HAND GESTURE RECOGNITION

USING TENSORFLOW

**A Project Report Submitted in Partial Fulfilment of the**

**Requirement for the Degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**INFORMATION TECHNOLOGY**

**By**

**Group No. : 7**

**JANHAVI OJHA (14800218038)**

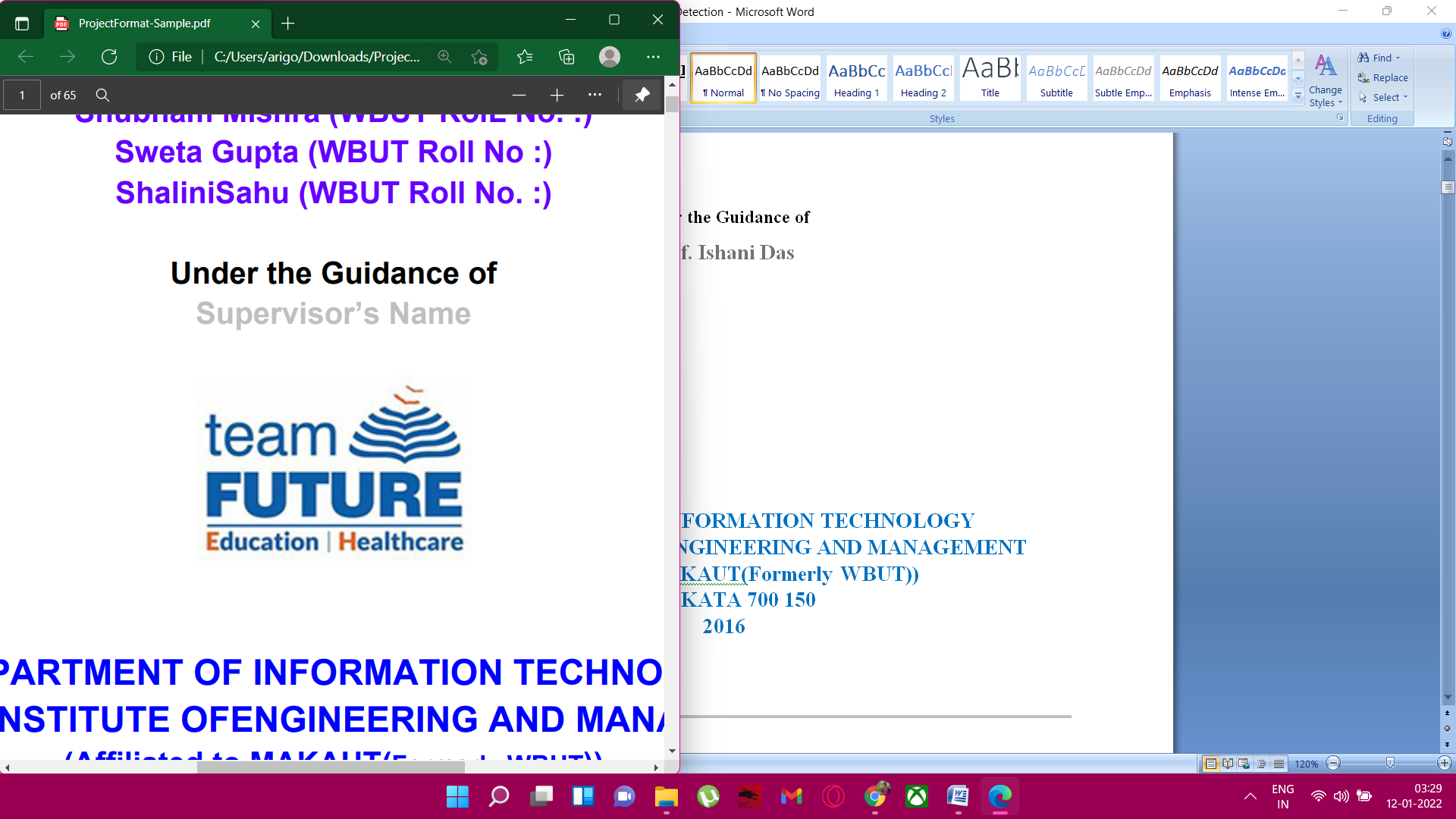
**SAMRIDHI KUMARI (14800218024)**

**SARFARAZ AHMAD (14800218023)**

**ARIJEET DASGUPTA (14800218050)**

**Under the Guidance of**

**PROF. ISHANI DAS**

****

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**FUTURE INSTITUTE OFENGINEERING AND MANAGEMENT**

**(Affiliated to MAKAUT (Formerly WBUT))**

**KOLKATA 700 150**

**2022**

Department of Information Technology

FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT

Sonarpur Station Road, Kolkata – 700150

Tel: 033-2434 5640 (Extn. – 238) URL: www.futureengineering.in



# CERTIFICATE

*We do hereby declaring that the work which is being presented in the Project Report entitled* ***“HAND GESTURE RECOGNITION”,*** *in partial fulfilment of the requirements for the award of the* ***Bachelor of Technology*** *in* ***Information Technology*** *and submitted to the* ***Department of Information Technology*** *of* ***Future Institute of Engineering and Management, Kolkata,*** *is an authentic record of our own work carried out during the period of 7th Semester, under the supervision of* ***Prof.******Ishani Das****.*

*The matter presented in this thesis has not been submitted by us for the award of any other degree elsewhere.*

*Full* *Signature of the Students(s)*

a) Janhavi Ojha

b) Arijeet Dasgupta

c) Sarfaraz Ahmad

d) Samridhi Kumari

*This is to certify that the above statement made by the students, is correct to the best of my knowledge.*

#### *Date:12 January’22*

#### *Prof. Ishani Das*

#### *Signature of the Supervisor*

#### 

**Head *Signature of the External Examiner/***

*Department of Information Technology* ***Panel Members***

**Future Institute of Engineering and Management**

**Kolkata, WB**

TABLE OF CONTENTS

**CHAPTER 1. INTRODUCTION4-6**

1.1 BRIEF INTRODUCTION4

1.1.1 Learning5

1.1.2 Detection5

1.1.3 Recognition5

1.2 MOTIVATION5

1.2.1 Problem Statement5

1.2.2 Solution Proposed6

1.3 OUR MISSION6

**CHAPTER2. IMPLEMENTATION DETAILS7-11**

2.1 INTRODUCTION7

2.1.1 Why SSD7

2.1.2 Why MobileNet8

2.1.3 Why FPNLite8

2.2 ALGORITHM9

2.2.1 Grid Cells10

2.2.2 Anchor Box10

2.2.3 Aspect Ratio10

2.2.4 Zoom Level11

**CHAPTER 3. CODE12-14**

**CHAPTER 4. FUTURE SCOPE15**

**CHAPTER 5. CONCLUSION16**

**CHAPTER 6. BIBLIOGRAPHY17**

**ACKNOWLEDGEMENT**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them. We are highly indebted to our guide Prof. Ishani Das for her guidance and constant supervision as well as for providing necessary information regarding the project and also for her support in completing the project.

