LENSEDIT USING ML

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

In

Computer Science and Engineering

By

Rishabh Malhotra (161263)

Under the supervision of

Dr. Pradeep Kumar Gupta

Associate Professor

To

Department of Computer Science & Engineering and Information Technology

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

Candidate's Declaration

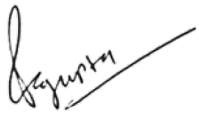
I hereby declare that the work presented in this report entitled **"LensedIt using ML"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2019 to December 2019 under the supervision of **Dr. Pradeep kumar Gupta** (Associate Professor , Computer Science).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Rishabh Malhotra, 161263

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



(Supervisor Signature)

Dr. Pradeep Kumar Gupta

Associate Professor

Computer Science and Engineering/Information Technology

Dated:27/05/20

ACKNOWLEDGMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to Dr…….. for their guidance and constant supervision as well as for necessary information regarding the project and also for their support in completing the project.

We would like to express our gratitude towards our parents and Jaypee University of Information Technology for their kind co-operation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

Machine Learning(ML) is the logical investigation of calculations and measurable models that PC frameworks use to play out a particular assignment without utilizing unequivocal directions, depending on examples and making inferences. AI calculations manufacture a numerical model dependent on test information, known as "preparing information", to anticipate or settle on choices without being unequivocally customized to play out the assignment. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, providing the real-time meaning of a word or the conversion from one language to another. When it comes to us, people, we do not classify anything as natural view or virtual view. Everything is just "normal" for us. We do not work in different ways, for example, to read some news from a newspaper or to read it from the screen of our laptop or mobile. Although the eyes may be affected differently, but for the human mind, it is the same thing. We are not really aware when we interpret the real world. It is like a default setting for humans instead. It is not a test for us to experience and easily understand the specific elements of the visible world. Our subconscious mind performs each process without any problem. Our project aims to use such an application of computer vision to translate and interpret a piece of natural scene text in real time. The project mainly consists of 3 stages, text detection, text recognition, translation of the word.

1. Introduction

1.1 Introduction

With regards to us, the individuals, we don't group anything as a characteristic scene or virtual scene. Everything to us is simply "typical". We don't utilize various techniques, for instance, to peruse some report from a paper or to peruse it from a screen of our PCs or mobiles on some other screen. In spite of the fact that the eyes may get influenced contrastingly however to the human brain, it is something very similar. We are not so much mindful when we decipher this present reality. It is rather a sort of a default setting for people. Encountering unmistakable components of the noticeable world and seeing easily is a no test to us. Our subconscious mind does every single procedure with no issue.

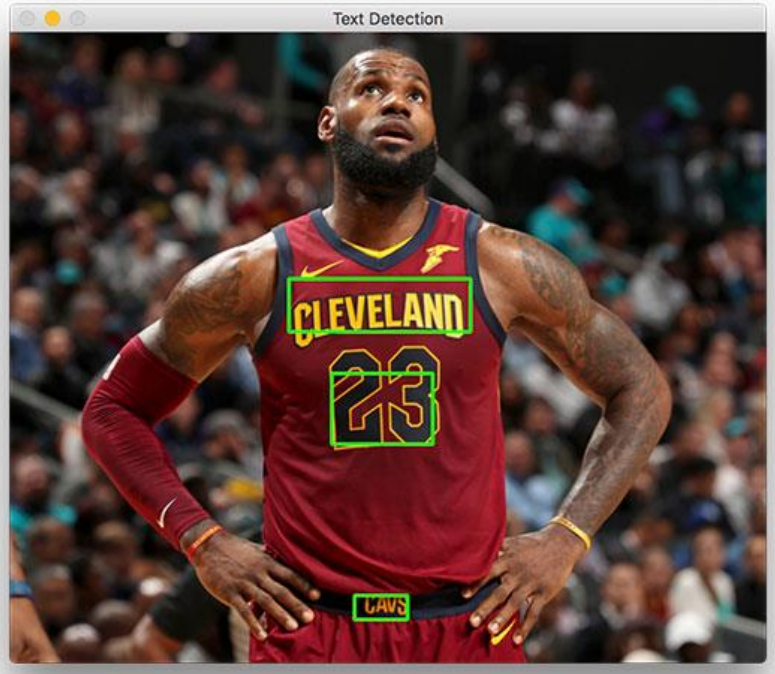


Figure 1.1: Text detection example

Contrary to our minds, a machine considers visuals to be an assortment of numerical amounts and scans for designs in pictures, a video, a practical scene, or even live scene (real-time), to see and recognize key features of the sort of information. The manner by which a PC or a framework deciphers an image is not exactly equivalent to people. PC vision uses picture taking care of figuring to investigate and comprehend visuals from a single picture or a plan of pictures. An instance of PC vision is to distinguish text and afterward remember it from recordings and photographs. Along these lines, perusing a piece von text for machines is substantially more confused. Despite the fact that it is confounded for the human brain too with little memory units like neural connection and everything except for it appears to be so evident for people. While its execution we fairly get the significance of how people think.

Our undertaking targets utilizing such use of computer vision to discover interpretations and implications of a bit of characteristic scene message continuously. The project contains three stages, text detection, text recognition and providing translation with the detected test.

Figure 1.2: Meaning of A Word in Real Time

Text, as one of the most compelling innovations of humankind, has assumed a significant role in human life. The rich and exact data exemplified in text is valuable in a wide scope of vision-based applications; along these lines, text location and acknowledgment in characteristic scenes have become significant and dynamic research themes in PC vision and archive examination. Particularly as of late, the network has seen a flood of research endeavors and generous advances in these fields, however an assortment of difficulties (for example clamor, obscure, mutilation, impediment variety) despite everything remain. The motivations behind this study are three-overlap: 1) acquaint up-with date works, 2) recognize best in class calculations, and 3) foresee potential research bearings later on. In addition, this paper gives extensive connects to freely accessible assets, including benchmark datasets, source codes, and online demos. In synopsis, this writing survey can fill in as a decent reference for scientists in the regions of scene text identification and acknowledgment.

Initially, the researchers in order to detect and recognize text hard coded features. This means that the researches actually fed the features itself to the model along with the input so as to do such computer vision tasks. It was only after the boom of deep learning that such tasks became mainstream. These deep learning driven models, with small adjustments to weights and biases learned the features themselves and even gave better prediction than the older models. In 2014, EAST, Efficient and Accurate Scene Text detection, algorithm came out of Facebook. The tech mammoth managed to create a deep learning model that detected scene text, i.e. the natural text where we see on things like billboards. This was a huge leap as it eliminated the need of various stages in the task of text detection and gave out predictions in one shot. Similar to this, CRNN model came out in the field of text recognition. This too, just like EAST, made predictions in one shot and provided with a output with a one-stage model. An image is taken as input and encoded into a vector then fed to a bi-directional recurrent neural network, which gives the output of recognized text. This recognized text can be then fed into a recurrent neural network. For the purpose of translation, encoder-decoder model is used which is state-of-art technique for the same.

