

MACHINE LEARNING

(Object Detection)

Summer Internship Report Submitted in partial fulfilment
of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

MOTUPALLY SUNDARA CHARYA

221710302038

Under the guidance of

Mr.

Assistant Professor



Department of Computer science Engineering

GITAM School of Technology

GITAM(Deemed to be University) Hyderabad-502329,July 2020

DECLARATION

I submit this industrial training work entitled “OBJECT DETECTION” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science Engineering**”. I declare that it was carried out independently by me under the guidance of **Mr.** Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

MOTUPALLY SUNDARA CHARYA

Date:

Roll No: 221710302038



GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Date:

CERTIFICATE

This is to certify that the Industrial Training Report entitled “OBJECT DETECTION” is being submitted by MOTUPALLLY SUNDARA CHARYA (221710302038) in partial fulfillment of the requirement for the award of Bachelor of Technology in Computer Science Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2020-21

It is faithful record work carried out by him at the Computer Science Engineering Department, GITAM University Hyderabad Campus under my guidance and supervision.

Mr.

Assistant Professor

Department of CSE

Dr. S . Phani Kumar

Professor and HOD

Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful completion of this internship.

I would like to thank Dr. N. Siva Prasad, Pro Vice Chancellor, GITAM Hyderabad and Dr. N.Seetharamaiah, Principal, GITAM Hyderabad

I would like to thank respected Dr. S.PhaniKumar, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize of what we study for

I would like to thank the respected faculties Mr. who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

MOTUPALLY SUNDARA CHARYA

Roll No:221710302038

ABSTRACT

Computer vision is the branch of the science of computers and software systems which can recognize as well as understand Images and scenes. Computer vision is consists of various aspects such as Image recognition, Object Detection ,Image generation , Image super - resolution and many more . Object detection is widely used for face detection, vehicle detection , Pedestrian counting , web images , security systems and self driving cars . In this project, we are using highly accurate object detection algorithms and methods such as R-CNN, Fast-RCNN, Faster-RCNN, and fast yet highly accurate ones like SSD and YOLO. Using these methods and algorithms ,based on dep learning which is also based on machine learning requires lots of mathematical and deep learning framework understanding by using dependencies such as Tensor Flow , OpenCV , imageai etc , we can detect each and every object in an image by the area object in an highlighted rectangular boxes and identify each and every object and assign its tag to the object . This also includes the accuracy of each method for identifying objects

TABLE OF CONTENTS:

CHAPTER 1:MACHINE LEARNING	1
1. INTRODUCTION	1
1.1 IMPORTANCE OF MACHINE LEARNING.	1
1.2 USES OF MACHINE LEARNING	2
1.3 TYPES OF MACHINE LEARNING	5
1.3.1 Supervised Learning	6
1.3.2 Unsupervised Learning	6
1.3.3 Reinforcement Learning	7
1.4 DEEP LEARNIG AND ITS IMPORTANCE	7
1.5 USES OF DEEP LEARNING	8
1.6 RELATION SHIP BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING	10
CHAPTER 2: PYTHON	12
2.1 INTRODUCTION	12
2.2 SETUP OF PYTHON	12
2.3 INSTALLATION (using python IDLE)	12
2.4 PYTHON INSTLLATION USING ANNACONDA	14
2.5 FEATURES	15
2.6 VARIABLES TYPE	16
2.6.1 Python Number	16
2.6.2 Python Strings	17
2.6.3 Python Lists	17
2.6.4 Python Tuples	17
2.6.5 Python Dictionary	18
2.7 FUNCTION	18
2.7.1 Defining a Function	18
2.7.2 Calling a Function	18

2.8 OOP's CONCEPT	18
2.8.1 Class	18
CHAPTER 3: CASE STUDY	21
3.1 PROBLEM STATEMENT	21
3.2 DATA SET	21
3.3 OBJECTIVE OF THE CASE STUDY	21
CHAPTER 4: MODEL BUILDING	22
4 PREPROCESSING OF THE DATA	22
4.1 GETTING THE DATA	22
4.2 SETTING THE ENVIRONMENT	24
4.3 DOWNLOADING TENSORFLOW MODEL	29
4.4 CREATE TF RECORDS FOR TRAIN AND TEST	36
CHAPTER 5: TRAIN THE OBJECT DETECTION MODEL	43
5.1 CREATE LABELMAP.PBTEXT FILE	43
5.2 OPEN TENSOR BOARD	43
5.3 EDIT YOUR MODEL CONFIG FILE	44
5.4 TRAIN YOUR MODEL	45
CHAPTER 6: TEST YOUR MODEL	52
6.1 EXPORT THE TRAIN MODEL	56
6.2 RUN THE OBJECT DETECTION MODEL	56
6.3 OUTPUT OF THE MODEL	58
CHAPTER 7: OBJECT DETECTION	61
7.1 SSD (SINGLE SHOT DETECTOR)	62
7.2 FASTER R-CNN	62
7.3 YOLO	65
CONCLUSION	66
REFERENCE	68
GITHUB LINK	68

LIST OF FIGURES:

Figure 1.1 Usage of machine learning in different fields	2
Figure 1.2 Uses of ML	5
Figure 1.3 types of ML	6
Figure 1.4.1 Deep neural network	7
Figure 1.4.2 Deep learning process	10
Figure 1.4.3 Relation between DM, ML, DL	10
Figure 1.4.4 Process in machine learning	11
Figure 2.3.1 Python Download	13
Figure 2.3.2 Python installation	13
Figure 2.3.3 IDLE	14
Figure 2.3.4 After installation	15
Figure 2.3.5 Jupyter notebook	15
Figure 2.8.1.1 class defining	19
Figure 2.8.1.2 Example of class	20
Figure 4.1.1 images of my data set containing aeroplanes	23
Figure 4.1.2 annotations for each image	23
Figure 4.2.1 setting the GPU	24
Figure 4.2.2 creating the main directory	25
Figure 4.2.3 creation of the folders for testing and training	25
Figure 4.2.4 checking the folders in data directory	26
Figure 4.2.5 mounting the google drive	26
Figure 4.2.6 unzipping and extracting the Images, airplanes_annotations files	27
Figure 4.2.7 moving the files	28
Figure 4.2.8 splitting the data	28
Figure 4.2.9 checking the train_labels and test_labels files	28
Figure 4.3.1 tensor flow version 1.X	29
Figure 4.3.2 importing the libraries	29
Figure 4.3.3 installing the required packages	30

Figure 4.3.4 model folder in colab	30
Figure 4.3.5 downloading tf_slim	34
Figure 4.4.1 train data frame	35
Figure 4.4.2 test data frame	37
Figure 4.4.3 shows no rows present with that row no's	37
Figure 4.4.4 converting to .csv file	39
Figure 4.4.5 checking the .csv file for train	39
Figure 4.4.6 checking the .csv file for test	40
Figure 4.4.7 generation of tf records	41
Figure 4.4.8 box plot on an image	42
Figure 5.1.1 creation of labelmap.pbtxt file	43
Figure 5.1.2 display of the file	44
Figure 5.2.1 downloading the ngrok	44
Figure 5.2.2 link to tensor board	45
Figure 5.3.1 store the checkpoint under training folder	46
Figure 5.3.2 download base model	47
Figure 5.3.3 checking the pretrained model	47
Figure 5.3.4 config_Base	47
Figure 5.4.1 training the model	52
Figure 5.4.2 output of the training	53
Figure 5.4.3 DetectionBoxes_Precision_mAP (large)	53
Figure 5.4.4 DetectionBoxes_Precision_mAP (medium)	53
Figure 5.4.5 DetectionBoxes_Precision_mAP@.50IOU	53
Figure 5.4.6 global_norm_gradient_norm	53
Figure 5.4.7 global_step_sec	54
Figure 5.4.8 learning_rate	54
Figure 5.4.9 loss	54
Figure 5.4.10 loss_1	54
Figure 5.4.11 Loss_classification_loss	54
Figure 5.4.12 Loss_total_loss	54

Figure 5.4.13 Loss_HardExampleMiner_NumNegatives	54
Figure 5.4.14 TargetAssignment_Loss_TargetAssignment_Avg	54
Figure 5.4.15 tensor board image 1	55
Figure 5.4.16 tensor board image 2	55
Figure 5.4.17 tensor board image 3	55
Figure 6.1.1 exporting the trained model	56
Figure 6.1.2 checking finetuned model	57
Figure 6.1.3 downloading the three files	58
Figure 6.2.1 loading the images in the numpy array	58
Figure 6.2.2 to upload the images.	58
Figure 6.2.3 importing the libraries	59
Figure 6.2.4 detection of the graph and categorical index	59
Figure 6.2.5 code to detect the object	60
Figure 6.3.1 image 1	61
Figure 6.3.2 image 2	61
Figure 7.1.1 SSD	63
Figure 7.1.2 default boundary boxes	63
Figure 7.1.3 loss function	64
Figure 7.2.1 Faster R-CNN(1)	65
Figure Faster RCNN (2)	66
Figure 7.3.1 YOLO(1)	67
Figure 7.3.2 YOLO (2)	67

CHAPTER 1

MACHINE LEARNING

1 INTRODUCTION:

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress. Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on-the-job improvement of existing machine designs. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down. Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign. New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

1.1 Importance of Machine Learning:

Machine learning is a branch of artificial intelligence that aims at enabling machines to perform their jobs skillfully by using intelligent software. The statistical learning methods constitute the backbone of intelligent software that is used to develop machine intelligence. Because machine learning algorithms require data to learn, the discipline must have connection with the discipline of database. Similarly, there are familiar terms such as Knowledge Discovery from Data (KDD), data mining, and pattern recognition. One wonders how to view the big picture in which such connection is illustrated.

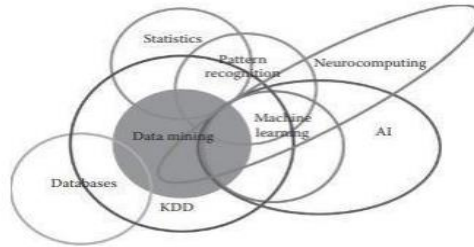


Figure 1.1 usage of Machine learning in different fields

There are some tasks that humans perform effortlessly or with some efforts, but we are unable to explain how we perform them. For example, we can recognize the speech of our friends without much difficulty. If we are asked how we recognize the voices, the answer is very difficult for us to explain. Because of the lack of understanding of such phenomenon (speech recognition in this case), we cannot craft algorithms for such scenarios. Machine learning algorithms are helpful in bridging this gap of understanding .

The idea is very simple. We are not targeting to understand the underlying processes that help us learn. We write computer programs that will make machines learn and enable them to perform tasks, such as prediction. The goal of learning is to construct a model that takes the input and produces the desired result. Sometimes, we can understand the model, whereas, at other times, it can also be like a black box for us, the working of which cannot be intuitively explained. The model can be considered as an approximation of the process we want machines to mimic. In such a situation, it is possible that we obtain errors for some input, but most of the time, the model provides correct answers. Hence, another measure of performance (besides performance of metrics of speed and memory usage) of a machine learning algorithm will be the accuracy of results.

1.2 Uses of Machine Learning:

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don't even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very similar to human brain). Herein, we share few examples of machine learning that we use everyday and perhaps have no idea that they are driven by ML. These are some the uses and applications of ML.

i . Virtual Personal Assistants:

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask "What is my schedule for today?", "What are the flights from Germany to London", or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like "Set an alarm for 6 AM next morning", "Remind me to visit Visa Office day after tomorrow".

Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example:

- Smart Speakers: Amazon Echo and Google Home
- Smartphones: Samsung Bixby on Samsung S8
- Mobile Apps: Google Allo

ii. Predictions while Commuting:

Traffic Predictions: We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there are less number of cars that are equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks: When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer

is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in an interview Movie Recommender System ~ 12 ~ that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

iii. Social Media Services:

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

- **People You May Know:** Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.

- **Face Recognition:** You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.

- **Similar Pins:** Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

iv. Search Engine Result Refining:

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

v. Product Recommendations:

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow matches with your taste. On the basis of your behaviour with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

vi. Online Fraud Detection:

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: Paypal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.



Figure 1.2 Uses of Machine learning

1.3 Types of Machine Learning:

There are 3 types of Machine learning which are widely used in today's world these are:

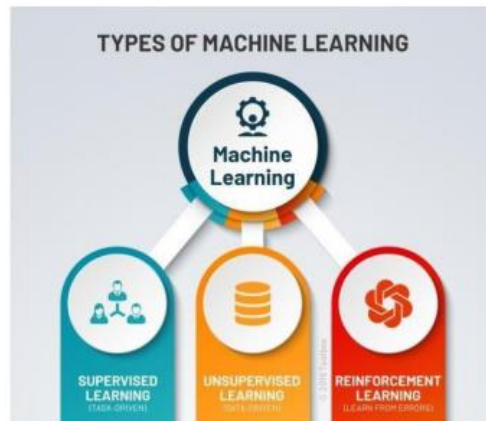


Figure 1.3 types of ML

1.3.1 Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labelled data. Even though the data needs to be labelled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances. In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labelled parameters required for the problem. The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

1.3.2 Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabelled data. This means that human labour is not required to make the dataset machine readable, allowing much larger datasets to be worked on by the program. In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings. The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a

defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

1.3.3 Reinforcement Learning:

It directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’. Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not. In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result. In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

DEEP LEARNING

1.4 Deep Learning and It's Importance:

Deep learning algorithms run data through several “layers” of neural network algorithms, each of which passes a simplified representation of the data to the next layer. Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning model is highly similar to a human brain with large number of neurons and nodes like neurons in human brain thus resulting in artificial neural network. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then

train them which becomes a very tedious job for ML scientist but in neural networks we don't have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition , when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it's correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that its not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were impossible, a decades ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it, be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendations, Mail Services, Self-Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning. Furthermore, Deep learning can most profoundly be considered as future of Artificial Intelligence due to constant rapid increase in amount of data as well as the gradual development in hardware field as well, resulting in better computational power.