We express our thanks to our Principal Dr. Aloke Ghosh for extending his support. We would also thank our Institution and the faculty members without whom this project would have been a distant reality. Our thanks and appreciations also go to all people who have willingly helped us out with their abilities.

**Janhavi Ojha**

**Samridhi Kumari**

**Sarfaraz Ahmad**

**Arijeet Dasgupta**

INTRODUCTION

1.1 BRIEF INTRODUCTION

Gesture recognition is a technique which is used to understand and analyze the human body language and interact with the user accordingly. This in turn helps in building a bridge between the machine and the user to communicate with each other. Gesture recognition is useful in processing the information which cannot be conveyed through speech or text. Gestures are the simplest means of communicating something that is meaningful.

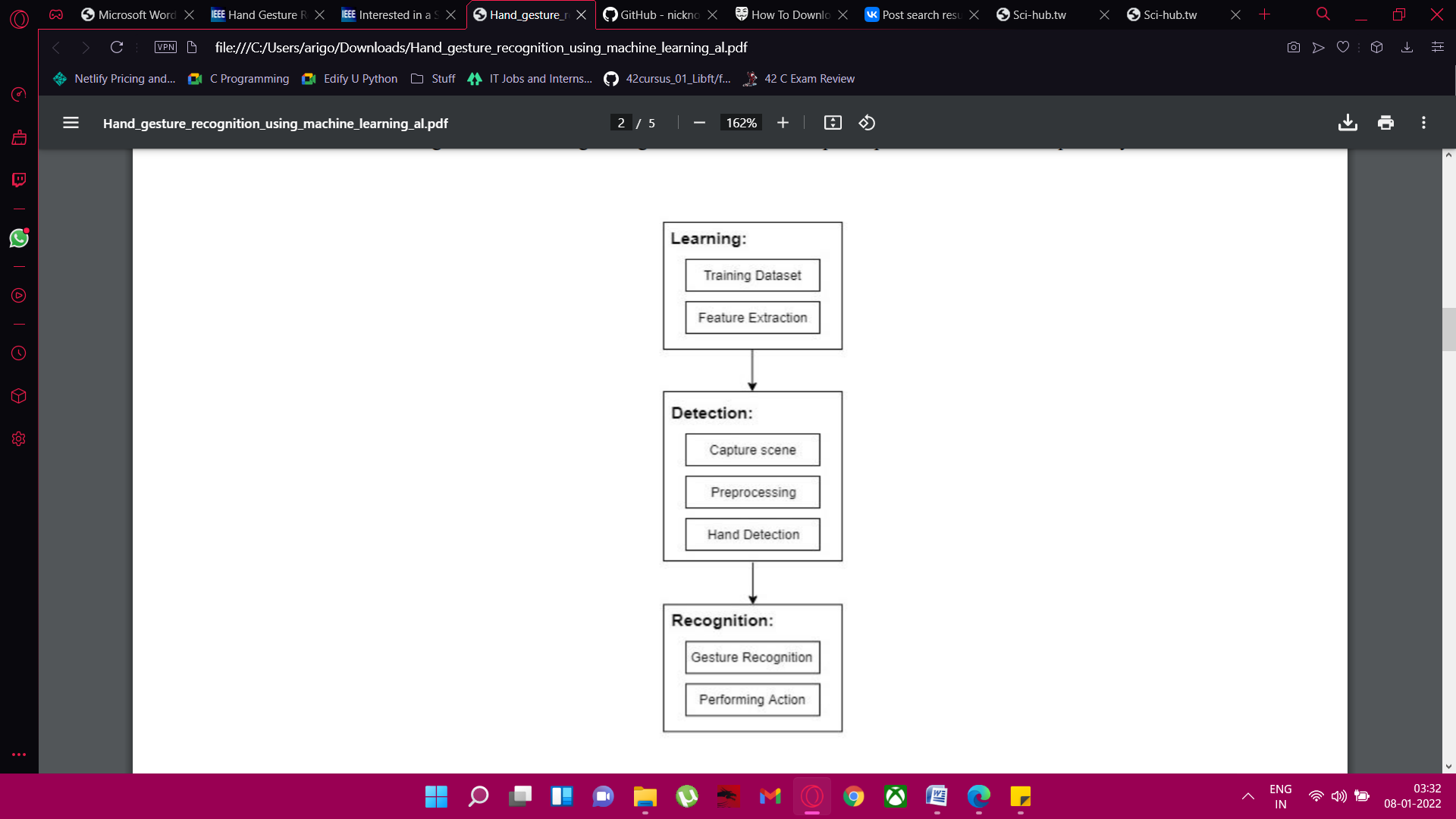


Figure 1. Flowchart human computer interaction

This project involves implementation of the system that aims to design a vision-based hand gesture recognition system with a high correct detection rate along with a high-performance criterion, which can work in a real time Human Computer Interaction system without having any of the limitations (gloves, uniform background etc.) on the user environment. The system can be defined using a flowchart that contains three main steps, they are: Learning, Detection, Recognition as shown in Figure 1.

* + - * **Learning**

**Training dataset**

This is the dataset that consists of different types of hand gestures that are used to train− the system based on which the system performs the actions.

**Feature Extraction**: It involves determining the centroid that divides the image into two halves at its− geometric Centre.

* + - * **Detection**

**Capture scene:** Captures the images through a web camera, which is used as an input to the system.

**Pre-processing:** Images that are captured through the webcam are compared with the dataset to recognize− the valid hand movements that are needed to perform the required actions.

**Hand Detection:** The requirements for hand detection involve the input image from the webcam. The image should be fetched with a speed of 20 frames per second. Distance should also be maintained between the hand and the camera. Approximate distance that should be between hands the camera is around 30 to 100 cm. The video input is stored frame by frame into a matrix after pre-processing.