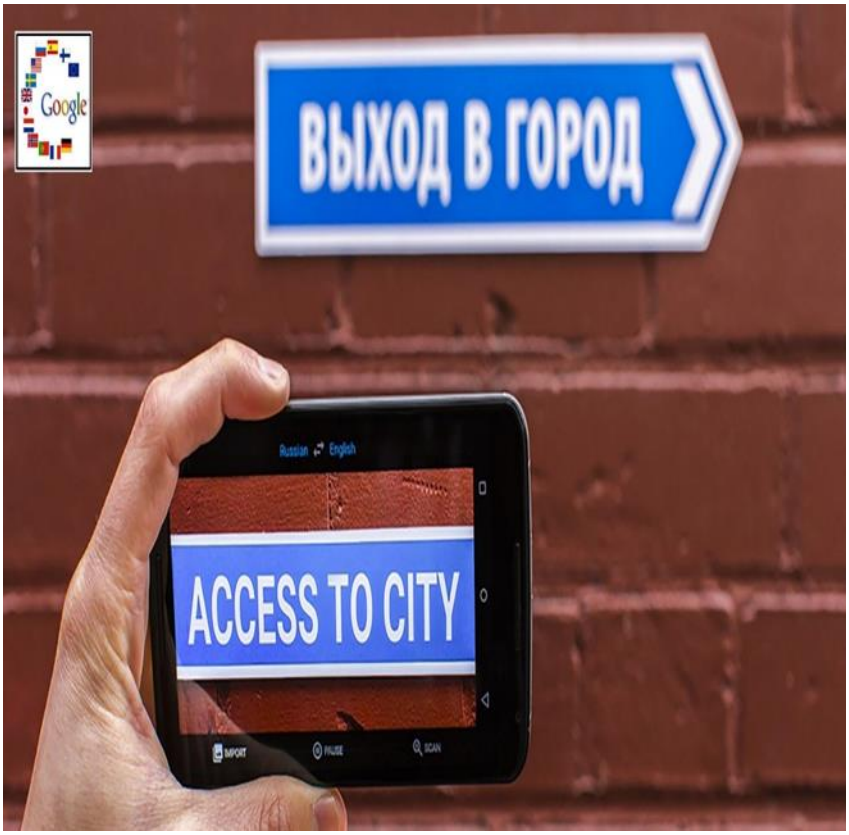


Figure 1.3: Conversion of a word from one language to another

1.2 Problem Statement

To provide with real-time translation of a scene-text or text “in the wild” using deep learning, integrated with a camera, in real time. For example, you are backpacking to mount Tibidabo, near Barcelona and you are somewhat lost. However, there have been some signboards in scattered places; they have all been in Spanish. A translator would come in handy in such a situation. A translator that can provide you with the translation in the language you are comfortable with and provide them to you in real-time.

Therefore, the problem at hand is to provide real-time solution to the user for obtaining translations. Real-time means that instead of clicking the photos and then processing the text from the photo, the solution can be provided to you then and there.

1.3 Objective

The aim of our project is to provide real-time solutions to the problems like finding meaning of some word or when you want to understand something in your own language, translator will come handy instead of going online and searching for the same. Different models are used for text detection and recognition, EAST and CRNN respectively. Text detection is mainly used for providing the translations on-screen as it provides with coordinates. The detected part is then fed to CRNN network, which gives out the recognized text. This text moves to next step in processing, i.e. translation part for which encoder-decoder model is employed. The encoder-decoder model is trained to convert English to Hindi.

1.4 Methodology

Firstly, we try to run the algorithm on images. The network that we design is first tested on images and the in next phase it will be moved to video.

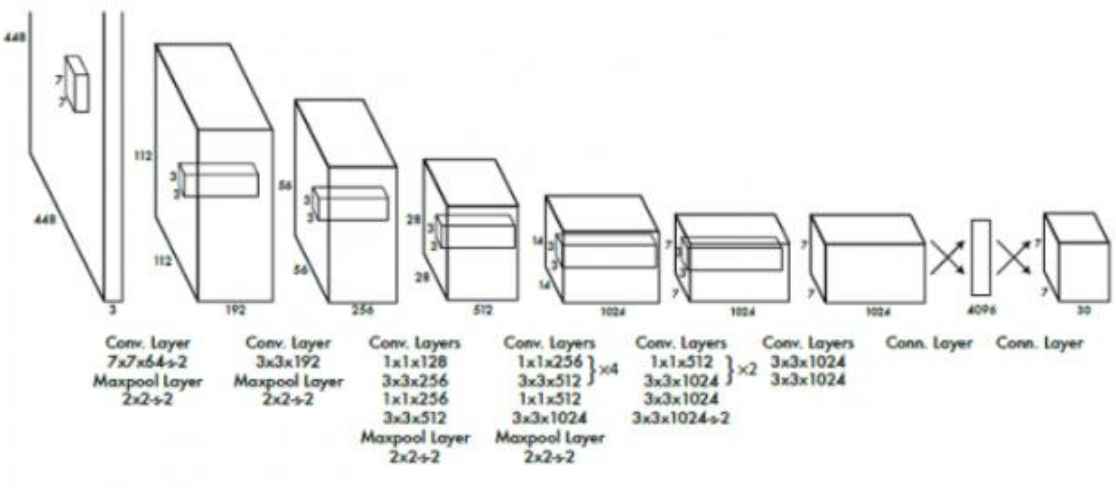


Figure 1.4:Working Of YOLO Algorithm

What we do is that we firstly run the text detection algorithm EAST(Efficient and Accurate Scene Text detector) to detect the text in a natural scene. The EAST algorithm is able to produce somewhat flexible bounding boxes. This means that they will not always be rectangular boxes parallel to the horizontal axes but some flexibility will be there. For

example, a text maybe curved like branding on a bottle of mineral water, which will be detected by EAST but not by YOLO. We are comparing EAST and YOLO because both can be used detect text in real-time though YOLO is an object detection algorithm. EAST has given remarkable results. It can detect a text in a video of 13fps at 720p resolution.

The next stpe was to apply the same algorithm on a video or the live video recording frpm the camera. The frame rate dropped to between 2.5 FPS to 2.8 FPS. Pre-trained model of EAST was downloaded from the internet and executed using the OpenCV library of cv2.