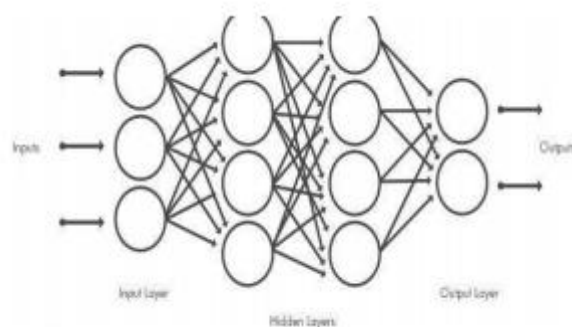


Figure 1.4.1 Deep Neural network

1.5 Uses of Deep Learning:

i. Translations:

Although automatic machine translation isn't new, deep learning is helping enhance automatic translation of text by using stacked networks of neural networks and allowing translations from images.

ii. Adding color to black-and-white images and videos:

It is used to be a very time-consuming process where humans had to add color to black-and-white images and videos by hand can now be automatically done with deep-learning models.

iii. Language recognition:

Deep learning machines are beginning to differentiate dialects of a language. A machine decides that someone is speaking English and then engages an AI that is learning to tell the differences between dialects. Once the dialect is determined, another AI will step in that specializes in that particular dialect. All of this happens without involvement from a human.

iv. Autonomous vehicles:

There's not just one AI model at work as an autonomous vehicle drives down the street. Some deep-learning models specialize in streets signs while others are trained to recognize pedestrians. As a car navigates down the road, it can be informed by up to millions of individual AI models that allow the car to act.

V. Computer vision:

Deep learning has delivered super-human accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.

Vi. Text generation:

The machines learn the punctuation, grammar and style of a piece of text and can use the model it developed to automatically create entirely new text with the proper spelling, grammar and style of the example text. Everything from Shakespeare to Wikipedia entries have been created.

vii. Deep-learning robots:

Deep-learning applications for robots are plentiful and powerful from an impressive deep-learning system that can teach a robot just by observing the actions of a human completing a task to a housekeeping robot that's provided with input from several other AIs in order to take action. Just like how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.

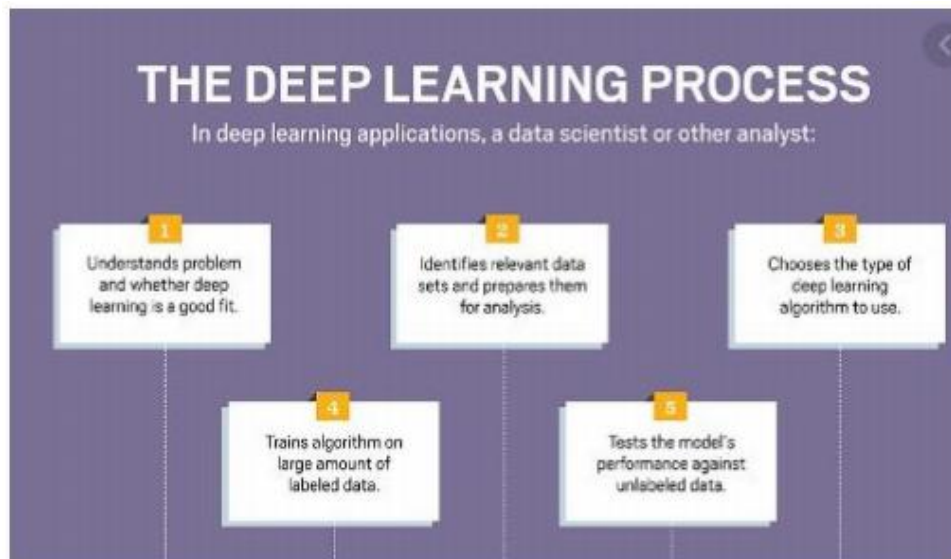


Figure 1.4.2 The deep learning process

1.6 Relation between Data Mining, Machine Learning and Deep Learning:

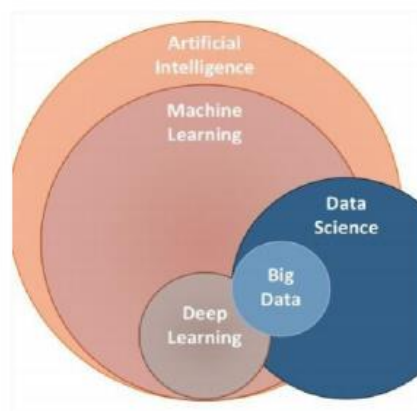


Figure 1.4.3 Relation between DM, ML, DL

The deep learning, data mining and machine learning share a foundation in data science, and there certainly is overlap between the two. Data mining can use machine learning

algorithms Movie Recommender System ~ 20 ~ to improve the accuracy and depth of analysis, and vice-versa; machine learning can use mined data as its foundation, refining the dataset to achieve better results.

You could also argue that data mining and machine learning are similar in that they both seek to address the question of how we can learn from data. However, the way in which they achieve this end, and their applications, form the basis of some significant differences.

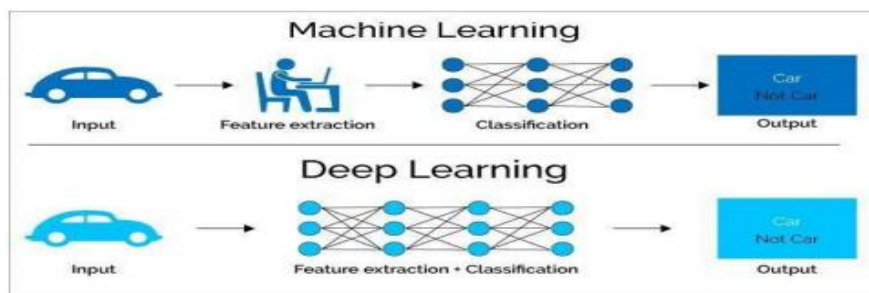


Figure 1.4.4 process in machine learning and deep learning

Machine Learning comprises of the ability of the machine to learn from trained data set and predict the outcome automatically. It is a subset of artificial intelligence.

Deep Learning is a subset of machine learning. It works in the same way on the machine just like how the human brain processes information. Like a brain can identify the patterns by comparing it with previously memorized patterns, deep learning also uses this concept.

Deep learning can automatically find out the attributes from raw data while machine learning selects these features manually which further needs processing. It also employs artificial neural networks with many hidden layers, big data, and high computer resources.

Data Mining is a process of discovering hidden patterns and rules from the existing data. It uses relatively simple rules such as association, correlation rules for the decision making process, etc. Deep Learning is used for complex problem processing such as voice recognition etc. It uses Artificial Neural Networks with many hidden layers for processing. At times data mining also uses deep learning algorithms for processing the data.

CHAPTER 2

PYTHON

2.1 Introduction:

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

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

Python is an interpreted language , object oriented, high level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for Use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse .The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

2.2 Setup of Python:

- Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org/>

2.3 Installation(using python IDLE):

- To start, go to python.org/downloads and then click on the button to download the latest version of Python.
- We can download python IDLE in windows, mac and linux operating systems also.



Figure 2.3.1 Python download

- Run the .exe file that you just downloaded and start the installation of Python by clicking on Install Now
- We can give environmental variable i.e path after completion of downloading



Figure 2.3.2 python installation

- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



Figure 2.3.3 IDLE

2.4 Python Installation using Anaconda:

- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
 - Conda is a package manager quickly installs and manages packages. Anaconda for Windows installation:
- i. Go to the following link: [Anaconda.com/downloads](https://anaconda.com/downloads)



- ii. Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- iii. Select path(i.e. add anaconda to path & register anaconda as default python 3.4)
- iv. Click finish v. Open jupyter notebook.



Figure 2.3.4 After installation



Figure 2.3.5 jupyter notebook

2.5 Features:

- i. **Readable:** Python is a very readable language.
- ii. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn
- iii. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.
- iv. **Open Source:** Python is a open source programming language.
- v. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

- vi. You do not have to bother clearing the memory. **Free:** Python is free to download and use. This means you can download it for free and use it in your application. Python is an example of a FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.
- vii. **Supports exception handling:** If you are new, you may wonder what is an exception? An exception is an event that can occur during program execution and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.
- viii. **Advanced features:** Supports generators and list comprehensions. We will cover these features later.
- ix. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically

2.6 Variable Types:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python has five standard data types –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

2.6.1 Python Numbers:

Number data types store numeric values. They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

Python supports four different numerical types –

- Int (signed integers) – They are often called just integers or ints, are positive or negative whole numbers with no decimal point.

- long (long integers) – Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.

- float (floating point real values) – Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 ($2.5e2 = 2.5 \times 10^2 = 250$).

2.6.2 Python Strings:

In Python, Strings can be created by simply enclosing characters in quotes. Python does not support character types. These are treated as length-one strings, and are also considered as substrings. Substrings are immutable and can't be changed once created. Strings are the ordered blocks of text that are enclosed in single or double quotations. Thus, whatever is written in quotes, is considered as string. Though it can be written in single or double quotations, double quotation marks allow the user to extend strings over multiple lines without backslashes, which is usually the signal of continuation of an expression, e.g., 'abc', "ABC".

2.6.3 Python lists:

- List is a collection data type in python. It is ordered and allows duplicate entries as well. Lists in python need not be homogeneous, which means it can contain different data types like integers, strings and other collection data types. It is mutable in nature and allows indexing to access the members in a list.

- To declare a list, we use the square brackets.

- List is like any other array that we declare in other programming languages. Lists in python are often used to implement stacks and queues. The lists are mutable in nature. Therefore, the values can be changed even after a list is declared.

2.6.4 python tuples:

A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also.

2.6.5 Python Dictionary:

It is a collection data type just like a list or a set, but there are certain features that make python dictionary unique. A dictionary in python is not ordered and is changeable as well. We can make changes in a dictionary unlike sets or strings which are immutable in nature. Dictionary contains key-value pairs like a map that we have in other programming languages. A dictionary has indexes. Since the value of the keys we declare in a dictionary are always unique, we can use them as indexes to access the elements in a dictionary.

2.7 Functions:

2.7.1 Defining a Function:

- Function blocks begin with the keyword `def` followed by the function name and parentheses `()`.
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or docstring.
- The code block within every function starts with a colon `(:)` and is indented.
- The statement `return [expression]` exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as `return None`.

2.7.2 Calling a Function:

- Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.
- Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.8 OOPs Concepts:

2.8.1 Class:

- Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stresses on objects..

- An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.
- We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.
- As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called instantiation.
- Like function definitions begin with the def keyword in Python, class definitions begin with a class keyword.
- The first string inside the class is called docstring and has a brief description about the class.

```
class MyNewClass:  
    '''This is a docstring. I have created a new class'''  
    pass
```

Figure 2.8.1.1 Class defining

- As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

```
class Person:
    "This is a person class"
    age = 10

    def greet(self):
        print('Hello')

# Output: 10
print(Person.age)

# Output: <function Person.greet>
print(Person.greet)

# Output: 'This is my second class'
print(Person.__doc__)
```

Figure 2.8.1.2 Example of class

CHAPTER 3

OBJECT DETECTION

(CASE STUDY)

3.1 Problem statement:

To detect aeroplane as an object in an image.

3.2 Data set:

- Collect the images from the chrome or from the other web sites . For each image there must be annotations present for objects to be detected.
- There are separate annotation files for each image. The Annotations are the csv files.
- There are 2 csv files used for training the object detection model i.e. train_labels.csv and test_labels.csv.
- The data set of the train_labels and test_labels contains the following parameters:
- **Filename:** It is the column for the image filename with respective path.
- **Points of the Single Class Object-**The points for the single class object are given as **Xmin, Ymin, Xmax and Ymax**. These points represent the location of the object in the image.
- **Class-**It is the column for telling the object class present in the image (particular object image).

3.3 OBJECTIVE OF CASE STUDY:

To detect the objects on the images based on the trained image data set using the Tensor flow and Deep Learning (convolutional Neural Network). This model will help the airplane management system .

CHAPTER 4

MODEL BUILDING

4. PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

4.1 Getting the data set :

Get the data set from the chrome or the other web sites . (If you want to check with my data set then get it from my “git hub” link that is provided at the last paste documentation).

Upload the data set from your local system to your google drive as a zip file format.

If the files are in the zip format that would help us to easily extract those files and use them. Otherwise we have to always upload those data set every time when we want to run , it becomes more complicated .

My data set has the images folder and annotations folder.

My data set images need not be labelled separately because ,I had its annotations with my images its itself.