* + - * **Recognition**

**Gesture Recognition**: The number of fingers present in the hand gesture is determined by making use of− defect points present in the gesture. The resultant gesture obtained is fed through a 3Dimensional Convolutional Neural Network consecutively to recognize the current gesture.

**Performing action**: The recognized gesture is used as an input to perform the actions required by the user.

**1.2 MOTIVATION**

* + - * **Problem Statement**

With the onset of the corona virus pandemic since 2020, the use of internet has increased to a great extent for almost all our day to day life activities. Unfortunately, the internet has not been designed in a way that makes it easy for people with speech and hearing impairment to use. Almost every other formal institution, be it school, colleges or offices have transitioned into an online model where a majority of their students/employees have to use video conferencing on a daily basis. In normal times, they were provided with translators who could’ve translated their sign languages to other people conveying what they want to say, but with limited screen size, it becomes excessively difficult to carry out the same process in an online meeting. It becomes a tedious and time consuming affair. In the current scenario, people who have speech impairment issues usually use the built-in chatbox to deliver their message. But, this leads to cluttered inbox. Also, many people don’t prefer to read the inbox very often while they are in the midst of an online meeting. The inbox is mainly used for delivering important notices or messages in an ongoing meeting and with people using it for talking, it gets spammed and important notices gets lost.

* + - * **Solution Proposed**

Keeping in mind the fact that video conferencing apps like Google Meet or Zoom has almost become a necessity at every school, college and workplace, we have tried to use the idea of Object Detection using Tensorflow to help create a middleware that can be integrated into various such apps to facilitate the real time translation of signs, symbols and hand gestures. This will help people on the other end understand and communicate with the deaf and dumb with much more ease. Our technology can serve as a translator, converting the hand signs and showing the exact message the impaired person is trying to convey, thus saving time and screen space.

**1.3 OUR MISSION**

Hand gesture recognition is of great importance for human computer interaction because of its extensive applications in virtual reality and sign language recognition etc. Hand gestures can help you point to people and things in your surroundings. Hand gestures can help you add emphasis and structure when you talk. Hand gestures give clues about your emotional state .The communication through Sign language is a successful way of communication for speech and hearing impaired humans. A Hand gesture recognition system will provide an aid to deaf and mute.

* + - * **OUR OBJECTIVE**
* We have designed an end-to-end deep learning-based Tensorflow object detection model to jointly detect and recognize static hand gesture.
* The choice of a 320x320 MobileNet was made to reduce the computational cost during the inference process, since it has been designed to work faster, for use, for instance, in embedded systems and resource constrained devices.
* The proposed system was evaluated on various standard hand datasets with varying degrees of complexity in terms of the clutter environment. The approach was tested on various gestural datasets with different vocabulary sizes.
* Finally, we are going to design our system to be adaptive, making it flexible to be used potentially.

**IMPLEMENTATION DETAILS**

**2.1 INTRODUCTION**

The very first challenge that we encountered is the selection of a Tensorflow model that’d allow us to give our vision some practical presence. Tensorflow is an extensive library with vast number of modules present in its model zoo. Our vision is to make an application which can serve as a middleware in real time image recognition. So, our first priority is that it should be fast in its predictions. There were a lot of bulky models present which had a tremendous high accuracy, but our first preference is the performance of our middleware, since users will be using them in real-time meetings and video conferencing. Our main objective is to reduce the time of the users by cutting out the translators in between. But, if the ping is high, then this whole concept gets nullified. Also, most of these video conferencing is carried out via our Smartphone and embedded systems which will succumb to heavyweight applications. So, another goal is to make the middleware as lightweight as possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Speed (ms) | COCO mAP | Output |
| SSD MobileNet V1 FPN 640 x 640 | 48 | 29.1 | Boxes |
| SSD MobileNet V2 FPNLite 320 x 320 | 22 | 22.2 | Boxes |
| SSD MobileNet V2 FPNLite 640 x 640 | 39 | 28.2 | Boxes |

Fig2. TF Model based on MobileNet Architecture

We decided to go with **SSD MobileNet V2 FPNLite 320x320.**

* **WHY SSD**

**Single Shot Detector (SSD)**

Before we jump into the reason of why did we choose SSD, let us first look into what SSD actually is. SSD has two components: a **backbone** model and **SSD head**. Backbone model usually is a pre-trained image classification network as a feature extractor.