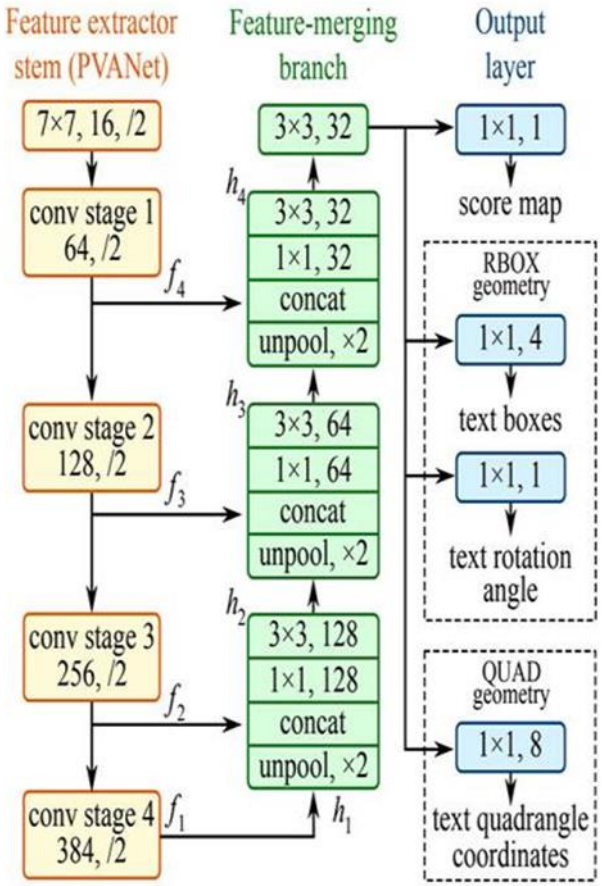
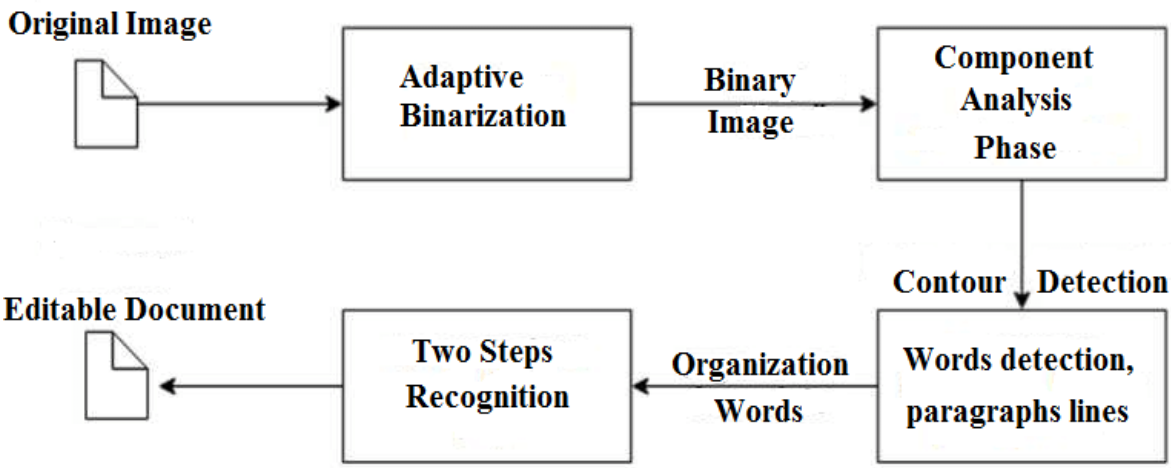


Figure 1.5: Working Of EAST Algorithm

The first layer of the EAST detector is taken from PVANet that is used as a Feature extractor net. There are 4 stages in this stage and out of each stage the features are taken out. These extracted features are then passed on to the Feature-merging branch. Going along the arrows of the branch we see that the features are unpooled and the other set of the

features are concatenated of we can say stacked on top of each other. Out of the last stage of the penultimate layer, we are able to obtain four different outputs as the Output layer namely, score map, RBOX geometry which provides us with the text boxes, text rotation angles and QUAD geometry which gives us text quadrangle coordinates.

The next step was to recognize the detected text. For the same purpose, Tesseract, developed by Hewlett Packards initially, but then got taken over by Google, was used initially. The detected part was transferred to the recognizer. Tesseract gave below par predictions of the recognized text where most of it was mostly gibberish.



1.6: Tesseract-ocr architecture

Since unfavorable predictions were obtained from tesseract in real-time scenario, we moved on to CRNN, Convolution Recurrent Neural Network, architecture. This architecture is trained as a whole and is deep learning inspired model. The input image is converted into a vector as the result of a CNN (Convolutional Neural Network) which is then fed into a RNN (Recurrent Neural Network) which gives the output as the recognized text. CRNN is a widely used recognizer. The authors of TextBox++, a natural scene text detection technique recommend using CRNN along with their model to detect and recognize the text. Given below is the architecture of the Convolutional Recurrent Neural Network.