The below one shows my data set:

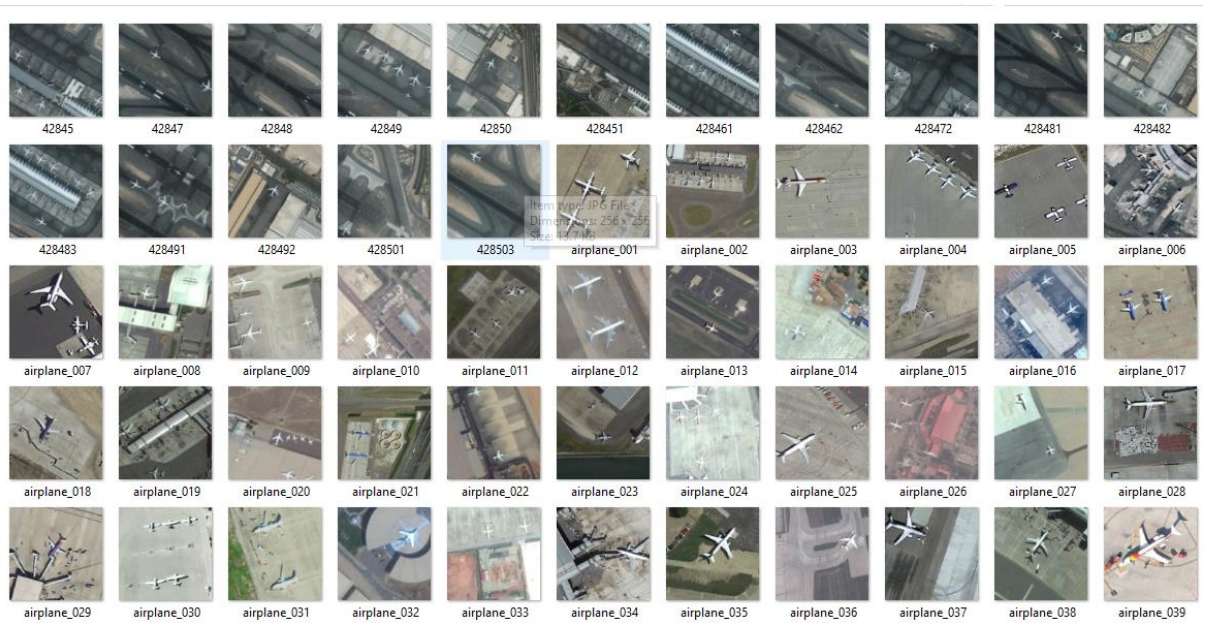


Figure 4.1.1 images of my data set containing aeroplanes

Name	Date modified	Type	Size
42845	19-07-2020 22:01	Microsoft Excel C...	1 KB
42847	19-07-2020 22:02	Microsoft Excel C...	1 KB
42848	19-07-2020 22:07	Microsoft Excel C...	1 KB
42849	19-07-2020 22:08	Microsoft Excel C...	1 KB
42850	19-07-2020 22:08	Microsoft Excel C...	1 KB
428451	19-07-2020 22:08	Microsoft Excel C...	1 KB
428461	19-07-2020 22:09	Microsoft Excel C...	1 KB
428462	19-07-2020 22:09	Microsoft Excel C...	1 KB
428472	19-07-2020 22:10	Microsoft Excel C...	1 KB
428481	19-07-2020 22:10	Microsoft Excel C...	1 KB
428482	19-07-2020 22:10	Microsoft Excel C...	1 KB
428483	19-07-2020 22:10	Microsoft Excel C...	1 KB
428491	19-07-2020 22:11	Microsoft Excel C...	1 KB
428492	19-07-2020 22:11	Microsoft Excel C...	1 KB
428501	19-07-2020 22:11	Microsoft Excel C...	1 KB
428503	19-07-2020 22:12	Microsoft Excel C...	1 KB
airplane_001	19-07-2020 22:13	Microsoft Excel C...	1 KB
airplane_002	19-07-2020 22:13	Microsoft Excel C...	1 KB
airplane_003	19-07-2020 22:13	Microsoft Excel C...	1 KB
airplane_004	19-07-2020 22:13	Microsoft Excel C...	1 KB
airplane_005	19-07-2020 22:14	Microsoft Excel C...	1 KB
airplane_006	19-07-2020 22:14	Microsoft Excel C...	1 KB
airplane_007	19-07-2020 22:15	Microsoft Excel C...	1 KB
airplane_008	19-07-2020 22:15	Microsoft Excel C...	1 KB
airplane_009	19-07-2020 22:15	Microsoft Excel C...	1 KB
airplane_010	19-07-2020 22:15	Microsoft Excel C...	1 KB
airplane_011	19-07-2020 22:15	Microsoft Excel C...	1 KB
airplane_012	19-07-2020 22:16	Microsoft Excel C...	1 KB

Figure 4.1.2 annotations for each image

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	159 159 187 202																				
2	193 169 225 212																				
3	230 181 264 227																				
4	308 211 339 254																				
5	379 238 410 284																				
6	416 250 447 294																				
7	453 263 483 309																				
8	489 279 518 327																				
9	524 289 558 343																				
10	693 361 722 400																				

The above csv file contains the plane12 image annotations in the csv format.

It contain the xmin , ymin , xmax , ymax values in the csv file.

4.2 Setting the environment :

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

- **(Highly Recommended) Mount Google Drive to the Colab notebook:**
- When training starts, checkpoints, logs and many other important files will be created. When the kernel disconnects, these files, along with everything else, will be deleted if they don't get saved on your Google Drive or somewhere else.
- The kernel disconnects shortly after your computer sleeps or after using the Colab **GPU** for 12 hours. Training will need to be restarted from zero if the trained model did not get saved.

From the google drive , we have to access the google colaboratory.

Now we need to set run time to GPU .The GPU compiles the cells more faster than the CPU.

Setting the run type in the google colaboratory.

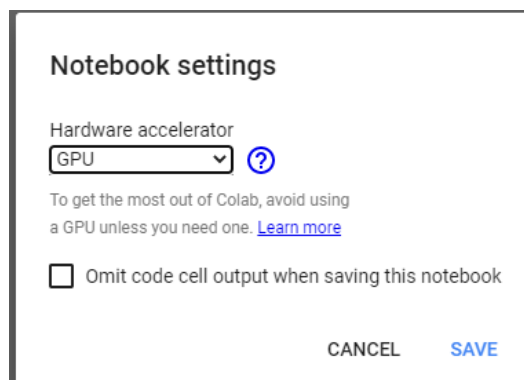
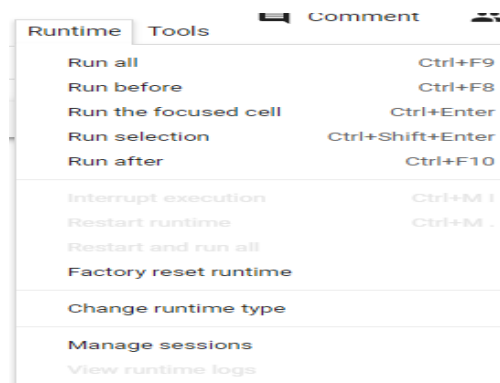


Figure 4.2.1 setting the GPU

Now creating the main directory for the whole project.

```
[ ] #creates a directory for the whole project
!mkdir aeroplane_detection
```

Figure 4.2.2 creating the main directory

We to access this directory from the content , so change the current directory to content/aeroplane_detection.

```
▶ # changing the current directory
%cd /content/aeroplane_detection

📁 /content/aeroplane_detection
```

As of now just created the directory , but now we should store the data set in the colab .

To upload those data set in our main directory we need create the particular folders.

Creation of data directory under the main directory and creating the three folders under it as images , train_labels , test_labels.

```
[ ] # creating a directory to store the training and testing data
!mkdir data

# folders for the training and testing data.
!mkdir data/images data/train_labels data/test_labels
```

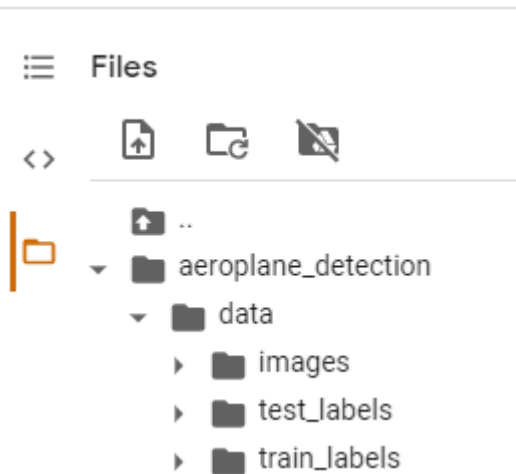
Figure 4.2.3 creation of the folders for testing and training

```
▶ # importing the os to check the files in the directory
import os
os.listdir('data')

📁 ['images', 'test_labels', 'train_labels']
```

Figure 4.2.4 checking the folders in data directory

- The below figure shows the how they are created.



- As the zip files are in the google drive , we need to mount the google drive to access those files and to extract them.
- Colab will ask the authorization for access your files before mounting. Once the Authorization is confirmed then Google Drive is Mounted.

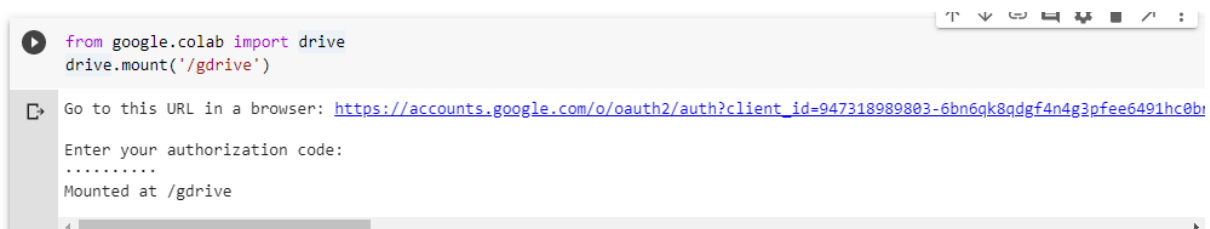
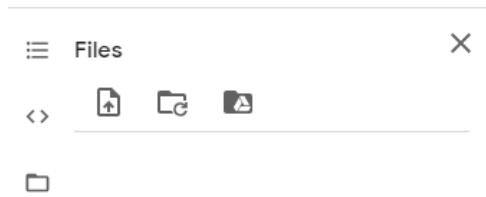
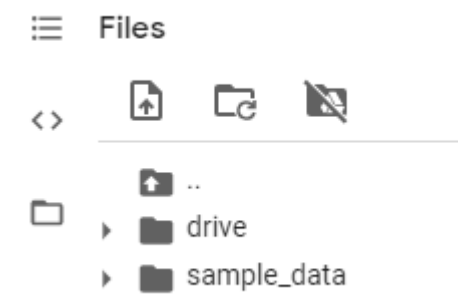


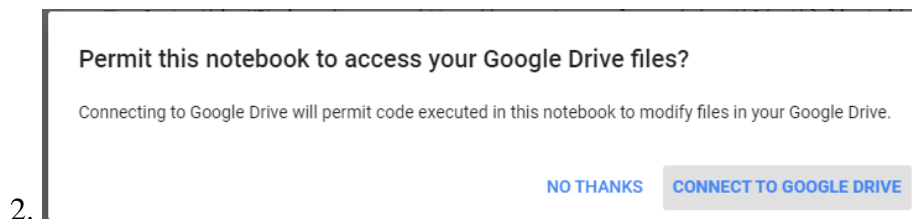
Figure 4.2.5 mounting the google drive



1.



3.



2.

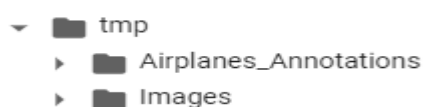
Now we should extract the images, annotations files by unzipping them ,through below code.

```
[ ] # extracting the the images zip files from the drive and storing it in the tmp folder
import os
import zipfile
local_zip = "/content/drive/My Drive/Images.zip" ## which zip file you want to extarct
zip_ref = zipfile.ZipFile(local_zip,'r')
zip_ref.extractall('/tmp') ## In which location you want to store extracted files
zip_ref.close()
```

```
▶ # extracting the the Airplanes_Annotations zip files from the drive and storing it in the tmp folder
import os
import zipfile
local_zip = "/content/drive/My Drive/Airplane_Annotation.zip" ## which zip file you want to extarct
zip_ref = zipfile.ZipFile(local_zip,'r')
zip_ref.extractall('/tmp') ## In which location you want to store extracted files
zip_ref.close()
```

Figure 4.2.6 unzipping and extracting the Images, airplanes_annotations files

- The above extracted files are store in the /tmp folder



```

▶ # checking the images folder whether present or not
os.listdir("/tmp/Images")

✎ 'airplane_608.jpg',
  'airplane_540.jpg',
  'airplane_017.jpg',
  'airplane_430.jpg',
  '42849.jpg',
  'airplane_516.jpg',
  'airplane_446.jpg',
  'airplane_664.jpg',
  'airplane_154.jpg',
  'airplane_273.jpg',
  'airplane_401.jpg'

▶ # checking the airplanes_annotations file present or not
os.listdir("/tmp/Airplanes_Annotations")

✎ 'airplane_101.csv',
  'airplane_630.csv',
  'airplane_662.csv',
  'airplane_118.csv',
  'airplane_095.csv',
  'airplane_124.csv',
  'airplane_017.csv',
  'airplane_323.csv',
  'airplane_416.csv',
  'airplane_121.csv',

```

- We need to move the files from /tmp to data directory , all Images to images folders and all the aeroplane_annotations to train_labels.
- Before moving we need check the present directory.

```

[ ] !pwd

✎ /content/aeroplane_detection