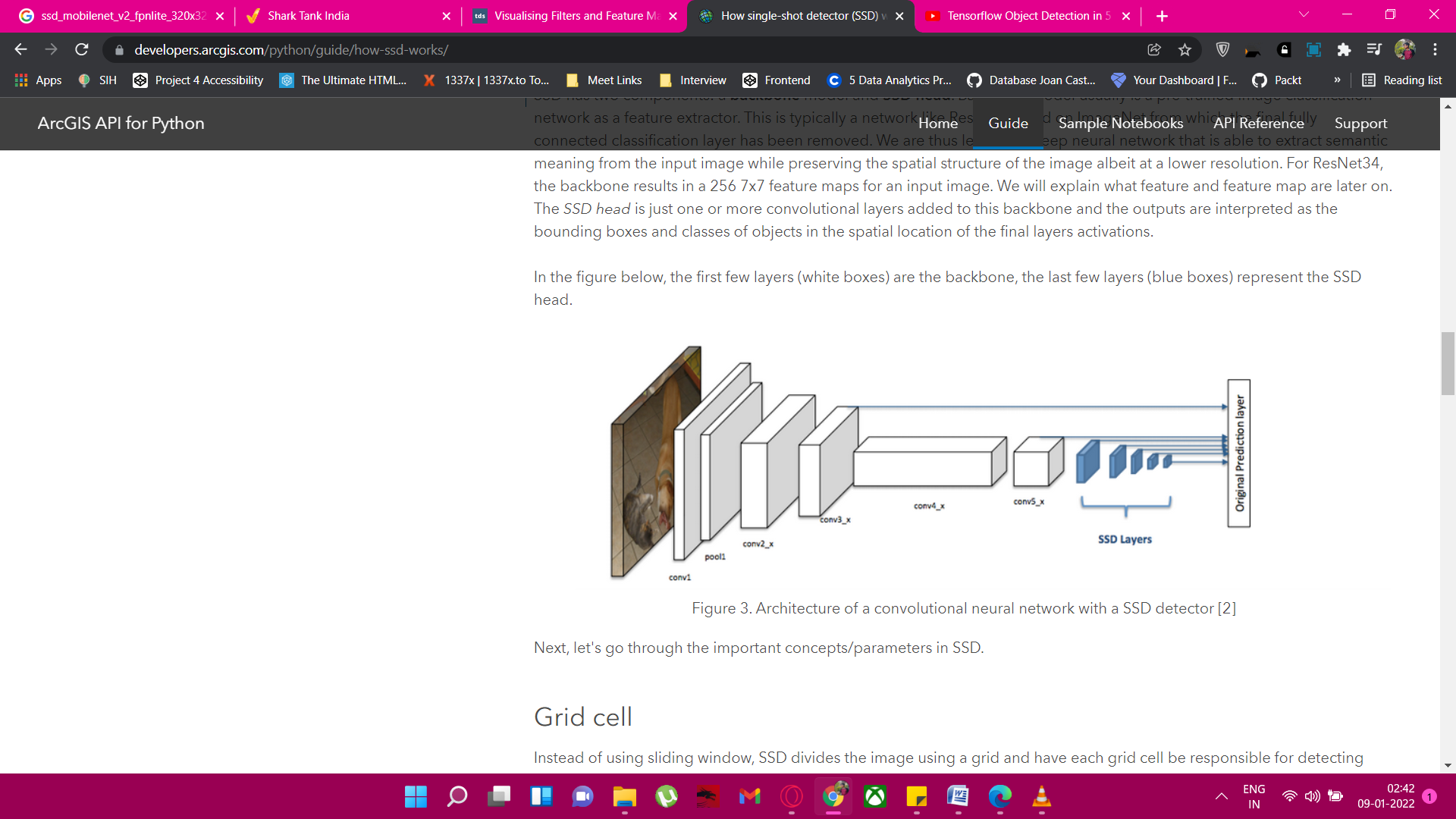


Figure 3. Architecture of a Convolutional neural network with a SSD detector

This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that is able to extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lower resolution. The SSD head is just one or more Convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers activations.

**Advantage of SSD**

SD is designed for object detection in real-time. Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second. Far below what real-time processing needs. SSD speeds up the process by eliminating the need for the region proposal network. To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN’s accuracy using lower resolution images, which further pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN. (Accuracy is measured as the mean average precision mAP: the precision of the predictions.). Hence, we will prefer SSD over other algorithm for our project.

* **WHY MOBILENET**

To achieve best results, ideally, we want to find the smallest possible neural net that can most accurately represent the thing we want it to learn. This is an open problem in machine learning, and until there is a good theory of how to do this; we’re going to have to start with a larger network and lobotomize it. When we compress a neural network, the trade off is network size vs. accuracy. In general, the smaller the network, the faster it runs and the worse its predictions are. But, MobileNet is 32 times smaller and 10 times faster than VGG16 (which is the alternative option to use with SSD as an image classifier model) and produces the same results.

Our aim was to make our application adaptive, considering the fact that we want it to be integrated with video conferencing software; it will be extensively used with not just computers having high GPU but also with mobile phones and other embedded devices. As it is an image classification model based on a streamlined architecture that uses depthwise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices, this was our best option.

* **WHY FPNLite**

Detecting objects in different scales is challenging in particular for small objects. We can use a pyramid of the same image at different scale to detect objects (the left diagram below).However, processing multiple scale images is time consuming and the memory demand is too high to be trained end-to-end simultaneously. Hence, we may only use it in inference to push accuracy as high as possible, in particular for competitions, when speed is not a concern.

Alternatively, we create a pyramid of feature and use them for object detection (the right diagram). However, feature maps closer to the image layer composed of low-level structures that are not effective for accurate object detection.