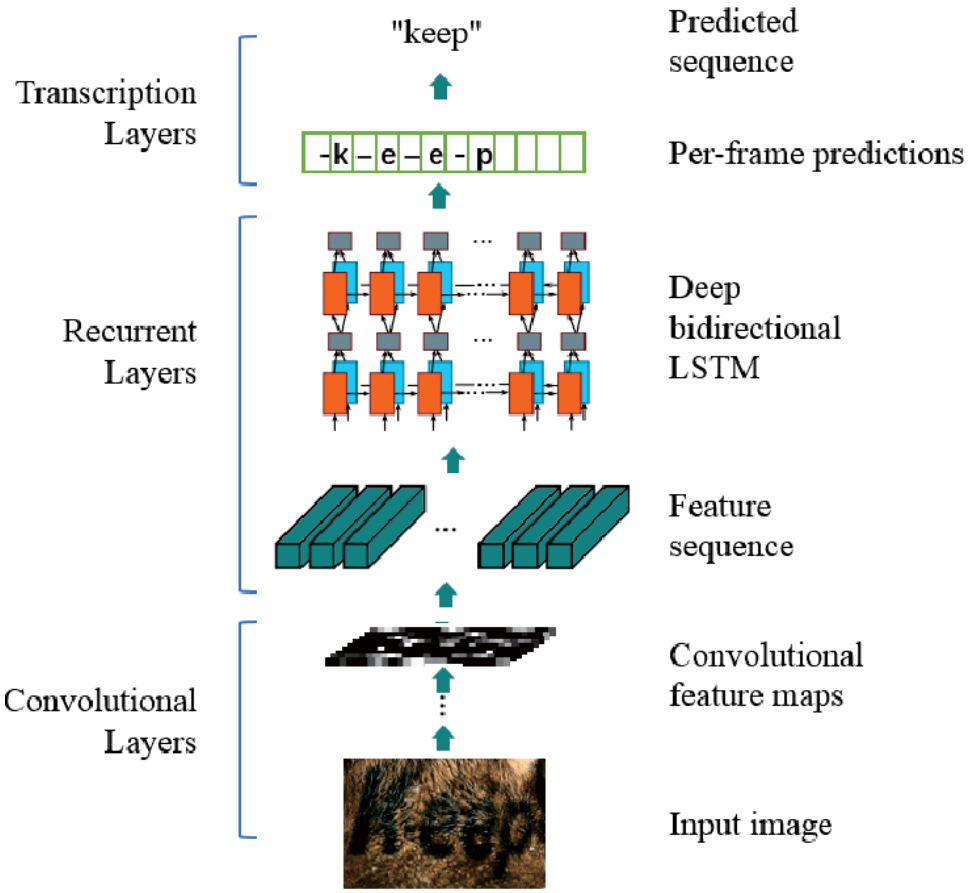


Fig 1.7: Architecture of CRNN

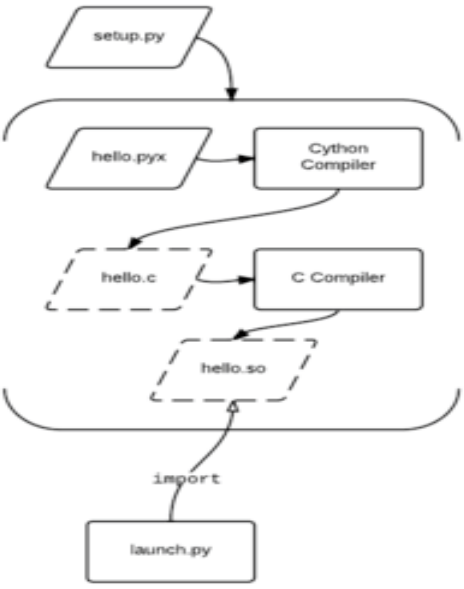
Image is fed into the model which converted into a bunch of feature maps by the Convolutional Layer. These feature maps are then converted to a bunch of feature sequences which are nothing but one-by-one parts of the convolutional feature maps. These feature sequences are then fed into the bidirectional LSTM. The ensemble of a bidirectional Long-Short Term Memory layer is such that one LSTM layer runs forward while the other one runs backwards thus enabling the layer to learn the context of the text for a longer period of time. After some post-processing the output of the network is obtained as the recognized word.

A pre-trained model was used by us and integrated into our project. Thus the text regions detected are then fed to this RNN model to recognize the text.

Integrating the recognizer part of the project to the already established real-time system brought down the rate of video recording, that is, frames per second(FPS) to 0.1-0.2FPS. Thus, after some research, we found out that converting the code to Cython might be of help.

Cython is a superset of the Python programming language. Although most of the part is written in Python, it is supposed to give the executeion and runtime comparable to C and C++

Cpython modules are assembled using cython. Clarified python-like code is amassed to C or C++ then thusly encompassed by interface code, conveying development modules that can be stacked and used by typical Python code using the import announcement, anyway with essentially less computational overhead at run time. Cython similarly energizes wrapping free C or C++ code into python-importable modules.



.Fig1.8: Compiling process in Cython

Classes and functions were converted into Cython using above compiling process. This resulted in a small increase in the frame rate. The frame moved from 0.2 frames per second to 0.5-0.6 frames per second.

Since the speedup we got was not sufficient to support our project, we tried to use threading. In software engineering, a thread of execution is the littlest grouping of customized directions that can be overseen autonomously by a scheduler, which is ordinarily a piece of the working framework. The execution of threads and procedures contrasts between working frameworks, however much of the time a thread is a part of a procedure.

Threads are nothing but small functional units that are used for concurrent processing. The processing done by threads is not parallel but concurrent although it appears to be parallel. What actually happesns is a threade changes its context. This meansa that inspite of running two trades parallely they are run one by one by changing the contexts one after the other. This is what actually happens and it is different than in multiprocessing.

Threadeing in generally used we are to work with I/O devices like cameras, and monitor screens etcetera. It is recommended to use wit hI/O devices in order to gain some seedes ups. But if you are already using hundered percent of your Central processing Unite, tou cannot do anything about it but use multiprocessing.

Pytho, the main language of the project has a class Threading which allows you to handle thread manually or automatically. That meansa you can also let computer handle you threads. The best part about python is that it provides similar interface for both, the multiprocessinge and the threadinge.

Threades actually share resources unlike multiprocessing where every process has to have its own copy. This means that under one processor, many threads can run simultaneously. This can also result in race condition like situatiuons. Therefore appropiate locks can be used to handle it. It mainly works with the intructiouns pointerese also called program counter .

Using threading was not much of use as the frame reate went up from 0.5 frames per second to 0.8-0.85 frames per second.

Therefore, we decided to use multiprocessing.

Multiprocessing is using many cores of CPu, the brain of the computer, the central processing unit which is responsible for all of you activities and task that take place on your machine. Many prcesses are created in this technique and they are run parallely. This means that at the same time two processes are running unlike Threadeing in which two threads are run one-by-one actualy inspite being run together as everyone expects to run them.

Each process has its own copy of the data thus no race codition. Race condition can take place only in the case of shred variables where the lockas ans the semaphores come into the picture. Earlier busy waiting was employued to ensure that race condition does not take place but since that took a lot of CPU's processing powers and dumped them to waste, the technique of busy waiting is frowned upon these days. Therefore semaphores and locks prevail.

Semaphores too are of two types, binary and counting. Counting semaphores can enbale multiple processes to access the shared data at the same time which will violate the conditions but still they are used in some cases where accessing them atleast two at a time does not matter. Binary ssemaphoreses on the other hande can assume only two values either zero or one thus ensuring no race conditions. There are tqwo types of instructions that can increase or decrease that vakues of semaphores and mind you they are only decreases by one at a time. Up() and Down() methods are used to handle them.

These methods can alwasys result in a dead;ock. Now, these deadlockes can be handles in multiple wayes like deadlock detection, deadlock prevention and deadlock indentificatione. Deadlock, by the waay, means that all the processes are stuck and cannot move forwad, lol. Therefore, often the processes have to be interrupted and killed. Different operaing systems desl with deadlocks in varied ways. Windows uses some methods to do that but ubuntu usually ignores because they do not usually occure or ioccur as frwuently therefore they ignore it.

In Flynn's scientific categorization, multiprocessors as characterized above are MIMD machines. As the expression "multiprocessor" regularly alludes to firmly coupled frameworks in which all processors share memory, multiprocessors are not the whole class

of MIMD machines, which additionally contains message passing multicomputer systems. In a multiprocessing framework, all CPUs might be equivalent, or some might be saved for unique purposes. A mix of equipment and working framework programming structure contemplations decide the evenness (or deficiency in that department) in a given framework. Obviously so, dealing with deadlock is really easy. Some methods to handle them are bankers algo which tries to predict whether some allocations of resources will ocucur in deadlock or not. Allocation graphs can be used to see the minimum amount of resources required so that no deadlock will occure. Such allocation can be done like one process's requirements can be fulfilled by allocating number of require resiurces-1 to other processes. This will result in completion of one resource which will release all the resources being held by the process and these resources will be used by the other processes so that they can complete their executione.

There are some frameworks present which treat all the central processing units on the same scale. These frameworks are called symmetric multiprocessing(SMP). While there are other frameworks which do not treat them equally or in other words treat the asymmetrically. There are twor such types of these, NUMA and ASMP.