```

- Now we can move the files to the locations mentioned above.

```

[ ] # moving the img files from tmp and to the data/images
    !mv /tmp/Images/* data/images

▶ # moving the annotaion from tmp and to data/train labels
    !mv /tmp/Airplanes_Annotations/* data/train_labels

```

Figure 4.2.7 moving the files

- We should split the data into train and test for the further process to be done.
- We should split the data as train 80% and test 20%.
- Below code would specify splitting the data .

```

▶ # lists the files inside 'annotations' in a random order (not really random, by their hash value instead)
  # Moves the first 200 labels to the testing dir: `test_labels`
  !ls data/train_labels/* | sort -R | head -200 | xargs -I{} mv {} data/test_labels

```

Figure 4.2.8 splitting the data

- The data set contain total 729 Images and 729 Aeroplane_Annotations.
- So we split the data for the train as 529 files and test as 200 files.

```
[22] # to divide into train labels into 529
      !ls data/train_labels/ | wc -l

529

# to divide into the test labels
!ls data/test_labels/ | wc -l

200
```

Figure 4.2.9 checking the train_labels and test_labels files

4.3 DOWNLOAD TENSORFLOW MODELS:

- We need to select the tensor flow version before starting the download of the model .

```
[ ] %tensorflow_version 1.x

TensorFlow 1.x selected.
```

Figure 4.3.1 tensor flow version 1.X

```
#we need tensorflow v 1.15.0, object detection API is removed from tf v 2.0+
#!pip install tensorflow==1.15.0

print(tf.__version__)

1.15.2
```

- The code checking whether the tensor flow 1.x version or not. For this model we need the 1.x
- we need to import the packages that are required for the model building.

```

▶ from __future__ import division, print_function, absolute_import

import pandas as pd
import numpy as np
import csv
import re
import cv2
import os
import glob
import xml.etree.ElementTree as ET

import io
import tensorflow.compat.v1 as tf

from PIL import Image
from collections import namedtuple, OrderedDict

import shutil
import urllib.request
import tarfile

from google.colab import files

```

Figure 4.3.2 importing the libraries

- We need to install the required packages

```

[4] !apt-get install -qq protobuf-compiler python-pil python-lxml python-tk

!pip install -qq Cython contextlib2 pillow lxml matplotlib

!pip install -qq pycocotools

```

Figure 4.3.3 installing the required packages

```

Selecting previously unselected package python-bs4.
(Reading database ... 144487 files and directories currently installed.)
Preparing to unpack .../0-python-bs4_4.6.0-1_all.deb ...
Unpacking python-bs4 (4.6.0-1) ...
Selecting previously unselected package python-pkg-resources.
Preparing to unpack .../1-python-pkg-resources_39.0.1-2_all.deb ...
Unpacking python-pkg-resources (39.0.1-2) ...
Selecting previously unselected package python-chardet.
Preparing to unpack .../2-python-chardet_3.0.4-1_all.deb ...
Unpacking python-chardet (3.0.4-1) ...
Selecting previously unselected package python-six.
Preparing to unpack .../3-python-six_1.11.0-2_all.deb ...
Unpacking python-six (1.11.0-2) ...
Selecting previously unselected package python-webencodings.
Preparing to unpack .../4-python-webencodings_0.5.2-1_all.deb ...
Unpacking python-webencodings (0.5.2) ...
Selecting previously unselected package python-html5lib.
Preparing to unpack .../5-python-html5lib_0.99999999-1_all.deb ...
Unpacking python-html5lib (0.99999999-1) ...
Selecting previously unselected package python-lxml:amd64.
Preparing to unpack .../6-python-lxml_4.2.1-1ubuntu0.1_amd64.deb ...
Unpacking python-lxml:amd64 (4.2.1-1ubuntu0.1) ...
Selecting previously unselected package python-olefile.
Preparing to unpack .../7-python-olefile_0.45.1-1_all.deb ...
Unpacking python-olefile (0.45.1-1) ...
Selecting previously unselected package python-pil:amd64.
Preparing to unpack .../8-python-pil_5.1.0-1ubuntu0.3_amd64.deb ...
Unpacking python-pil:amd64 (5.1.0-1ubuntu0.3) ...
Setting up python-pkg-resources (39.0.1-2) ...

```

- Description of packages we have imported.

NUMPY:

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in Numpy is called `ndata`, it provides a lot of supporting functions that make working with `ndarray` very easy.

PANDAS:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

CSV:

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. The `csv` module implements classes to read and write tabular data in **CSV** format.

RE:

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.

CV2:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. `cv2.imread()` method loads an image from the specified file.

SEABORN:

Seaborn is a Python data visualization library based on `matplotlib`. It provides a high-level interface for drawing attractive and informative statistical graphics.

MATPLOTLIB:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. Matplotlib along with Numpy can be considered as the open source equivalent of MATLAB.

OS:

OS module provides allows you to interface with underlying operating systems that python is running on – be that Windows, Linux, or Mac.

GLOB:

In Python, the glob module is used to retrieve files/pathnames matching a specified pattern. The pattern rules of glob follow standard Unix path expansion rules.

IO:

The `io` module provides the Python interfaces to stream handling.

COLLECTIONS:

Collections in Python are containers that are used to store collections of data, for example, list, dict, set, tuple.

SHUTIL:

Python `shutil`. Python `shutil` module enables us to operate with file objects easily and without diving into file objects a lot. It takes care of low-level semantics like creating file objects, closing the files once they are copied and allows us to focus on the business logic of our program.

TARFILE:

The `tarfile` module makes it possible to read and write tar archives, including those using gzip or bz2 compression. Use the `zipfile` module to read or write .zip files, or the higher-level functions in `shutil`. `shutil` reads and writes gzip and bz2 compressed archives if the respective modules are available.

- Now we need to select which model to download for the detection of an object.
- The model used for this project is `ssd_mobilenet_v2_coco`.

- I am choosing a model that has a high inference speed (ms) with relatively high MAP on COCO.

```
# Some models to train on
MODELS_CONFIG = {
    'ssd_mobilenet_v2': {
        'model_name': 'ssd_mobilenet_v2_coco_2018_03_29', # Layers input and output shape how many neurons in each layer(Model building)
        'pipeline_file': 'ssd_mobilenet_v2_coco.config', # While compiling and fitting the model some parameters should be tuned (Defining the arguments values)
    },
    'faster_rcnn_inception_v2': {
        'model_name': 'faster_rcnn_inception_v2_coco_2018_01_28',
        'pipeline_file': 'faster_rcnn_inception_v2_pets.config',
    },
    'rfcn_resnet101': {
        'model_name': 'rfcn_resnet101_coco_2018_01_28',
        'pipeline_file': 'rfcn_resnet101_pets.config',
    }
}

# Select a model in `MODELS_CONFIG`.
# I chose ssd_mobilenet_v2 for this project, you could choose any
selected_model = 'ssd_mobilenet_v2'
```

- From the following link download the models folder of TensorFlow.

<https://github.com/tensorflow/models>


Extract the zip folder and place it into your main directory.

- Models Folder consist of the following files and folders:

TF Object Detection Team Merge pull request #9021 from kmindspark:context_fix2 3b384f5 4 hours ago 5,189 commits		
.github	Update README_TEMPLATE.md	last month
community	Add Intel's object detection models to community README (#8867)	17 days ago
official	PY3 migration	16 hours ago
orbit	Internal change	yesterday
research	Merge pull request #9021 from kmindspark:context_fix2	4 hours ago
.gitignore	Update .gitignore	3 months ago
AUTHORS	Spatial Transformer model	4 years ago
CODEOWNERS	Update CODEOWNERS	3 months ago
CONTRIBUTING.md	Add new templates and update README files (#8467)	3 months ago
ISSUES.md	Update ISSUES.md	3 months ago
LICENSE	Update LICENSE	5 years ago
README.md	Update README.md	22 days ago

Check the present working directory to download the tensor flow model, make sure it is `aeroplane_detection` directory .

```
# Downloads Tenorflow
%cd /content/aeroplane_detection/
!git clone --q https://github.com/tensorflow/models.git
```

 /content/aeroplane_detection

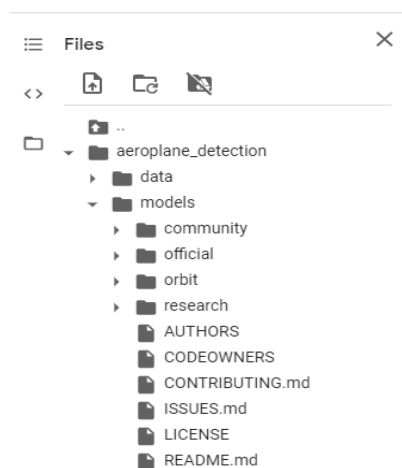


Figure 4.3.4 model folder in colab

Now we have move the official to under the research folder, for this we use the “move” command.

```
!mv models/official models/research/official
```

We have to set the environment for the python path for environment variables .

Store the environment variable under the research folder, so change the directory to research.

```
%cd /content/aeroplane_detection/models/research
#compiling the proto buffers (not important to understand for this project but you can learn more
about them here: https://developers.google.com/protocol-buffers/)
!protoc object_detection/protos/*.proto --python_out=.

# exports the PYTHONPATH environment variable with the reasearch and slim folders' paths
os.environ['PYTHONPATH'] += '/content/aeroplane_detection/models/research:/content/aeropl
ane_detection/models/research/slim/'
```

```
/content/aeroplane_detection/models/research
```

We need check the whether all we need are installed for training.

- Install the tf.slim package to run the model_builder_test.py code that is present in the directory.
- Run the model_builder_test.py program.

```
# testing the model builder
!pip install tf_slim

!python3 object_detection/builders/model_builder_test.py
```

```
Collecting tf_slim
  Downloading https://files.pythonhosted.org/packages/02/97/b0f4a64df018ca018cc035d44f2ef08f91e2e8aa67271f6f19633a015ff7/tf_slim-1.1.0-py2.py3-none-any.whl (352k)
    358kB 6.7MB/s
Requirement already satisfied: absl-py>=0.2.2 in /usr/local/lib/python3.6/dist-packages (from tf_slim) (0.9.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from absl-py>=0.2.2->tf_slim) (1.15.0)
Installing collected packages: tf-slim
Successfully installed tf-slim-1.1.0
```

Figure 4.3.5 downloading tf_slim

Tf_slim:

TF-Slim is a library that makes defining, training and evaluating neural networks simple.

Models can be succinctly *defined* using *TF-Slim* by combining its variables, layers and

scopes.

4.4 CREATE TF RECORDS FOR TRAIN AND TEST DATA:

Now we have to generate the two tf records for the train and the test data.

But these tf records are generated when we have .csv files in the data folder .

So we have to get .csv files from the train_labels and test_labels.

Here we are going convert my annotations from both test_labels and train_labels to single data frame for test and train.

So here we have to take the list to store all the csv and then converting it into data frame

Now change directory to train labels to access the files.

```
[25] !pwd
```

```
↳ /content/aeroplane_detection/data
```

```
[26] cd train_labels
```

```
↳ /content/aeroplane_detection/data/train_labels
```

```
▶ train=[]  
import os  
for f in os.listdir():  
    file1=open(f,'r')  
    for line in file1.readlines():  
        line=line.strip('\n')  
        l=[str(f.split('.')[0])+".jpg"]  
        l=l+line.split(' ')  
        l.append('aeroplane')  
        train.append(l)  
  
print(len(train))
```

```
↳ 2157
```



```
[90] train.head()
```

	filename	xmin	ymin	xmax	ymax	class
0	airplane_336.jpg	80	72	184	171	aeroplane
1	42847.jpg	84	71	116	101	aeroplane
2	42847.jpg	225	154	255	190	aeroplane
3	airplane_128.jpg	29	25	98	77	aeroplane
4	airplane_128.jpg	69	84	145	145	aeroplane

Figure 4.4.1 train data frame

Now change the directory to test_labels.

```
[28] cd -
```

```
/content/aeroplane_detection/data
```

```
[29] cd test_labels
```

```
/content/aeroplane_detection/data/test_labels
```

```
[30] !pwd
```

```
/content/aeroplane_detection/data/test_labels
```

```
test=[]
import os
for f in os.listdir():
    file1=open(f,'r')
    for line in file1.readlines():
        line=line.strip('\n')
        l=[str(f.split('.')[0])+".jpg"]
        l=l+line.split(' ')
        l.append('aeroplane')
        test.append(l)
print(len(test))
```

```
750
```

```
[91] test.head()
```

	filename	xmin	ymin	xmax	ymax	class
0	airplane_212.jpg	132	140	154	155	aeroplane
1	airplane_212.jpg	156	191	172	209	aeroplane
2	airplane_212.jpg	196	17	242	56	aeroplane
3	airplane_602.jpg	114	90	148	127	aeroplane
4	airplane_602.jpg	137	137	181	177	aeroplane

Figure 4.4.2 test data frame

Now we have to generate the train_labels as one .csv file and other test_labels as one .csv file.

Check whether the data is clean or not , does the data contains any missing values or null values , if present any then check through the following code.

```
1 # checking the missing values for train
2 train.isnull().sum()

filename    0
xmin        0
ymin        0
xmax        1
ymax        1
class       1
dtype: int64
```

```
test.isnull().sum()

filename    0
xmin        0
ymin        0
xmax        2
ymax        2
class       2
dtype: int64
```

In the train data we have null values and test data also we have null values.