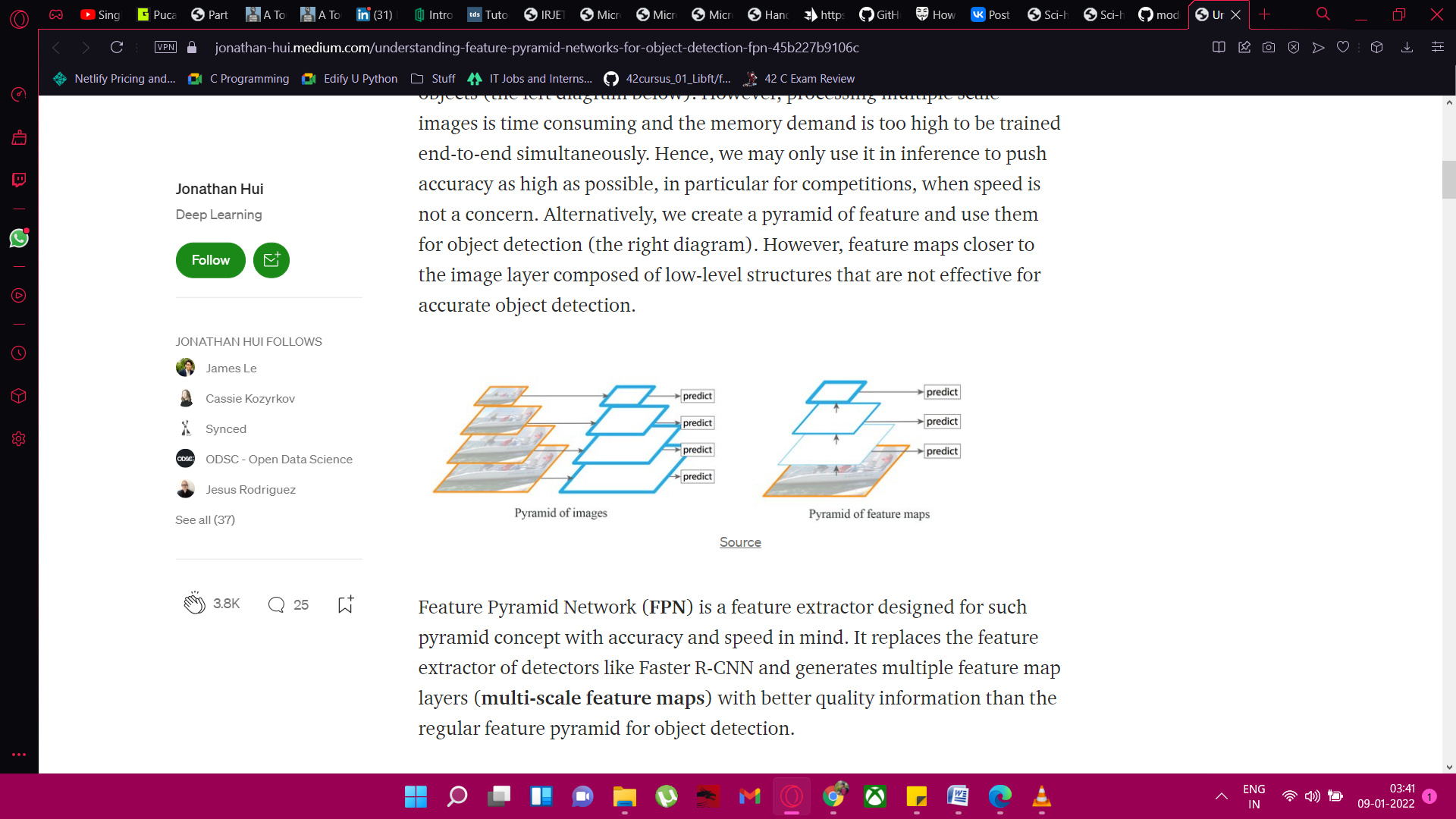


Figure 4. Pyramid architecture of FPN

Feature Pyramid Network (FPN) is a feature extractor designed for such pyramid concept with accuracy and speed, both in mind. It replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection.

FPN composes of a bottom-up and a top-down pathway. The bottom-up pathway is the usual Convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the semantic value for each layer increases. SSD makes detection from multiple feature maps. However, the bottom layers are not selected for object detection. They are in high resolution but the semantic value is not high enough to justify its use as the speed slow-down is significant. So SSD only uses upper layers for detection and therefore performs much worse for small objects. FPN provides a top-down pathway to construct higher resolution layers from a semantic rich layer.

**2.2 ALGORITHM**

The algorithm makes many predictions for every single class 8732 predictions for every single object using 6 convoluted layers, these layers will perform the classification object detection task. For every object SSD will predict 8732 bounding boxes. As we have a lot of prediction per object we have to use non max suppression, to remove duplicate predictions. Our algorithm will check the confidence score of each box and will pick top 200 predictions per object. The overlap should be greater than 50%, and chooses it will pick top 200 boxes and then it will perform no max suppression on it and that will give us the result. Next, let's go through the important concepts/parameters in SSD.

* **Grid Cell**

SSD divides the image using a grid and have each grid cell be responsible for detecting objects in that region of the image. Detection objects simply means predicting the class and location of an object within that region. If no object is present, we consider it as the background class and the location is ignored. For instance, we could use a 4x4 grid in the example below. Each grid cell is able to output the position and shape of the object it contains.

Now you might be wondering what if there are multiple objects in one grid cell or we need to detect multiple objects of different shapes. There is where anchor box and receptive field come into play.

* **Anchor Box**

Each grid cell in SSD can be assigned with multiple anchor/prior boxes. These anchor boxes are pre-defined and each one is responsible for a size and shape within a grid cell. For example, the hand in the image below corresponds to the taller anchor box while the building corresponds to the wider box.

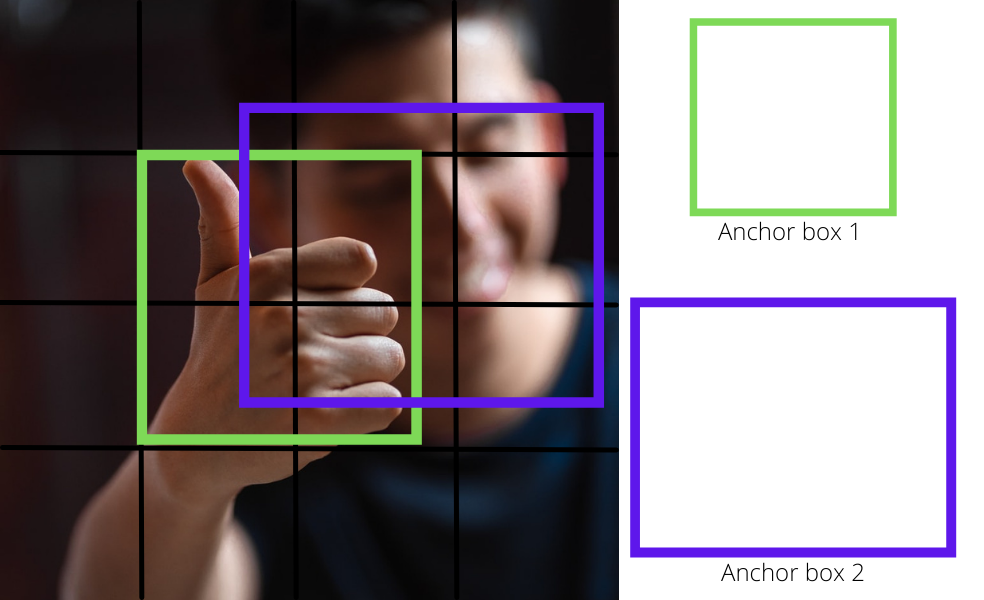


Fig4. Overlapping of Anchor Boxes

SSD uses a matching phase while training, to match the appropriate anchor box with the bounding boxes of each ground truth object within an image. Essentially, the anchor box with the highest degree of overlap with an object is responsible for predicting that object’s class and its location. This property is used for training the network and for predicting the detected objects and their locations once the network has been trained. In practice, each anchor box is specified by an aspect ratio and a zoom level.

* **Aspect Ratio**

Not all objects are square in shape. Some are longer and some are wider, by varying degrees. The SSD architecture allows pre-defined aspect ratios of the anchor boxes to account for this. The ratios parameter can be used to specify the different aspect ratios of the anchor boxes associates with each grid cell at each zoom/scale level.