Ace/slave multiprocessor framework

In an ace/slave multiprocessor framework, the ace CPU is in charge of the PC and the slave CPU(s) performs allocated errands. The CPUs can be totally unique regarding pace and engineering. A few (or the entirety) of the CPUs can have share regular transport, each can likewise have a private transport (for private assets), or they might be confined with the exception of a typical interchanges pathway. In like manner, the CPUs can share basic RAM as well as have private RAM that the different processor(s) can't get to. The jobs of ace and slave can change starting with one CPU then onto the next.

An early case of an ace/slave multiprocessor framework is the Tandy/Radio Shack TRS-80 Model 16 personal computer which turned out in February 1982 and ran the multi-client/performing various tasks Xenix working framework, Microsoft's rendition of UNIX (called TRS-XENIX). The Model 16 has 3 microchips, a 8-piece Zilog Z80 CPU running at 4MHz, a 16-piece Motorola 68000 CPU running at 6MHz and an Intel 8021 in the

console. At the point when the framework was booted, the Z-80 was the ace and the Xenix boot process instated the slave 68000, and afterward moved control to the 68000, whereupon the CPUs changed jobs and the Z-80 turned into a slave processor that was answerable for all I/O tasks including plate, interchanges, printer and system, just as the console and incorporated screen, while the working framework and applications ran on the 68000 CPU. The Z-80 could be utilized to do different undertakings.

The prior TRS-80 Model II, which was discharged in 1979, could likewise be viewed as a multiprocessor framework as it had both a Z-80 CPU and an Intel 8021microchip in the console. The 8021 made the Model II the primary work station framework with a different separable lightweight console associated with by a solitary slight adaptable wire,

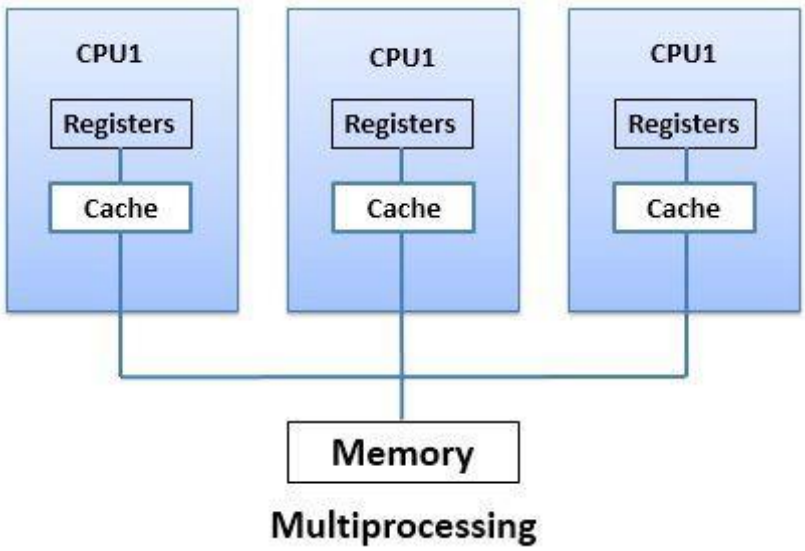


Fig1.9: Multiprocessing System

and likely the main console to utilize a devoted microchip, the two properties that would later be duplicated years after the fact by Apple and IBM.

Guidance and information streams

In multiprocessing, the processors can be utilized to execute a solitary grouping of directions in various settings (single-guidance, different information or SIMD, frequently utilized in vector preparing), numerous arrangements of guidelines in a solitary setting (various guidance, single-information or MISD, utilized for excess in safeguard

frameworks and at times applied to portray pipelined processors or hyper-stringing), or numerous successions of guidelines in different settings (various guidance, numerous information or MIMD).

Firmly coupled multiprocessor framework

These frameworks contain various central processing units that are relted to each other at the transport level. Such an arrangement may force the central processing units to acquire a shared memory or, in other case, they may take up both the nreaby and the shared memory. An example for SMP framework, a pretty good one actually, would be the p690 Regetta by IBM. For business PCs, Intel Xeon was at the the top of its game being a x86 alternative untile the AMD's Opteron processors arrived. These processors hand their own reserves along with the shared memory. The Intel Xeon accessed this shared memory using a typical funneling whear AMD's Opteron used the means of free pathways to the framework Random Access Memory.

Chip multiprocessers have more than one processors on one single chip. Like most of the momdern chip sets where thay have multiple cores in one chipset it can also be termed as miltiple cpus in one chipset.

Approximately coupled multiprocessor framework

Approximately coupled multiprocessor frameworks (regularly alluded to as bunches) depend on different independent single or double processor ware PCs interconnected by means of a rapid correspondence framework (Gigabit Ethernet is normal). A Linux Beowulf group is a case of an inexactly coupled framework.

Firmly coupled frameworks perform preferable and are truly littler over inexactly coupled frameworks, yet have truly required more noteworthy beginning ventures and may devalue quickly; hubs in an approximately coupled framework are generally economical item PCs and can be reused as free machines upon retirement from the bunch.

Force utilization is additionally a thought. Firmly coupled frameworks will in general be substantially more vitality productive than bunches. This is on the grounds that significant economy can be acknowledged by structuring parts to cooperate from the earliest starting

point in firmly coupled frameworks, while approximately coupled frameworks use segments that were not really planned explicitly for use in such frameworks.

Inexactly coupled frameworks can run diverse working frameworks or OS forms on various frameworks.

Using multiprocessing gave us a huge speedup and took our frame rate to 124 frames per second. But, even at this rate, the sytem was hanging. This was because the model was not able to detect images at the rate they were coming in. What we did was while one core of the processor continued to read frames, the other processor core took a frame and did processing on it. Therefore, the processing of thr frame, the detection and recognition parts were kind of out sources to another core. This helped enormously in moving forward with the project.

Now comes the next stage, the stage of translation. For the purpose of translation, Recurrent Neural Networks or as machine learning developers say, RNNS are used. These Recurrent Neural Networks are of three types, the vanilla recurrent neural network, the GRU cel or Gated Recurrent Unit and the LSTM cell or the Long Short Term Memory cell.