To remove those rows we need drop them through “drop” command.

```
1 # missing values in the data frame in train
2 train[train['ymin']=='aeroplane']
```

	filename	xmin	ymin	xmax	ymax	class
1282	airplane_112.jpg	8	aeroplane	None	None	None

```
test[test['ymin']=='aeroplane']
```

	filename	xmin	ymin	xmax	ymax	class
502	airplane_112.jpg	8	aeroplane	None	None	None
608	airplane_287.jpg	2	aeroplane	None	None	None

Dropping the rows

- Dropping the 1282th row in train data frame.
- Dropping the 502th , 608th rows in test data frame.

```

1 # dropping
2 train.drop([1282],axis=0,inplace=True)
3

```

```
[70] test.drop([502,608],axis=0,inplace=True)
```

```

▶ train[train['ymin']=='aeroplane']
┌───┬──────────┬───┬───┬───┬───┬───┐
│   │ filename │ xmin │ ymin │ xmax │ ymax │ class │
├───┴──────────┴───┴───┴───┴───┴───┴───┤
[71] test[test['ymin']=='aeroplane']
┌───┬──────────┬───┬───┬───┬───┬───┐
│   │ filename │ xmin │ ymin │ xmax │ ymax │ class │
├───┴──────────┴───┴───┴───┴───┴───┴───┤

```

Figure 4.4.3 shows no rows present with that row no's

Now we can convert it into.csv file for both test and train.Store it under data folder.

```

[72] train.to_csv('/content/aeroplane_detection/data/train_labels.csv',index=False)

▶ test.to_csv('/content/aeroplane_detection/data/test_labels.csv',index=False)

```

Figure 4.4.4 converting to .csv file

```

▶ train_df=pd.read_csv('/content/aeroplane_detection/data/train_labels.csv')
train_df.head()
┌───┬──────────┬───┬───┬───┬───┬───┐
│   │ filename │ xmin │ ymin │ xmax │ ymax │ class │
├───┴──────────┴───┴───┴───┴───┴───┴───┤
0  airplane_336.jpg    80    72    184    171  aeroplane
1      42847.jpg     84    71    116    101  aeroplane
2      42847.jpg    225   154    255    190  aeroplane
3  airplane_128.jpg    29    25     98     77  aeroplane
4  airplane_128.jpg    69    84    145    145  aeroplane

```

Figure 4.4.5 checking the .csv file for train

```
test_df=pd.read_csv('/content/aeroplane_detection/data/test_labels.csv')
test_df.head()
```

	filename	xmin	ymin	xmax	ymax	class
0	airplane_212.jpg	132	140	154	155	aeroplane
1	airplane_212.jpg	156	191	172	209	aeroplane
2	airplane_212.jpg	196	17	242	56	aeroplane
3	airplane_602.jpg	114	90	148	127	aeroplane
4	airplane_602.jpg	137	137	181	177	aeroplane

Figure 4.4.6 checking the .csv file for test

Now we have the .csv files to generate the tf records.

```
#adjusted from: https://github.com/datitran/raccoon\_dataset

# converts the csv files for training and testing data to two TFRecords files.
# places the output in the same directory as the input

from object_detection.utils import dataset_util
%cd /content/aeroplane_detection/models/

DATA_BASE_PATH = '/content/aeroplane_detection/data/'
image_dir = DATA_BASE_PATH + 'images/'

def class_text_to_int(row_label):
    if row_label == 'aeroplane':
        return 1
    else:
        None

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]
```

```

def create_tf_example(group, path):
    with tf.io.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        width, height = image.size

        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        xmins = []
        xmaxs = []
        ymins = []
        ymaxs = []
        classes_text = []
        classes = []

        for index, row in group.object.iterrows():
            xmins.append(row['xmin'] / width)
            xmaxs.append(row['xmax'] / width)
            ymins.append(row['ymin'] / height)
            ymaxs.append(row['ymax'] / height)
            classes_text.append(row['class'].encode('utf8'))
            classes.append(class_text_to_int(row['class']))

```

```

tf_example = tf.train.Example(features=tf.train.Features(feature={
    'image/height': dataset_util.int64_feature(height),
    'image/width': dataset_util.int64_feature(width),
    'image/filename': dataset_util.bytes_feature(filename),
    'image/source_id': dataset_util.bytes_feature(filename),
    'image/encoded': dataset_util.bytes_feature(encoded_jpg),
    'image/format': dataset_util.bytes_feature(image_format),
    'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
    'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
    'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
    'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
    'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
    'image/object/class/label': dataset_util.int64_list_feature(classes),
}))
return tf_example

```

```

for csv in ['train_labels', 'test_labels']:
    writer = tf.io.TFRecordWriter(DATA_BASE_PATH + csv + '.record')
    path = os.path.join(image_dir)
    examples = pd.read_csv(DATA_BASE_PATH + csv + '.csv')
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), DATA_BASE_PATH + csv + '.record')
    print('Successfully created the TFRecords: {}'.format(DATA_BASE_PATH + csv + '.record'))

```

```

➡ /content/aeroplane_detection/models
Successfully created the TFRecords: /content/aeroplane_detection/data/train_labels.record
Successfully created the TFRecords: /content/aeroplane_detection/data/test_labels.record

```

Figure 4.4.7 generation of tf records

Two major changes to be made before generation they are :

1. Give the base bath path as , access from the data folder
2. Changing the row label.

Next step is to check whether the boundary box is fixed on the object or not .

For that we use matplotlib library to plot the boundary box around the object .

Below code specify how to plot.

```
%cd /content/aeroplane_detection/data
from PIL import Image

os.listdir('images')
#image = Image.open("images/4f30dc4b8e2a583d.jpg")
#height,width=image.size
#print(height,width)
import matplotlib.pyplot as plt
import matplotlib.patches as patches
for row in train.values[1:]:
    img = plt.imread("images/"+row[0])
    plt.imshow(img)
    print(row[0],row[2],row[3],row[4])
    plt.plot([float(row[1]),float(row[3]),float(row[3]),float(row[1]),float(row[1]),float(row[1]),float(row[4]),float(row[4]),float(row[2]),float(row[2]),float(row[4])],color='r')
    break
```

We have to mention the xmin,ymin,xmax ,ymin row indices values to plot the box

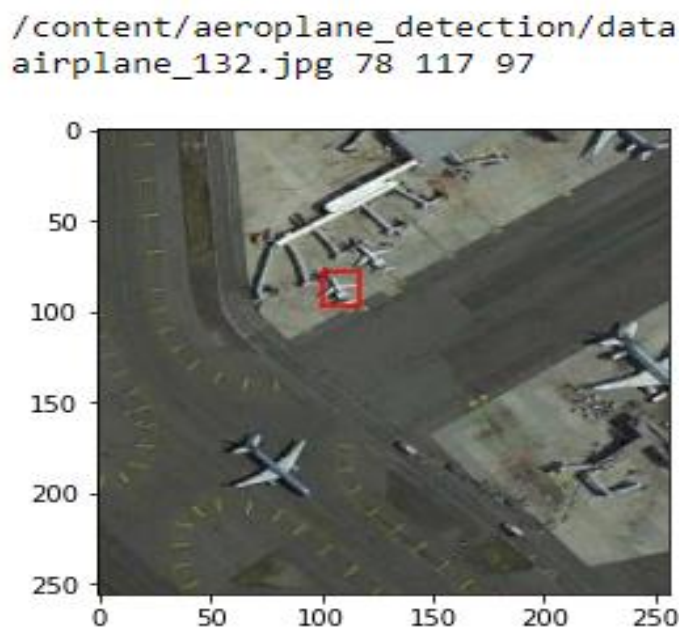


Figure 4.4.8 box plot on an image

CHAPTER 5

TRAIN THE OBJECT DETECTION MODEL

5.1 Creation of labelmap.pbtxt files:

- We need to create the labelmap for the model. This is to be create under the main directory data folder .
- Here we mention the classes as a list .
- Here we have to provide id, class name and the display name .
- Below codes specifies all the things mentioned above.

```
## creation pb txt file and representation of the classess file
%cd /content/aeroplane_detection/data
label_map_path = os.path.join("label_map.pbtxt")
classes=['aeroplane']
pbtxt_content = ""
print(type(classes))
#creates a pbtxt file the has the class names.
for i, class_name in enumerate(classes):
    # display_name is optional.
    pbtxt_content = (
        pbtxt_content
        + "item {{\n    id: {0}\n    name: '{1}'\n    display_name: 'aeroplane'\n }}\n\n".format(i + 1, class_name)
    )
pbtxt_content = pbtxt_content.strip()
with open(label_map_path, "w") as f:
    f.write(pbtxt_content)
```

 /content/aeroplane_detection/data
<class 'list'>

Figure 5.1.1 creation of labelmap.pbtxt file

- We need to check the file whether it's giving the above mention parameters, this can be done through the cat command.

```
[ ] !pwd
[ ] /content/aeroplane_detection/data

#checking the ptxt file
!cat label_map.ptxt

item {
  id: 1
  name: 'aeroplane'
  display_name: 'aeroplane'
}
```

```
[ ] # to check the files under the data folder
!ls -l

total 196
drwxr-xr-x 2 root root 32768 Aug 2 05:26 images
-rw-r--r-- 1 root root 71 Aug 2 05:27 label_map.ptxt
drwxr-xr-x 2 root root 12288 Aug 2 05:27 test_labels
-rw-r--r-- 1 root root 32101 Aug 2 05:27 test_labels.csv
drwxr-xr-x 2 root root 32768 Aug 2 05:27 train_labels
-rw-r--r-- 1 root root 85115 Aug 2 05:27 train_labels.csv
```

Figure 5.1.2 display of the file

5.2 Open tensor board:

- We need generate a tensor board for the visualization of the scalar graphs and the images.
- For this we to unzip the ngrok and download .
- From the we have to download <https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip>

```
#downlaoding ngrok to be able to access tensorboard on google colab
!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
!unzip -o ngrok-stable-linux-amd64.zip

--2020-08-02 06:12:24-- https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
Resolving bin.equinox.io (bin.equinox.io)... 52.20.173.116, 34.227.164.168, 52.22.53.129, ...
Connecting to bin.equinox.io (bin.equinox.io)|52.20.173.116|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13773305 (13M) [application/octet-stream]
Saving to: 'ngrok-stable-linux-amd64.zip'

ngrok-stable-linux- 100%[=====>] 13.13M 13.2MB/s in 1.0s

2020-08-02 06:12:26 (13.2 MB/s) - 'ngrok-stable-linux-amd64.zip' saved [13773305/13773305]

Archive: ngrok-stable-linux-amd64.zip
inflating: ngrok
```

Figure 5.2.1 downloading the ngrok

- We need give the host and port how it should visible in the tensor board.
- Below code it specifies all the parameters


```

▶ #the logs that are created while training
LOG_DIR = model_dir
get_ipython().system_raw(
    'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &'
    .format(LOG_DIR)
)
get_ipython().system_raw('./ngrok http 6006 &')

```

```

[80] #The link to tensorboard.
      #works after the training starts.

      ### note: if you didnt get a link as output, rerun this cell and the one above
      !curl -s http://localhost:4040/api/tunnels | python3 -c \
          "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"

```

🔗 <https://59d2d4c27b8a.ngrok.io>

Figure 5.2.2 link to tensor board

- The link for the tensor board will be accessed when the training of the model is done.
- Once the training of the model completes the we can able open the tensor and download the graphs.
- Tensor Board provides the visualization and tooling needed for machine learning experimentation
- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph (ops and layers)
- Viewing histograms of weights, biases, or other tensors as they change over time
- Displaying images, text, and audio data

5.3 EDIT YOUR MODEL CONFIG FILE:

1. We need to store the check points that are generated while training.

```
[81] !pwd
```

```
↳ /content/aeroplane_detection/data
```

```
▶ # where the model will be saved at each checkpoint while training
model_dir = 'training/'

# Optionally: remove content in output model directory to fresh start.
!rm -rf {model_dir}
os.makedirs(model_dir, exist_ok=True)
model_dir
```

```
↳ 'training/'
```

Figure 5.3.1 store the checkpoint under training folder

2. Now we need to edit config file as per the model that is selected, here I have selected the SSD, so that should be edited.
3. Before editing the config file base to downloaded .

```
▶ # downloading the base model
%cd /content/aeroplane_detection/models/research

# Name of the object detection model to use.
MODEL = MODELS_CONFIG[selected_model]['model_name']

# Name of the pipeline file in tensorflow object detection API.
pipeline_file = MODELS_CONFIG[selected_model]['pipeline_file']

#selecting the model
MODEL_FILE = MODEL + '.tar.gz'

#creating the download link for the model selected
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object\_detection/'

#the destination folder where the model will be saved
fine_tune_dir = '/content/aeroplane\_detection/models/research/pretrained\_model'
```

```

#checks if the model has already been downloaded
if not (os.path.exists(MODEL_FILE)):
    urllib.request.urlretrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)

#unzipping the file and extracting its content
tar = tarfile.open(MODEL_FILE)
tar.extractall()
tar.close()

# creating an output file to save the model while training
os.remove(MODEL_FILE)
if (os.path.exists(fine_tune_dir)):
    shutil.rmtree(fine_tune_dir)
os.rename(MODEL, fine_tune_dir)