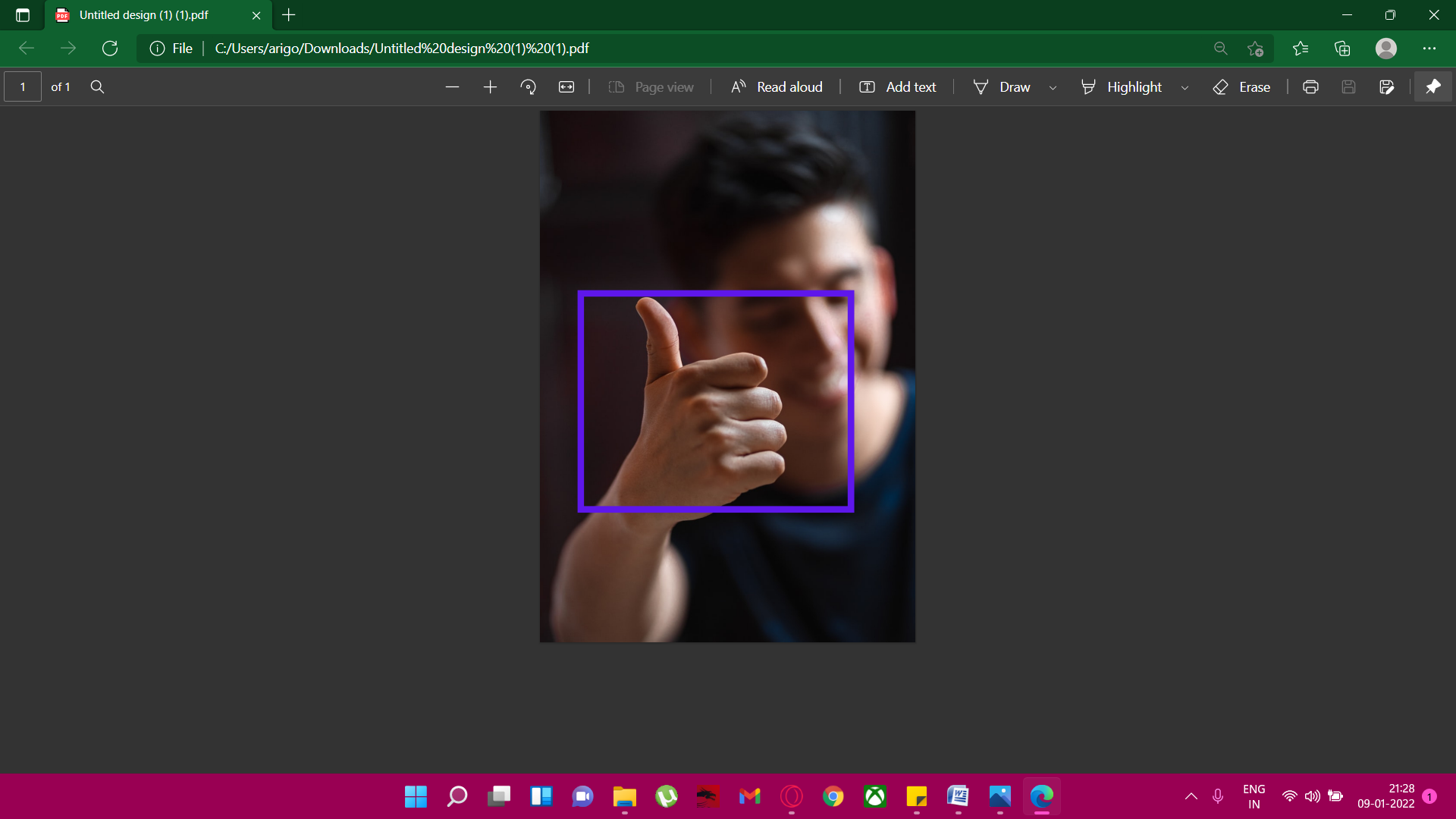
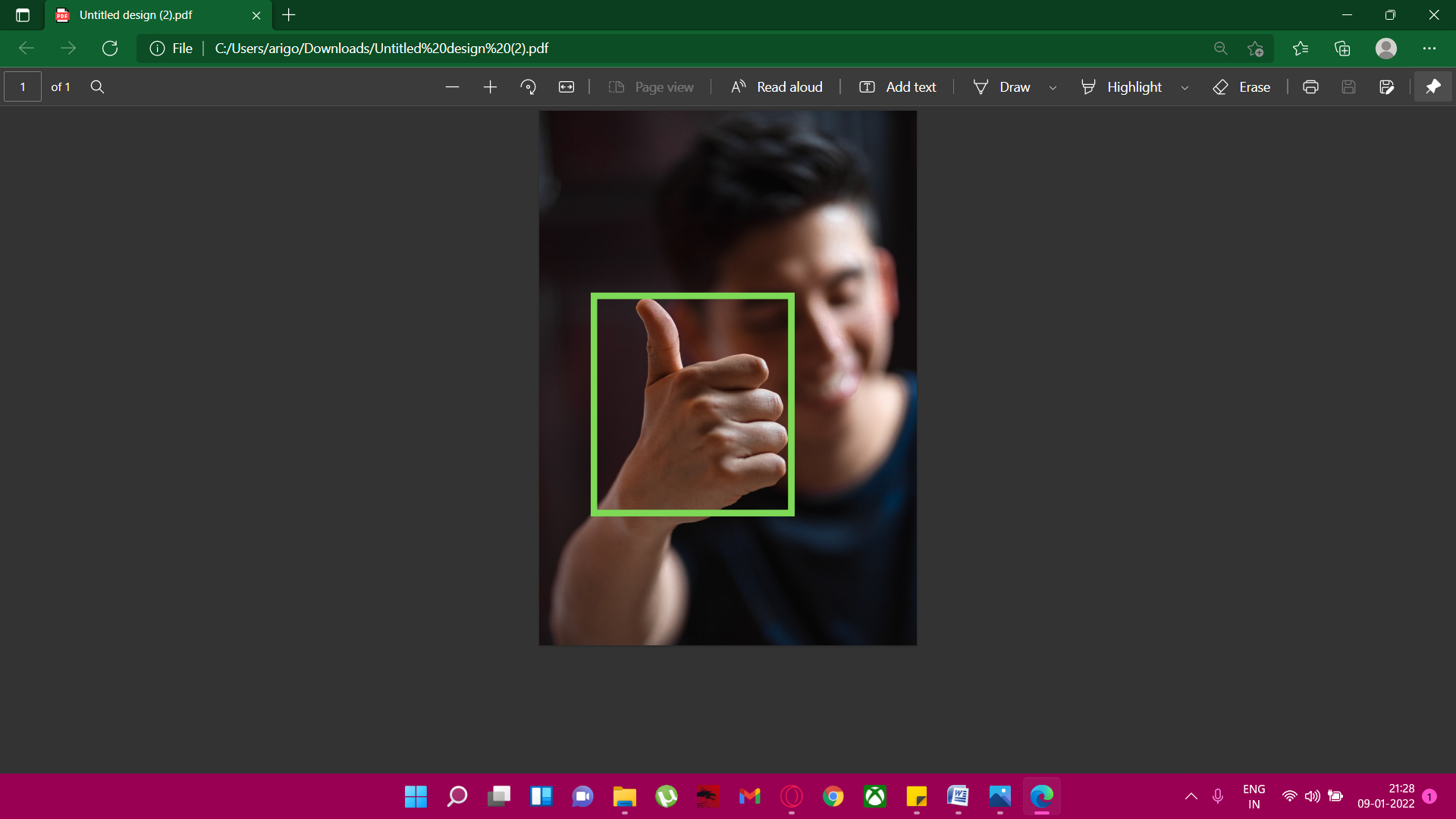