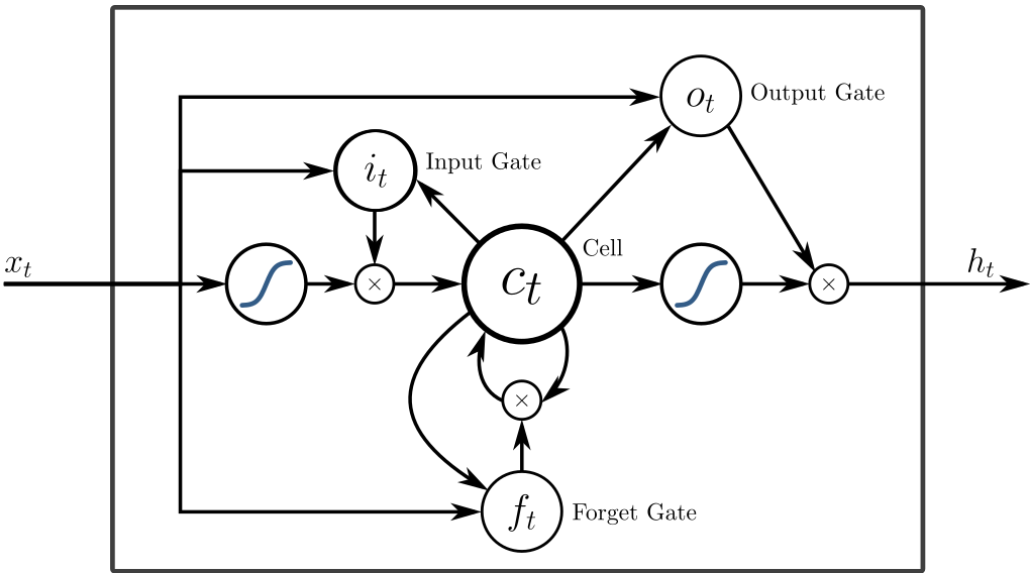


Fig 2.0: LSTM Cell

The vanilla RNN is not able to keep the hold of the contect of the input text for the longer durations just like GRU cells. Therefore, for the purposes of translation LSTM cells are used.

LSTM or the full for, Long Short Term Memory and RNN cells or the smallers parts of a Recurrent Neural Network. These are used where we need to learn from the text. These provide better ouputs than teir counterparts because they are able to learn the context of the text for the longer peroiods of time. Taking an example, Rishabh is a boy. He is also a student. He studies hard and maintains is pointer. But is yet unable to bring the plagiarism levels of his report down to twenty percent which raises a lot of question. Now, a human mind is aware that about what person we are talking up untill the end of the sentence but it is not necessary that a computer will learn that too. GRUs and vanilla RNNs fails to do so so we use LSTM networks.

The CRNN models actually uses a bidirection LSTM layer and that too twice. This arrangement is done inorder to learn longer contextx of the sentences. This means that for some problems even the LSTMs are not sufficient.

LSTMs actually originates back in 1970s but their such aa wide use has startes only just now. Tis, the season because of the advent of data present today along with higher computing powers like graphics processing units and tensor processing units.

We are using the LSTM cells in a encoder-decoder. The encoder-decoder model is a reltively new discovery in the result of text. What happens here is the model overall is divided into two parts, the Encoder and the Decoder as the name suggests. The encoder model encodes the input sequence into a vector. This vector, which contains the aggragated cell and hidden sate of the encoder part is then fed into the decoder side. This decoder is fed with this particular vector and also the target token of the target sequence. The model is trained as such that for a good prediction, the target sentences are prefixed and postfixed with a start and an end token respectively. The start token is used to initialize the translation prediction part and the end token is used to indicate that the desired ouput has been obtained and thus the algorithm stops. Each word of the target sentence is obtained at each time step of the decoder model intil the end token comes across.

The encoder-decoder model provides state of the art results for translation purposes. Text, however, cannot be fed direcly to the model. The input as well as the output text is encoded and represented in numbers. Techniques like word2vec can be used for this purpose. Word2Vec is actually a part of a broader concept word embeddings. Word embeddings not only provide the translation into number of the target text, they also learn the dependencies and similarities in word. For example, great and good can be used in a similar sentence. Therefore, word embeddings can learn such similarities in the dataset and represent them accordingly. Word embeddings are found out using neural networks. Word2Vec model uses a simple two layer mode(one hidden and one ouput layer) inroder to achieve the aformentioned

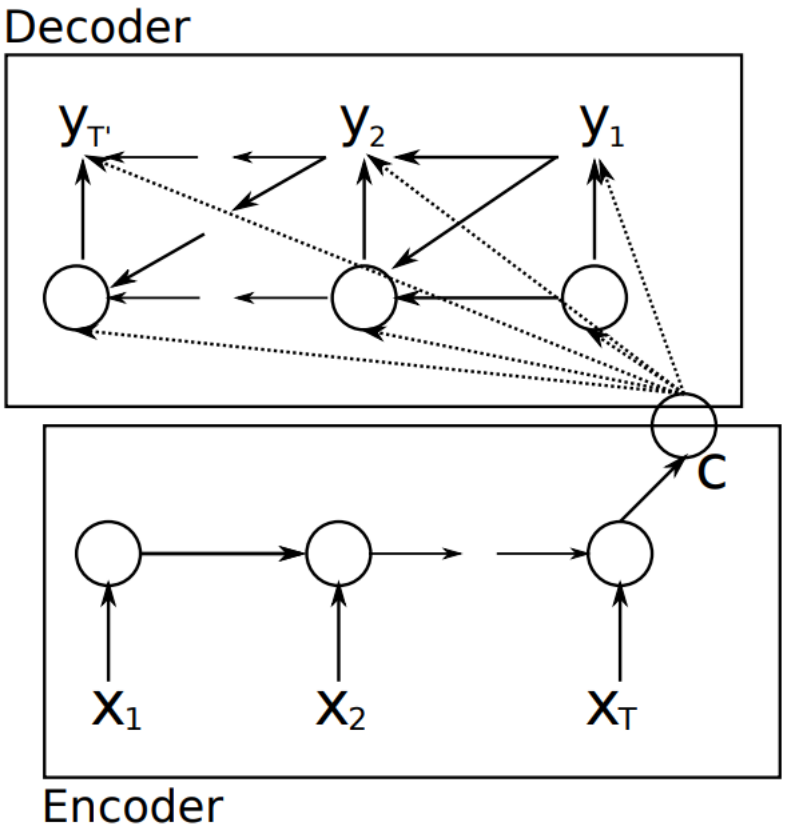


Fig2.1: The Encoder-Decoder Model

Another popular method is one-hot encoding. Unlike word embeddings, one-hot encodings are just representations of of the text in numerical form. Usually, this representation makes use of zeros and ones; one indicating presence of a particular word and zero marking the

absence. Is does not learn any deeper meanings or similarities like the word embeddings. This method often comes to the rescue while making an encoder for the ouput sequence like in the case of the final stage of out project, the neural machine translation.

Summarsing the methodology, we started with testing image detectin on images and then moced on to work with videos and produce results along with the recording through the front facing camera of the laptop. Then we moved to the next stage of the project, the text recognition part. Firstly, we tried using Tesserect ocr(optical character recognition). Because of poor predictions from the tesseract model, we moved on to CRNN model. The CRNN models provided decent predictions but the frame rate took a toll. Due to low frame rate, we tried to speed up the process using Cython and then threading. When we were unable to obtain a considerable speed up, we moved to using multiprocessing. Multiprocessing helped in the process by speeding it up many folds. Eventually the frame rate was brought down so that the text detaction and recognition part can work properly. The next phase is translatins. We use encoder-decoder model for getting accurate transalation. The model is being currently trained and we are using the dataset of English-Hindi sentences so that english can be converted to Hindi.