```

📁 /content/aeroplane_detection/models/research

Figure 5.3.2 download base model

```

#checking the content of the pretrained model.
# this is the directory of the "fine_tune_checkpoint" that is used in the config file.
!echo {fine_tune_dir}
!ls -alh {fine_tune_dir}

```

📁 /content/aeroplane_detection/models/research/pretrained_model

```

total 135M
drwxr-xr-x  3 345018 89939 4.0K Mar 30 2018 .
drwxr-xr-x 24 root    root  4.0K Aug  2 06:43 ..
-rw-r--r--  1 345018 89939   77 Mar 30 2018 checkpoint
-rw-r--r--  1 345018 89939  67M Mar 30 2018 frozen_inference_graph.pb
-rw-r--r--  1 345018 89939  65M Mar 30 2018 model.ckpt.data-00000-of-00001
-rw-r--r--  1 345018 89939  15K Mar 30 2018 model.ckpt.index
-rw-r--r--  1 345018 89939  3.4M Mar 30 2018 model.ckpt.meta
-rw-r--r--  1 345018 89939  4.2K Mar 30 2018 pipeline.config
drwxr-xr-x  3 345018 89939 4.0K Mar 30 2018 saved_model

```

Figure 5.3.3 checking the pretrained model

```

# configuring the training the pipe line
#the path to the folder containing all the sample config files
CONFIG_BASE = "/content/aeroplane_detection/models/research/object_detection/samples/configs/"

#path to the specified model's config file
model_pipeline = os.path.join(CONFIG_BASE, pipeline_file)
model_pipeline

📁 '/content/aeroplane_detection/models/research/object_detection/samples/configs/ssd_mobilenet_v2_coco.config'

```

Figure 5.3.4 config_Base

4. Editing the config file as per the requirements of the model.

```

▶ #editing the configuration file to add the path for the TFRecords files, pbtxt, batch_size, num_steps, num_classes.
# any image augmentation, hyperparameter tuning (drop out, batch normalization... etc) would be edited here

%%writefile {model_pipeline}
model {
  ssd {
    num_classes: 1 # number of classes to be detected
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
  }
}

```

```

  anchor_generator {
    ssd_anchor_generator {
      num_layers: 6
      min_scale: 0.2
      max_scale: 0.95
      aspect_ratios: 1.0
      aspect_ratios: 2.0
      aspect_ratios: 0.5
      aspect_ratios: 3.0
      aspect_ratios: 0.3333
    }
  }
  # all images will be resized to the below W x H.
  image_resizer {
    fixed_shape_resizer {
      height: 300
      width: 300
    }
  }
}

```

```

box_predictor {
  convolutional_box_predictor {
    min_depth: 0
    max_depth: 0
    num_layers_before_predictor: 0
    #use_dropout: false
    use_dropout: true # to counter over fitting. you can also try tweaking its probability below
    dropout_keep_probability: 0.8
    kernel_size: 1
    box_code_size: 4
    apply_sigmoid_to_scores: false
    conv_hyperparams {
      activation: RELU_6,
      regularizer {
        l2_regularizer {
          # weight: 0.00004
          weight: 0.001 # higher regularization to counter overfitting
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
    }
  }
}

```

```

    batch_norm {
      train: true,
      scale: true,
      center: true,
      decay: 0.9997,
      epsilon: 0.001,
    }
  }
}
feature_extractor {
  type: 'ssd_mobilenet_v2'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        # weight: 0.00004
        weight: 0.001 # higher regularization to counter overfitting
      }
    }
    initializer {
      truncated_normal_initializer {
        stddev: 0.03
        mean: 0.0
      }
    }
  }
}

```

```

        batch_norm {
            train: true,
            scale: true,
            center: true,
            decay: 0.9997,
            epsilon: 0.001,
        }
    }
}
loss {
    classification_loss {
        weighted_sigmoid {
        }
    }
    localization_loss {
        weighted_smooth_l1 {
        }
    }
}
hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.95
    loss_type: CLASSIFICATION
    max_negatives_per_positive: 3
    min_negatives_per_image: 3
}
classification_weight: 1.0
localization_weight: 1.0

```

```

}
normalize_loss_by_num_matches: true
post_processing {
    batch_non_max_suppression {
        score_threshold: 1e-8
        iou_threshold: 0.6

        #adjust this to the max number of objects per class.
        # ex, in my case, i have one pistol in most of the images.
        # . there are some images with more than one up to 16.
        max_detections_per_class: 16
        # max number of detections among all classes. I have 1 class only so
        max_total_detections: 16
    }
    score_converter: SIGMOID
}
}
}

```

```

train_config: {
  batch_size: 16 # training batch size
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.003
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
    }
    momentum_optimizer_value: 0.9
    decay: 0.9
    epsilon: 1.0
  }
}

#the path to the pretrained model.
fine_tune_checkpoint: "/content/aeroplane_detection/models/research/pretrained_model/model.ckpt"
fine_tune_checkpoint_type: "detection"
# Note: The below line limits the training process to 200K steps, which we
# empirically found to be sufficient enough to train the pets dataset. This
# effectively bypasses the learning rate schedule (the learning rate will
# never decay). Remove the below line to train indefinitely.
num_steps: 70000

```

```

#data augmentation is done here, you can remove or add more.
# They will help the model generalize but the training time will increase greatly by using more data augmentation.
# Check this link to add more image augmentation: https://github.com/tensorflow/models/blob/master/research/object\_detection/protos/preprocessor.proto

data_augmentation_options {
  random_horizontal_flip {
  }
}
data_augmentation_options {
  random_adjust_contrast {
  }
}
data_augmentation_options {
  ssd_random_crop {
  }
}
}

train_input_reader: {
  tf_record_input_reader {
    #path to the training TFRecord
    input_path: "/content/aeroplane_detection/data/train_labels.record"
  }
  #path to the label map
  label_map_path: "/content/aeroplane_detection/data/label_map.pbtxt"
}

```

```

eval_config: {
  # the number of images in your "testing" data (was 200 but we removed one above :) )
  num_examples: 200
  # the number of images to display in Tensorboard while training
  num_visualizations: 20

  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  #max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    #path to the "/content/aeroplane_detection/data/test_labels.record (ctrl + click)"
    input_path: "/content/aeroplane_detection/data/test_labels.record"
  }
  #path to the label map
  label_map_path: "/content/aeroplane_detection/data/label_map.pbtxt"
  shuffle: false
  num_readers: 1
}

```

Overwriting {model_pipeline}

- Changes to be made:
- Change the number of classes, change extractor type, change the path for the fine tune check point.
- Add the path for the tf record and label map path for both train and test.
- Number of evaluations to be changed and the no of evaluations for the images to be changes while executing the cell.
- Change the no of steps to evaluate.

5.4 TRAIN YOUR MODEL:

- Once all the above steps are done then you are ready to run the model finally.
- For training we use the mode_main.py which is present in the object detection folder.
- This mode_main.py training and evaluation is done simultaneously.
- The training should be done more than 50,000 steps to get loss less than 2.0.
- But here I am training my model only for 50,000 steps due to the limit of colab that is for 12 hrs .
- I had got the final loss as 1.840329 , I took nearly seven hrs for training and evaluating.
- I the model to be only trained the we have the train.py instead of model_main.py.
- The train.py will be in legacy folder under the research/objectdetection/legacy.

TRAINING

```

: 1 model_dir
: 'training/'

: 1 model_pipeline
: '/content/aeroplane_detection/models/research/object_detection/samples/configs/ssd_mobilenet_v2_coco.config'

: 1 # training the data and saved it into model_dir
: 2 !python3 /content/aeroplane_detection/models/research/object_detection/model_main.py \
: 3     --pipeline_config_path={model_pipeline}\
: 4     --model_dir={model_dir} \
: 5     --alsologtostderr \
: 6

```

Figure 5.4.1 training the model


```

2020-07-25 13:55:36.151530: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1199] 0:  N
2020-07-25 13:55:36.151702: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS
had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2020-07-25 13:55:36.152359: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:983] successful NUMA node read from SysFS
had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2020-07-25 13:55:36.152936: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1325] Created TensorFlow device (/job:localhos
t/replica:0/task:0/device:GPU:0 with 10805 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 0000:00:04.0,
compute capability: 3.7)
INFO:tensorflow:Restoring parameters from training/model.ckpt-50000
I0725 13:55:36.155870 139696487446400 saver.py:1284] Restoring parameters from training/model.ckpt-50000
INFO:tensorflow:Assets added to graph.
I0725 13:55:36.621932 139696487446400 builder_impl.py:665] Assets added to graph.
INFO:tensorflow:No assets to write.
I0725 13:55:36.622152 139696487446400 builder_impl.py:460] No assets to write.
INFO:tensorflow:SavedModel written to: training/export/Servo/temp-b'1595685332'/saved_model.pb
I0725 13:55:37.462962 139696487446400 builder_impl.py:425] SavedModel written to: training/export/Servo/temp-b'1595685332'/sa
ved_model.pb
INFO:tensorflow:Loss for final step: 1.8403296.
I0725 13:55:37.839620 139696487446400 estimator.py:371] Loss for final step: 1.8403296.

```

Figure 5.4.2 output of the training

As observed above the training took 50,000 steps after the execution and got a loss of 1.8403296 finally. Which is less than 2.0

Now we can check the tensor board for the visualization, they are scalar graphs and images.

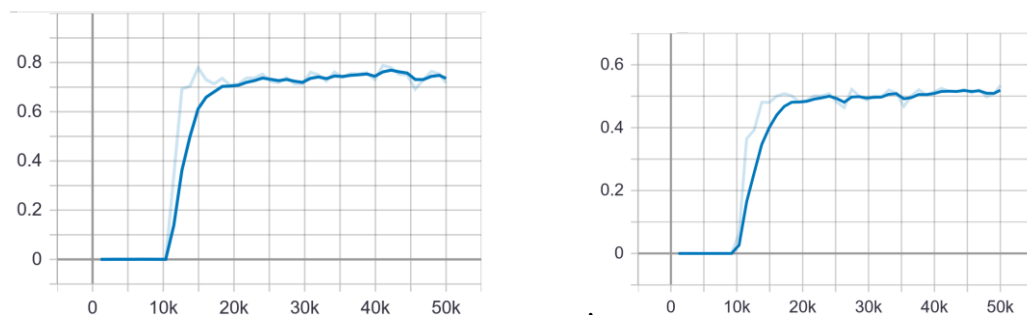


Figure 5.4.3 DetectionBoxes_Precision_mAP (large)

Figure 5.4.4 DetectionBoxes_Precision_mAP (medium)

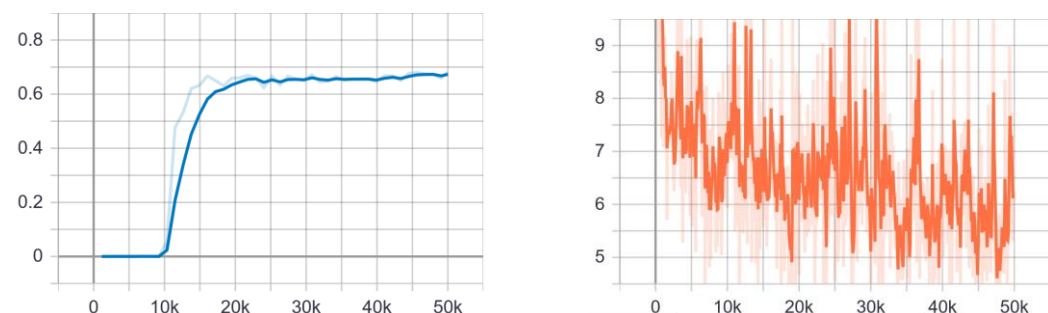


Figure 5.4.5 DetectionBoxes_Precision_mAP@.50IOU

Figure 5.4.6 global_norm_gradient_norm

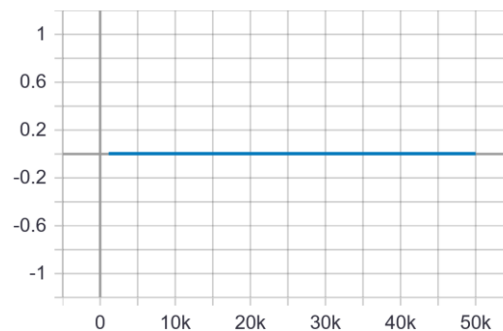
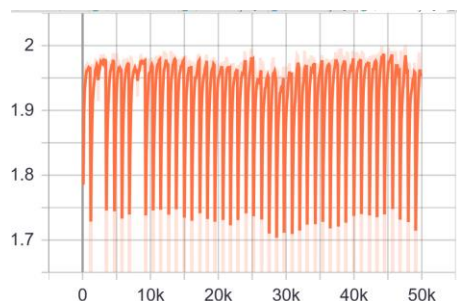


Figure 5.4.7 global_step_sec

Figure 5.4.8 learning_rate

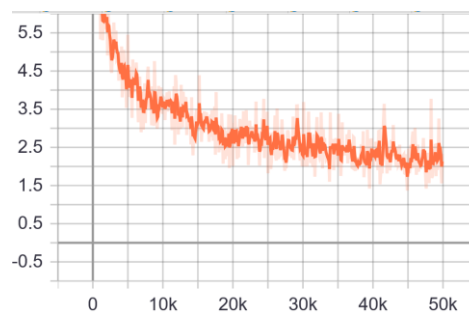
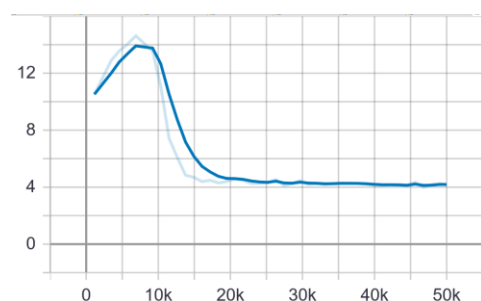


