cases ranging from personal security to productivity in the workplace. Object detection is applied in many areas of computer vision, including image retrieval, security, surveillance, automated vehicle systems and, machine inspection. Significant challenges stay in the field of object recognition. The possibilities are endless when it comes to future use cases for object detection. Some future Applications as follows

•  **OPTICAL CHARACTER RECOGNITION**

Optical character recognition or optical character reader, often abbreviated as OCR, is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or subtitle text superimposed on an image, we are extracting characters from the image or video.

**• SELF DRIVING CARS**

object detection for autonomous driving is in order for a car to decide what to do in next step whether accelerating, apply brakes or turn, it needs to know where all the objects are around the car and what those objects are that requires object detection and we would essentially train the car to detect known set of objects such as cars, pedestrians, traffic lights, road signs, bicycles, motorcycles. It is very beneficial in Safe Driving and could also reduce accident rates.

**BIBLIOGRAPHY**

**BOOKS:**

**WEBSITES**:

1.www.wikepedia.com

2..www.quora.com

3.www.towardsdatascience.com

**KEY TO ABBREVIATIONS**

ConvNet :Convolutional neural network

YOLO: You Only Look Once