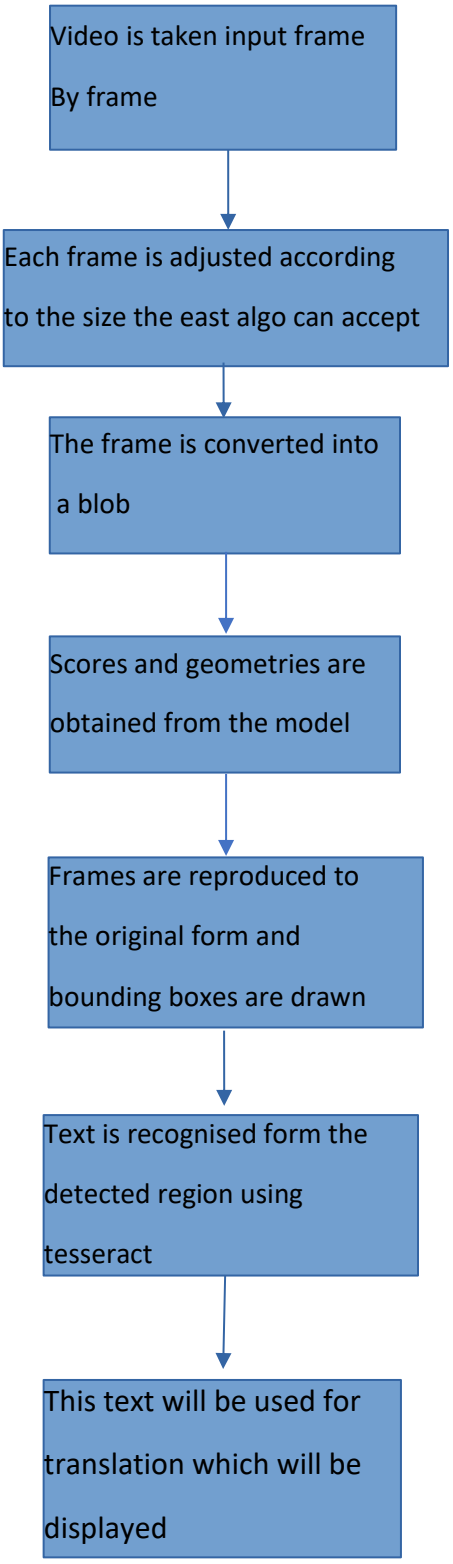
1.5 Organization

In Chapter 2, literature survey is discussed. We take into the consideration various current and state of the art approaches. We also look to various other methods inorder get the proper understanding of what are current method is.

In Chapter 3, System Development is discussed. Various libraries used and where they are used for what purposes. Our application is big application which conatins lots of code and this as many libraries. We have discussed them in detail in that section.

In Chapter 4, performance analysis is discussed. This topic too is discussed in detail here. Every output of every stage. Appropiate screenshots are also attached .

In Chapter 5, we have discussed the future scope what results we have got and what have shortcomings.


Video is taken input frame By frame
Each frame is adjusted according to the size the east algo can accept
The frame is converted into a blob
Scores and geometries are obtained from the model
Frames are reproduced to the original form and bounding boxes are drawn
Text is recognised form the detected region using tesseract
This text will be used for translation which will be displayed

CHAPTER – 2 LITERATURE SURVEY

Research Paper 1:

Text recognition based on image that is, the input is an image and the oupute we get is the recognized text has been a longstanding research topic in computer vision. The problem of scene text recognition, which is among the most important and challenging tasks in image-based sequence recognition is the expertise of this paper and the light is shed towards better methods and better appraoches giving faster and if not better, equally good results as previous methods. Previous methods being those methods that were into the scene before thhis method came along.. This task is often coupled with the text detection task but at the same time, also finds its use in many other text recognition based application or optical character recognotion based applications. Some properties of this method are:

(1) It saves a lot of time while training and giving comparable or much better results as it is one-shot trainable or end-to-end trainalblewhere most of the other algorithms are divided into phases which slows them down
(2) Arbitrary lenghts of input are good for this model although the height of the input has to be no more than 32.
(3) Whether lexicon free or not, this model give equally awesome results for both kinds of inputs
(4) The model genrated is much more suitable for practical purposes since it is small

The proposed neural system model is named as Convolutional Recurrent Neural Network (CRNN), since it is a blend of a Deep Convolution Neural Network (which means containing many layers) and Recurrent Neural Networks(since no more than 2-3 layers are used in any model). For grouping like items, CRNN has a few unmistakable points of interest over customary neural system models:

1) The algorithm works fine by working on a word level instead of going for the characters

2) Similar properties as DCNN on learning enlightening portrayals legitimately picture information, not one or the other hand-create highlights nor preprocessing steps, including binarization/division, segment limitation, and so on.;

3) The properties of RNNs are used like learning a large snetence.

4) It is unconstrained to the lengths of arrangement like articles, requiring just tallness standardization in both preparing and testing stages;

5) Input cann be of variable size

6) Contains considerably less parameters than a deepe CNN model, thus using lesser sotrage capacity

The CRNN model comprises mainly of three stage, the CNN part, the RNN part and the translation part. The CNN part takes in the input as images.Now remember, the input image must of the height 32 so that it can be fed to the model. The length can be arbitrary but the height needs to be 32. The CNN part produces feature maps which are then used to generate feature vectors. These feautre vector are 1X1 convolutionsal feature maps. These featiure vectors are fed into RNN. The RNN than gives a output which needs to be translated using some algorithms. This translated oupute is the final output we need.

In CRNN model, the segment of convolutional layers is developed by taking the convolutional and max-pooling layers from a standard CNN model (completely associated layers are evacuated). Such segment is utilized to separate a successive component portrayal from an info picture. Before being taken care of into the system, all the pictures should be scaled to a similar tallness. At that point a succession of highlight vectors is removed from the element maps delivered by the part of convolutional layers, which is the contribution for the intermittent layers. In particular, each element vector of a component succession is created from left to directly on the component maps by section. This implies the I-th highlight vector is the link of the I-th sections of the considerable number of maps. The width of every section in our settings is fixed to single pixel. As the layers of convolution, max-pooling, and elementwise enactment work on nearby areas, they are interpretation invariant. In this way, every section of the component maps compares to a square shape district of the first image (named the open field), and such square shape districts are in a similar request to their relating sections on the include maps from left to

right. Each vector in the element succession is related with an open field, and can be considered as the picture descriptor for that area.