Figure 5.4.9 loss

Figure 5.4.10 loss_1

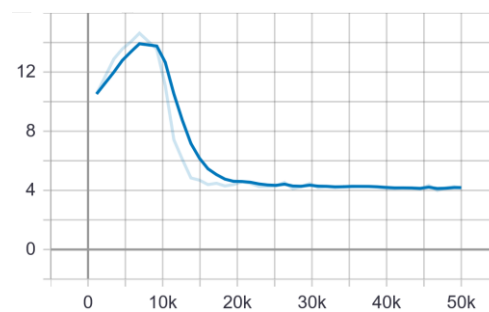
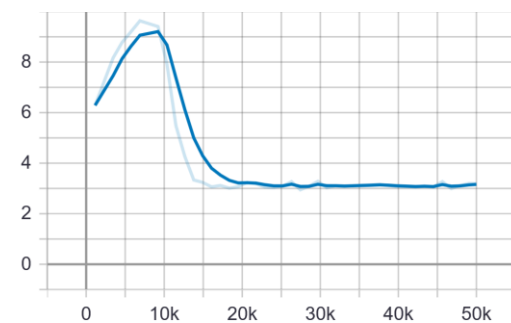


Figure 5.4.11 Loss_classification_loss

Figure 5.4.12 Loss_total_loss

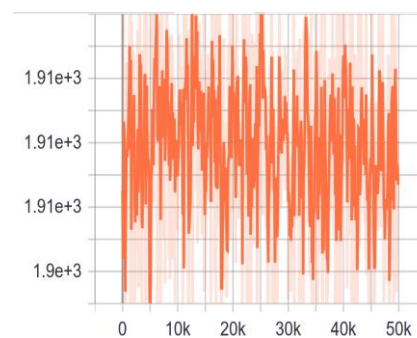
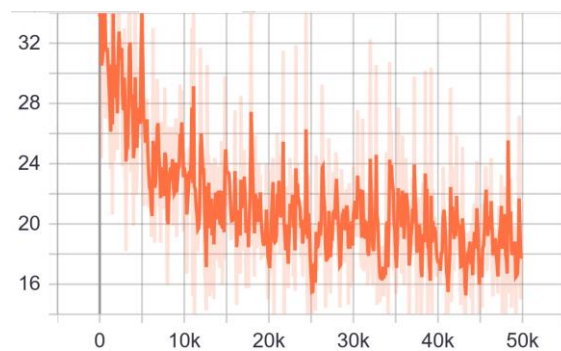


Figure 5.4.13 Loss_HardExampleMiner_NumNegatives

Figure 5.4.14 TargetAssignment_Loss_TargetAssignment_Avg

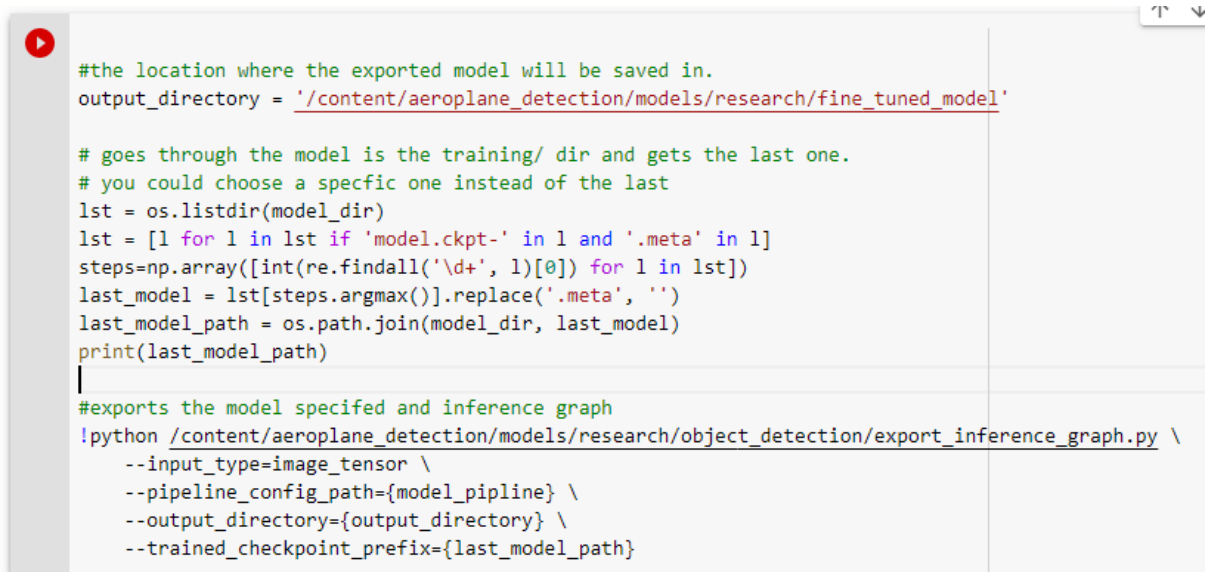
CHAPTER 6

TEST YOUR MODEL

- After the whole training part is over we must test the model by giving our own input images to check whether the required objects are getting detected or not.

6.1 EXPORT THE TRAIN MODEL:

- export the model specified and inference graphcontent/aeroplane_detection/models/research/object_detection folder.



```
#the location where the exported model will be saved in.
output_directory = '/content/aeroplane_detection/models/research/fine_tuned_model'

# goes through the model is the training/ dir and gets the last one.
# you could choose a specific one instead of the last
lst = os.listdir(model_dir)
lst = [l for l in lst if 'model.ckpt-' in l and '.meta' in l]
steps=np.array([int(re.findall('\d+', l)[0]) for l in lst])
last_model = lst[steps.argmax()].replace('.meta', '')
last_model_path = os.path.join(model_dir, last_model)
print(last_model_path)

#exports the model specified and inference graph
!python /content/aeroplane_detection/models/research/object_detection/export_inference_graph.py \
  --input_type=image_tensor \
  --pipeline_config_path={model_pipeline} \
  --output_directory={output_directory} \
  --trained_checkpoint_prefix={last_model_path}
```

Figure 6.1.1 exporting the trained model

```

This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info
or tf.compat.v1.saved_model.build_tensor_info.
W0725 13:58:48.085387 139719421781888 deprecation.py:323] From /content/aeroplane_detection/models/research/object_detection/
exporter.py:384: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a fut
ure version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info
or tf.compat.v1.saved_model.build_tensor_info.
INFO:tensorflow:No assets to save.
I0725 13:58:48.086369 139719421781888 builder_impl.py:640] No assets to save.
INFO:tensorflow:No assets to write.
I0725 13:58:48.086540 139719421781888 builder_impl.py:460] No assets to write.
INFO:tensorflow:SavedModel written to: /content/aeroplane_detection/models/research/fine_tuned_model/saved_model/saved_model.
pb
I0725 13:58:48.381668 139719421781888 builder_impl.py:425] SavedModel written to: /content/aeroplane_detection/models/researc
h/fine_tuned_model/saved_model/saved_model.pb
INFO:tensorflow:Writing pipeline config file to /content/aeroplane_detection/models/research/fine_tuned_model/pipeline.config
I0725 13:58:48.407660 139719421781888 config_util.py:254] Writing pipeline config file to /content/aeroplane_detection/model
s/research/fine_tuned_model/pipeline.config

```

- we need to check the model_dir files. whether the checkpoints are saved or not.

```

1 # checcking the files
2 lst = os.listdir(model_dir)
3 lst

['model.ckpt-45640.index',
'model.ckpt-49054.data-00000-of-00001',
'model.ckpt-45640.meta',
'model.ckpt-45640.data-00000-of-00001',
'model.ckpt-46783.meta',
'model.ckpt-50000.index',
'model.ckpt-46783.data-00000-of-00001',
'model.ckpt-47922.data-00000-of-00001',
'checkpoint',
'model.ckpt-49054.meta',
'model.ckpt-50000.data-00000-of-00001',
'graph.pbtxt',
'model.ckpt-49054.index',
'model.ckpt-50000.meta',
'events.out.tfevents.1595658966.fa9e4854495f',
'model.ckpt-47922.meta',
'eval_0',
'model.ckpt-46783.index',
'export',
'model.ckpt-47922.index']

```

- check whether the finetuned model is present or not .It shows true when it is present otherwise false.

```

1 # checking whether the fine_tuned model present or not
2 import os
3
4 os.path.isdir('./fine_tuned_model')

True

```

Figure 6.1.2 checking finetuned model

- we need to download the three files , to do further testing of the model.

- They are frozen_inference_graph.pb , label_map.pbtxt, model.ckpt.data-00000-of-00001.

```

1 #downloads the frozen model that is needed for inference
2 files.download(output_directory + '/frozen_inference_graph.pb')

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

1 !zip /content/fine_tuned_model.zip /content/aeroplane_detection/models/research/fine_tuned_model
adding: content/aeroplane_detection/models/research/fine_tuned_model/ (stored 0%)

1 #download the label map
2 files.download(DATA_BASE_PATH + '/label_map.pbtxt')

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

1 files.download('/content/aeroplane_detection/models/research/fine_tuned_model/model.ckpt.data-00000-of-00001')

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

```

Figure 6.1.3 downloading the three files

6.2 RUN THE OBJECT DETECTION MODEL:

- To run the object detection model we need to load the images .

```

1 # defining the fuction for loading the images
2 def load_image_into_numpy_array(image_path):
3     img_data = tf.io.gfile.GFile(image_path, 'rb').read()
4     image = Image.open(BytesIO(img_data))
5     (im_width, im_height) = image.size
6     return np.array(image.getdata()).reshape(
7         (im_height, im_width, 3)).astype(np.uint8)

```

Figure 6.2.1 loading the images in the numpy array

```

1 # creating the file path for testing images
2 PATH_TO_TEST_IMAGES_DIR='/content/aeroplane_detection/models/research/object_detection/test_images/aeroplane'
3 TEST_IMAGE_PATHS = [os.path.join(PATH_TO_TEST_IMAGES_DIR,name) for name in os.listdir(PATH_TO_TEST_IMAGES_DIR)]
4
5 IMAGE_SIZE=(12,8)
6 TEST_IMAGE_PATHS

['/content/aeroplane_detection/models/research/object_detection/test_images/aeroplane/airplane_035.jpg',
'/content/aeroplane_detection/models/research/object_detection/test_images/aeroplane/airplane_028.jpg',
'/content/aeroplane_detection/models/research/object_detection/test_images/aeroplane/airplane_039.jpg',
'/content/aeroplane_detection/models/research/object_detection/test_images/aeroplane/airplane_037.jpg',
'/content/aeroplane_detection/models/research/object_detection/test_images/aeroplane/airplane_047.jpg',

```

Figure 6.2.2 to upload the images.

- Under the object detection/test_images we need to create the some folder to upload the images for detecting.
- To proceed to further execution we need import particular libraries.


```

1  # importing the packages
2  import numpy as np
3  import os
4  import six.moves.urllib as urllib
5  import sys
6  import tarfile
7  import tensorflow as tf
8  import zipfile
9
10 from collections import defaultdict
11 from io import StringIO
12 from matplotlib import pyplot as plt
13 from PIL import Image
14 from IPython.display import display
15 from six import BytesIO

```

```

1  # import the packages
2  from object_detection.utils import ops as utils_ops
3  from object_detection.utils import label_map_util
4  from object_detection.utils import visualization_utils as vis_util

```

Figure 6.2.3 importing the libraries

- We need to give the path for the check point frozen inference graph to test the images.

```

1  # path to ckpt
2  PATH_TO_CKPT='/content/aeroplane_detection/models/research/fine_tuned_model/frozen_inference_graph.pb'

```

```

1  # detecting the graph
2  detection_graph=tf.Graph()
3  with detection_graph.as_default():
4      od_graph_def=tf.GraphDef()
5      with tf.gfile.GFile(PATH_TO_CKPT,'rb') as fid:
6          serialized_graph = fid.read()
7          od_graph_def.ParseFromString(serialized_graph)
8          tf.import_graph_def(od_graph_def,name='')

```

```

1  # display of the catogorical index
2  category_index = {
3      1: {'id': 1, 'name': 'aeroplane','display name':'aeroplane'} }

```

Figure 6.2.4 detection of the graph and categorical index

- Now the final code to be executed to detect the object.

```

1  # final output for the prediction of the objection detection wher it could it detect the eroplane
2  with detection_graph.as_default():
3      with tf.Session(graph=detection_graph) as sess:
4          for image_path in TEST_IMAGE_PATHS:
5              image_np=load_image_into_numpy_array(image_path)
6              image_np_expanded = np.expand_dims(image_np,axis=0)
7              image_tensor=detection_graph.get_tensor_by_name('image_tensor:0')
8              boxes=detection_graph.get_tensor_by_name('detection_boxes:0')
9              scores=detection_graph.get_tensor_by_name('detection_scores:0')
10             classes=detection_graph.get_tensor_by_name('detection_classes:0')
11             num_detections = detection_graph.get_tensor_by_name('num_detections:0')
12             image_np_with_detections = image_np.copy()
13
14             (boxes,scores,classes,num_detections) = sess.run(
15                 [boxes,scores,classes,num_detections],
16                 feed_dict={image_tensor:image_np_expanded})
17             print(scores)
18             vis_util.visualize_boxes_and_labels_on_image_array(
19                 image_np_with_detections,
20                 boxes[0],
21                 classes[0].astype(np.int32),
22                 scores[0],
23                 category_index,
24                 use_normalized_coordinates=True,
25                 min_score_thresh=.40,
26                 agnostic_mode=False,
27                 line_thickness=5)
28
29             plt.figure(figsize=IMAGE_SIZE)
30             plt.imshow(image_np_with_detections)
31             #print(boxes)
32
33

```

Figure 6.2.5 code to detect the object

6.3 Output of the model :



Figure 6.3.1 image 1



Figure 6.3.2 image 2

CHAPTER 7

OBJECT DETECTION

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc.