Fig5. Horizontal and Vertical Ratio Labeling of the Image

* **Zoom Level**

It is not necessary for the anchor boxes to have the same size as the grid cell. We might be interested in finding smaller or larger objects within a grid cell. The zooms parameter is used to specify how much the anchor boxes need to be scaled up or down with respect to each grid cell. Just like what we have seen in the anchor box example, the size of building is generally larger than swimming pool.

**CODE**

***# Import opencv***

**import** cv2

***# Import uuid***

**import** uuid

***# Import Operating System***

**import** os

***# Import time***

**import** time

labels **=** **[**'thumbsup'**,** 'thumbsdown'**,** 'thankyou'**,** 'livelong'**]**

number\_imgs **=** 5

We start our project by import python libraries into our jupyter notebook. We’ll need opencv for computer vision, i.e. using the camera of our device is being controlled by opencv2 while taking pictures of the training datasets. We are going to using uuid for naming convention. Basically, we are going to name each and every picture with their correct labels and for that purpose we are installing uuid which will serve as a post-name notation to avoid naming conflicts. The os package will help us to create directories instantly from the code itself. This will save a lot of time regarding branching of numerous folders which we would have had to do manually instead. We are going to automate the process of image collection where our model system will automatically capture the pictures and save them directly into the folders by naming them. For that purpose, we are adding time package to incorporate a few small pauses while clicking pictures.

For the time being, we are going to detect four particular hand signs. So, we are storing them in an array. We are going to collect 5 images of each of them.

IMAGES\_PATH **=** os**.**path**.**join**(**'Tensorflow'**,** 'workspace'**,** 'images'**,** 'collectedimages'**)**

**if** **not** os**.**path**.**exists**(**IMAGES\_PATH**):**

**if** os**.**name **==** 'posix'**:**

**!**mkdir **-**p **{**IMAGES\_PATH**}**

**if** os**.**name **==** 'nt'**:**

**!**mkdir **{**IMAGES\_PATH**}**

**for** label **in** labels**:**

path **=** os**.**path**.**join**(**IMAGES\_PATH**,** label**)**

**if** **not** os**.**path**.**exists**(**path**):**

**!**mkdir **{**path**}**

We are using the built in functions in os library to create directories and folder branches where we will store our training data.

**for** label **in** labels**:**

cap **=** cv2**.**VideoCapture**(**0**)** ***#CONNECTS TO OUR WEBCAM OR CAPTURE DEVICE ,***

***DEVICE NO-O***

***#if label == 'thumbsup': #will use this for specific images***

print**(**'Collecting images for {}'**.**format**(**label**))**

time**.**sleep**(**5**)**

**for** imgnum **in** range**(**number\_imgs**):**

print**(**'Collecting image {}'**.**format**(**imgnum**))**

ret**,** frame **=** cap**.**read**()** ***#CAPTURE THE FRAME USING A WEBCAM***

imgname **=** os**.**path**.**join**(**IMAGES\_PATH**,**label**,**label**+**'.'**+**'{}.jpg'**.**

format**(**str**(**uuid**.**uuid1**())))**

cv2**.**imwrite**(**imgname**,** frame**)**

cv2**.**imshow**(**'frame'**,** frame**)**

time**.**sleep**(**5**)**

"""wait key returns 32 bit, ord returns 8 bit ascii

for comparison we are multiplying 0xFF with waitkey which returns 8 bit

so comparison is made feasible"""

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

**break**

cap**.**release**()**

cv2**.**destroyAllWindows**()**

This is the main picture collection code where we have automated the process of picture collection and labelling. For each sign, the loop will iterate 5 times start clicking pictures of the subject sitting in front of the camera. The opencv2 library has built in codes to control the camera. We have added a sleep option of 5 second in between clicking the pictures so that the camera can capture non-pixelated proper images without shakes or breaks. After clicking the pictures, we have used format tool and uuid to name them and place them in their respective folders. We have also added an escape feature if we want to terminate the algorithm in midst.

**!**pip install **--**upgrade pyqt5 lxml

LABELIMG\_PATH **=** os**.**path**.**join**(**'Tensorflow'**,** 'labelimg'**)**

We installed pyqt5 package because we are going to store xml models of our labelled pictures into the dataset which we will use to train our model

**if** **not** os**.**path**.**exists**(**LABELIMG\_PATH**):**

**!**mkdir **{**LABELIMG\_PATH**}**

**!**git clone https**://**github**.**com**/**tzutalin**/**labelImg **{**LABELIMG\_PATH**}**

**if** os**.**name **==**'nt'**:**

**!**cd **{**LABELIMG\_PATH**}** **&&** pyrcc5 **-**o libs**/**resources**.**py resources**.**qrc

**!**cd **{**LABELIMG\_PATH**}** **&&** python labelImg**.**py

We found this great labelling tool on github which we are going to use to label our images. In this particular snippet of code, we are cloning the repository of that labelling tool and we running it. The rest of the work is done on the application which converts our labelling into xml files and stores them into their respective folders with the same name as that of the picture.