Each of these "eternal" problems is solved by trial and error by the efforts of numerous groups of the engineers and scientists. As modern technical solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required toolkit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows tool kit supports the creation of interfaces for solving various applied problems.

Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is refers to identifying the location of one or more objects in an image and drawing an a bounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term “object recognition”, they often mean “object detection”.

- There are many algorithms used for the object detection, they are:

7.1 SSD (Single Shot Detector):

- This is one of the algorithm's, where it's used for the object detection. It detects the object using a single deep neural network.
- The SSD approach discretises the output space of bounding boxes into a set of default boxes over different aspect ratios. After discretising, the method scales per feature map

location. The Single Shot Detector network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

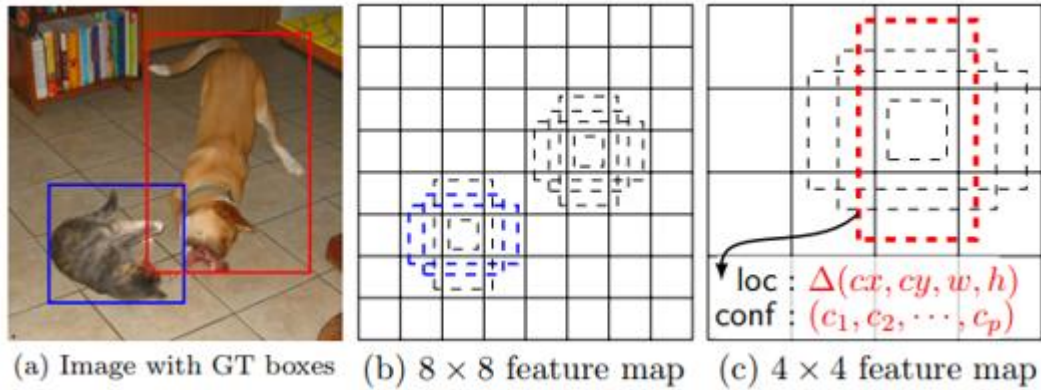


Figure 7.1.1 SSD

- This approach uses a feed-forward convolutional neural network that produces a collection of bounding boxes and scores for the presence of certain objects. Convolutional feature layers are added to allow for feature detection at multiple scales. In this model, each feature map cell is linked to a set of default bounding boxes.

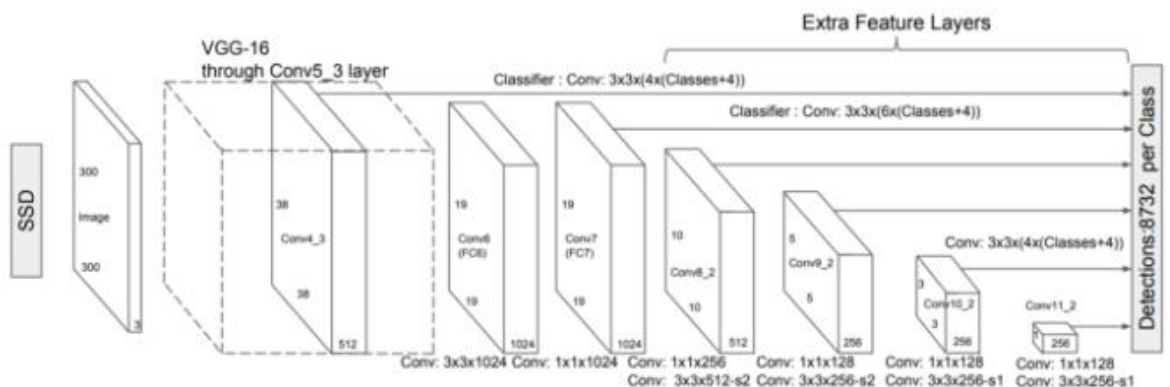


Figure 7.1.2 default boundary boxes

- SSD predictions are classified as **positive** matches or negative matches. SSD only uses positive matches in calculating the localization cost (the mismatch of the boundary box). If the corresponding default boundary box (not the predicted boundary box) has an IoU greater than 0.5 with the ground truth, the match is positive. Otherwise, it is negative. (IoU, the intersection over the union, is the ratio between the intersected area over the joined area for two regions.)

- The **localization loss** is the mismatch between the ground truth box and the predicted boundary box. SSD only penalizes predictions from positive matches. We want the predictions from the positive matches to get closer to the ground truth. Negative matches can be ignored.

The localization loss between the predicted box l and the ground truth box g is defined as the smooth L1 loss with cx, cy as the offset to the default bounding box d of width w and height h .

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx}) / d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy}) / d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \quad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

$$x_{ij}^p = \begin{cases} 1 & \text{if } IoU > 0.5 \text{ between default box } i \text{ and ground true box } j \text{ on class } p \\ 0 & \text{otherwise} \end{cases}$$

Figure 7.1.3 loss function

- The **confidence loss** is the loss in making a class prediction. For every positive match prediction, we penalize the loss according to the confidence score of the corresponding class. For negative match predictions, we penalize the loss according to the confidence score of the class “0”: class “0” classifies no object is detected.

It is calculated as the softmax loss over multiple classes confidences c (class score).

$$L_{conf}(x, c) = - \sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

where N is the number of matched default boxes.

The final loss function is computes as:

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where N is the number of positive match and α is the weight for the localization loss.

- **Advantages of ssd:**
- SSD completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network.

- Easy to train and straightforward to integrate into systems that require a detection component.
- SSD has competitive accuracy to methods that utilize an additional object proposal step, and it is much faster while providing a unified framework for both training and inference.

7.2 Faster R-CNN:

Towards Real-Time Object Detection with Region Proposal Networks State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations.

This paper proposes a training mechanism that alternates fine-tuning for regional proposal tasks and fine-tuning for object detection.

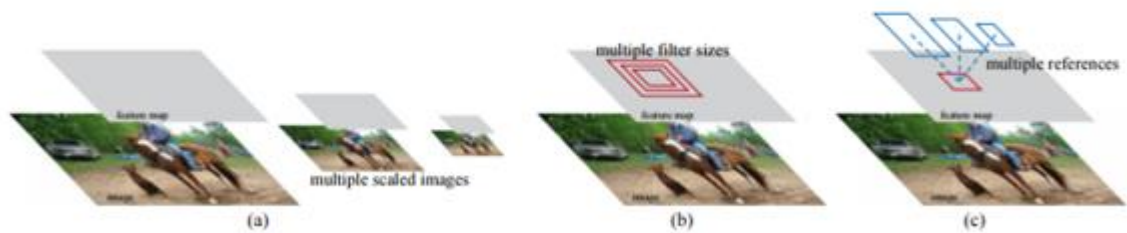


Figure 7.2.1 Faster R-CNN(1)

The Faster R-CNN model is comprised of two modules: a deep convolutional network responsible for proposing the regions, and a Fast R-CNN detector that uses the regions. The Region Proposal Network takes an image as input and generates an output of rectangular object proposals. Each of the rectangles has an objectness score.

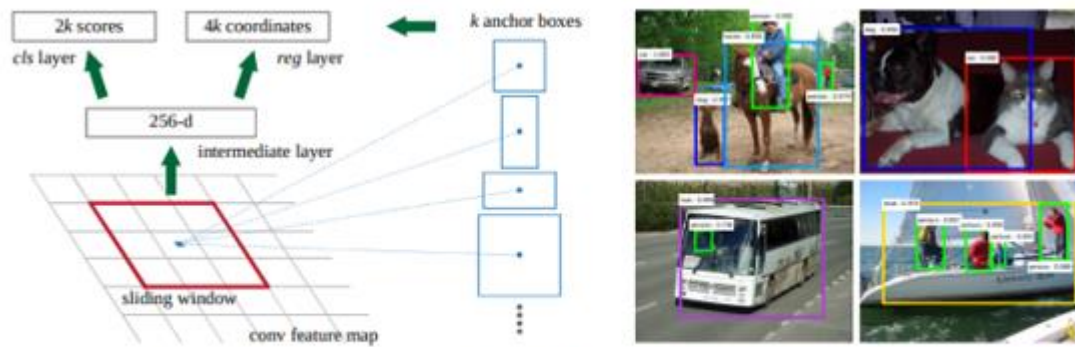


Figure 3: **Left:** Region Proposal Network (RPN). **Right:** Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Figure Faster RCNN (2)

7.3 You Only Look Once(YOLO):

This paper proposes a single neural network to predict bounding boxes and class probabilities from an image in a single evaluation.

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform...

The YOLO models process 45 frames per second in real-time. YOLO views image detection as a regression problem, which makes its pipeline quite simple. It's extremely fast because of this simple pipeline.

It can process a streaming video in real-time with a latency of less than 25 seconds. During the training process, YOLO sees the entire image and is, therefore, able to include the context in object detection.

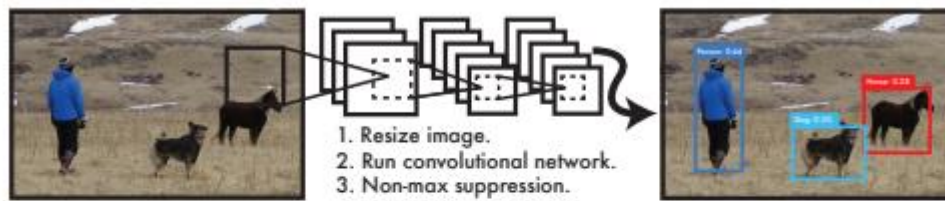


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Figure 7.3.1 YOLO(1)

In YOLO, each bounding box is predicted by features from the entire image. Each bounding box has 5 predictions; x , y , w , h , and confidence. (x, y) represents the center of the bounding box relative to the bounds of the grid cell. W and h are the predicted width and height of the whole image.

This model is implemented as a convolutional neural network and evaluated on the PASCAL VOC detection dataset. The convolutional layers of the network are responsible for extracting the features, while the fully connected layers predict the coordinates and output probabilities.

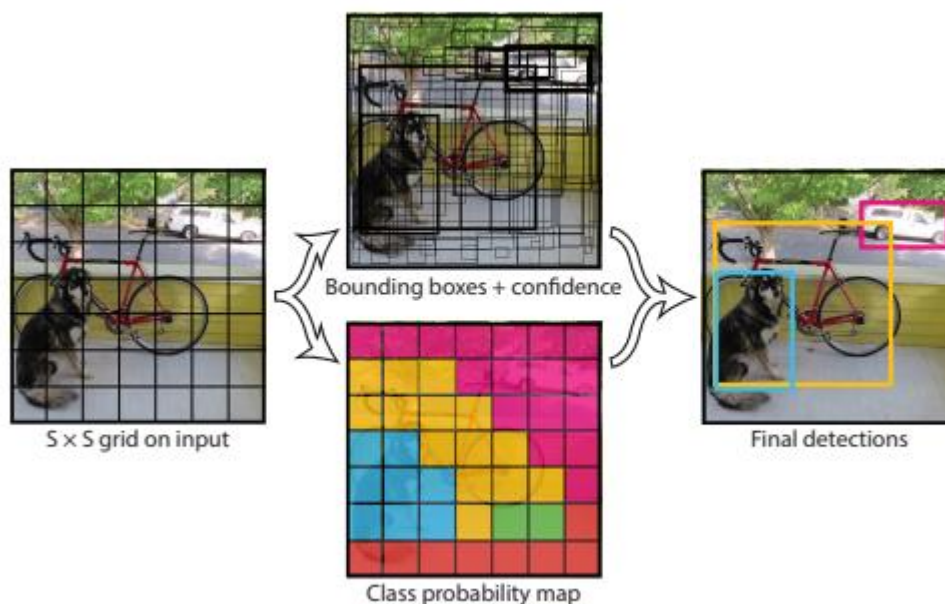


Figure 7.3.2 YOLO (2)

CONCLUSION:

- It is concluded that the object detection model of detecting the aeroplane has been successfully done. The training of the model has been done in 50,000 and got a loss of 1.8403296. This model is helpful for the airplane management system.
- The SSD is used because we will get more accurate results compared with other models like FASTER RCNN and YOLO.

REFERENCE:

- <https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/>
- <https://towardsdatascience.com/detailed-tutorial-build-your-custom-real-time-object-detector-5ade1017fd2d#5f34>
- <https://stackabuse.com/object-detection-with-imageai-in-python/>
- https://www.tensorflow.org/lite/models/object_detection/overview

GITHUB LINK:

- <https://github.com/Motupallysundaracharya38/OBJECTDETECTION>