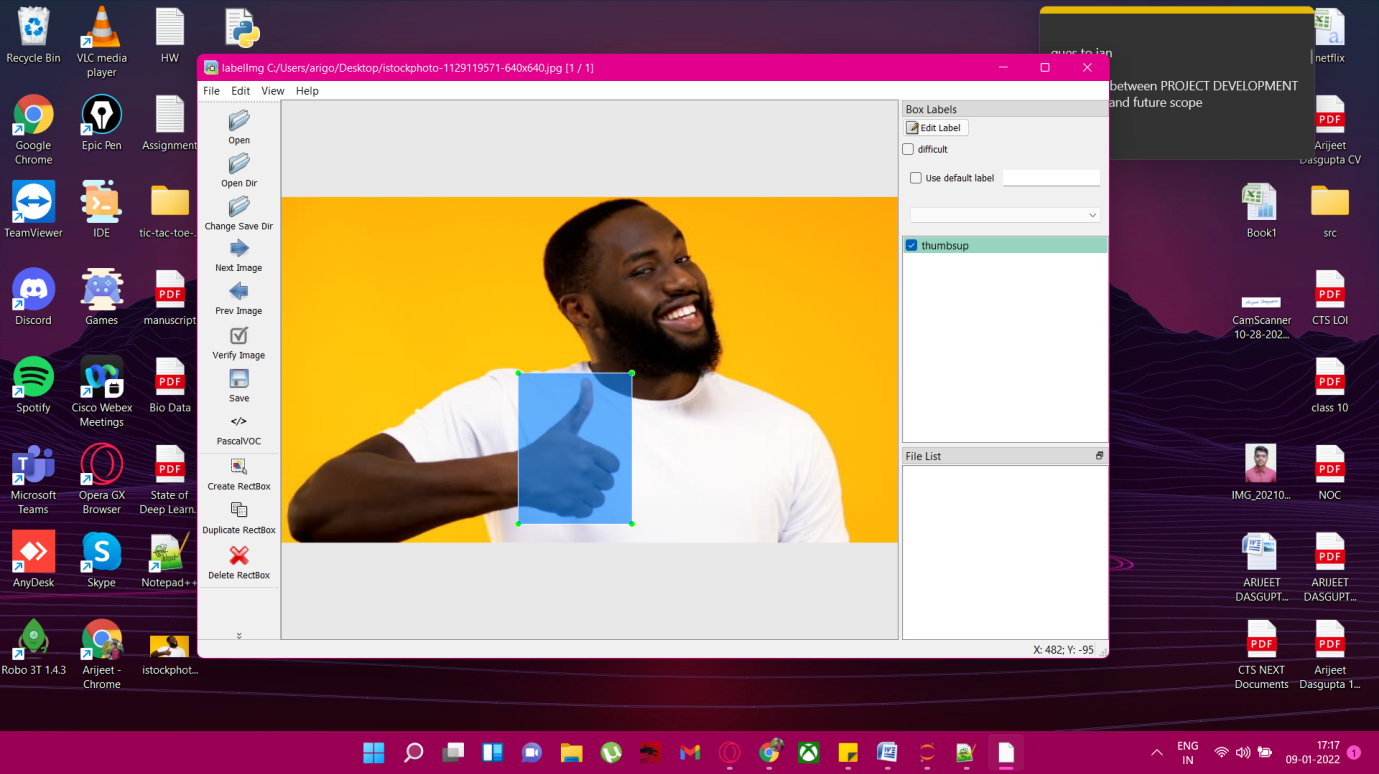


Fig6. Labeling of Images

Below is the xml code for the following labelled image:

<annotation>

<folder>Desktop</folder>

<filename>istockphoto-1129119571-640x640.jpg</filename>

<path>C:\Users\arigo\Desktop\istockphoto-1129119571-640x640.jpg</path>

<source>

<database>Unknown</database>

</source>

<size>

<width>640</width>

<height>360</height>

<depth>3</depth>

</size>

<segmented>0</segmented>

<object>

<name>thumbsup</name>

<pose>Unspecified</pose>

<truncated>0</truncated>

<difficult>0</difficult>

<bndbox>

<xmin>245</xmin>

<ymin>183</ymin>

<xmax>363</xmax>

<ymax>340</ymax>

</bndbox>

</object>

</annotation>

**FUTURE SCOPE**

Till now we have just started with the image collection automation part of our model. We are yet to go a long away. For our next step we are going to train the model on the algorithm we have chosen, i.e. SSD MobileNet FPNLite 320x320. This will complete our model as a middleware which can serve the purpose it intends to do. These are as follows:

* 1. **Training**

First we need to train our ML model in order to learn. All the models that are part of TensorFlow are already trained in some particular order to do a fraction of the work. This makes our job easier, as we have to train the model only to identify the kind of hand sign we are looking for. We don’t have to train it to detect objects whatsoever.

* 1. **Testing**

After training, the pivotal part of our project would be to test the data. We are going to feed it several pictures of subjects and we are going to judge how efficient our model is by calculation the percentage of it correct predictions.

**4.3 Web App and Hosting**

We are going to use React Js as the front end to develop a website where we will place our application for everyone to use. Our ML model will be the part of the backend which will be made using python. We are going to host this website on some free hosting provider such as Github or Netlify so that everyone can have access to this.

* 1. **More Gestures**

Currently, we have only four kind of gestures which our model can detect and predict. We are going to increase the number of hand signs so that it becomes more useful to the people using it and they have actual conversations using our product.

**CONCLUSION**

The project report entitled "Hand Gesture Recognition using TensorFlow" is still under construction. The work on the project is under progress. The part of our system has been developed with much care that it is free of errors and at the same time it is efficient and less time consuming. The important thing is that the system is robust. We have tried our level best to make the model as efficient as possible. Also provision is provided for future developments in the system.

The entire system is planned thoroughly and is yet to be complete. Once completed, this software can be very beneficial for the people who have speaking and listening impairments and can prove to be a boon for them, especially in this pandemic struck scenario, with everything going digital at the speed of light.

We gained a thorough knowledge while doing this project. A lot of research work went behind it. We tried to automate as much work as possible, be it folder making to taking pictures, cloning repositories, downloading packages etc and we plan on to do the same as the project paces forward. We learnt a lot about Deep Learning models and architectures such as SSD, Sliding Window and their pros and cons in order to select the best model for our project. We learnt how the algorithms in computer vision work in the speciality niche of object detection. We went through the Tensorflow model zoo in search of models that can optimize our middleware even more. We learnt a lot about python automation and scripting during this project which we were previously unfamiliar with. Also, we carried most of our project out on Jupyter Notebook which gave us an idea of how notebooks work in comparison to traditional IDEs.

**BIBLIOGRAPHY**

[www.towardsdatascience.com](http://www.towardsdatascience.com)

[www.ieeexplore.ieee.org](http://www.ieeexplore.ieee.org)

[www.machinelearningmastery.com](http://www.machinelearningmastery.com)

[www.machinethink.net](http://www.machinethink.net)

[www.coursedaddy.com](http://www.coursedaddy.com)

[www.wikipeida.org](http://www.wikipeida.org)

[www.medium.com](http://www.medium.com)

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)

[www.quora.com](http://www.quora.com)