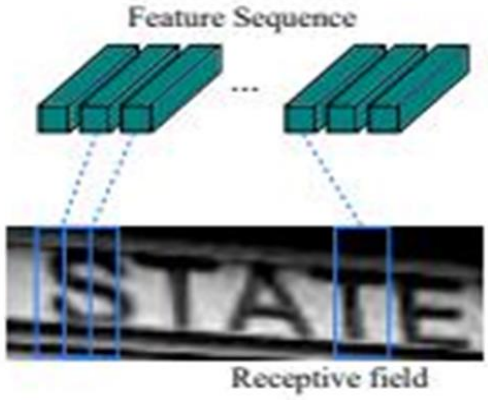


Fig 2.2: Feature sequence being extracted from feature maps

Deep convolutional neural networks have been widely adopted for different kinds of visual recognition tasks. The features ouput by these nets are robust, rich and are easily trainable. Some earlier approaches have employed CNN to learn a robust representation for sequence-like objects such as scene text. However, these approaches usually extract as a whole representation of the whole image by CNN, then the local deep features are collected for recognizing each element of a sequence like object. Since CNN requires the input images to be scaled to a limited size in order to fulfill with its fixed input dimension, it is not appropriate for sequence-like objects due to their large length variability. In order to be invariant to the length variation of sequencial objects in CRNN, we convey deep features into sequential representations.

A Bi-Recurrent Neural Networke is manufactured on the highest point of the convolutional layers, as the repetitive layers. The repetitive layers anticipate a name circulation of the output for each casing in the element arrangement of X . The favorable circumstances of the repetitive layers are three-crease. Right off the bat, RNN has a solid ability of catching logical data inside an arrangement. Utilizing logical prompts for picture based grouping acknowledgment is more steady and accommodating than treating every image autonomously. Taking scene content acknowledgment for instance, wide characters may require a few progressive casings to completely portray. Also, some equivocal characters are simpler to recognize while watching their unique circumstances, for example it is

simpler to perceive "il" by differentiating the character statures than by perceiving every one of them independently. Furthermore, RNN can back-engenders blunder differentials to its info, for example the convolutional layer, permitting us to together train the intermittent layers and the convolutional layers in a brought together system. Thirdly,

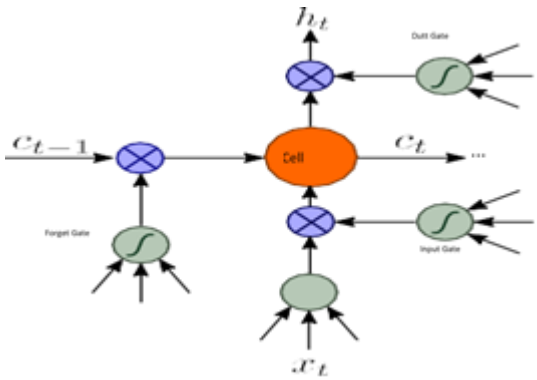


Fig 2.3: A LSTM Cell

This network is able to operate on different inputs and can produce different outputs for each input. In a one-to-one model a fixed input resluts in a single ouput, like reading the sentiment of an image. I a one-to-many setting, an input of fixed size can give a varied length output like generating a caption from an image. In many-to-one model, a varied length input will result in only one ouput. These moels are used mainly in rating what sentiment of a text input. And lastly, many-to-many setting where many inputs will result in many iutputs. Like sequence modelling or translation like in the case of this project.

In the scope of this project, man-to-one and one-to-many models are contaenated and form what is called and encoder-decoder model. The vareid length input of the source language is trurned into the one vector which fed to the cell state and the hidden state of the decoder model. This decoder model takes this as input along with the ttarget language during the process of training. The testing process becomes cumbersome as the training mode has to be sampled fo rthe predictions. The tricky part is to use to use it in a different file and thus integration of the translation part is resulting to be difficult.

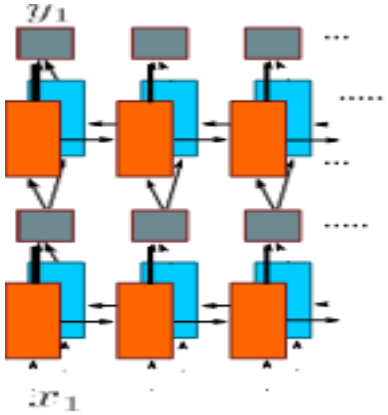


Fig 2.4: The BRNN architecure

LSTM is directional, it only uses past contexts. However, in image-based sequences, contexts from both directions are useful and complementary to each other. Therefore, we follow and combine two LSTMs, one forward and one backward, into a bidirectional LSTM. Furthermore, multiple bidirectional LSTMs can be stacked, resulting in a deep bidirectional LSTM. The CRNN paper also mentions how it can be trained on a music-notes input thus increasing the scope of this model.

The backpropagation through RNN is termed as back propagation through time since it is said that every input occurs at a new timestep. The backpropagation through time might seem complex to a beginner but it is fairly easy now that it can be done batch wise. In 2014 it was found by some researchers that the backpropagation through time too can be done batchwise. Andrej Karapathy, a renowned name in the field of deep learning promotes this by writing the code in merely 180 lines of python that too using numpy.

Research Paper 2:

YOLO is a rather different approach in the field of text detection. Where each and every of the previous approaches have been manually driven that is the features of object detection were typed in manually, Yolo promotes the deep learning approach where the machine is forced to learn the features on its own thus giving better results.

Yolo is an end-to-end trained pipeline. This means that where each of the existing model had several phases which increased the training time and thus the time for ensembling all the phases together. It eliminates all the intermediate steps and learns faster. This yolo algorithm is a real-time algorithm which means it detects object while still polling frames

in the mix. It works at 45 frames per second at higher resolutions and can move up to more than 100 frames per second at lowers resolution frames. A lighter version of YOLO exists which is termed as fast-Yolo. This model is trained using less weights thus resulting in lesser parameters and therefore, justifying the name.

At long last, YOLO learns very general portrayalse. Not only beats its competetion identification techniques, including DPM and R-CNN, but aloso summing up the characteristic pictures. The Convolutional Network used is able to produce bounding boxes by giving its dimensions along with the angles it has to arranged. This model has a few advatages over the previous models. Starting with what's better, it is insanely fast. Since we outline location as a relapse issue we needn't bother with a perplexing pipeline. We basically run our neural system on another picture at test time to anticipate recognitions. Our base system runs at 45 outlines every second with no bunch handling on a Titan X GPU and a quick form runs at in excess of 150 fps. This implies we can process spilling video progressively with under 25 milliseconds of inertness. Moreover, YOLO accomplishes more than double the mean normal exactness of other constant frameworks.

The second argument to the case of Yolo is that this algorithm goes about the whole picture as a whole. This means that YOLO sees the whole picture or the entire image in one go while both, the training and testing times, unlike the other protocols like sliding window protocols and other area based methods. This implies that it encodes thte data present int the image just like they appear in the image. A top notch algorithm in the field of object detection, Quick-RCNN, badly carries out the task of foundation fixes because it is unable to the picture asa whole as it is a sliding window protocol. Yolo on the other hand too makes the foundation mistakes but they are not of the scale of the Fast-RCNN.

Third argument in this series in the favour of YOLO is that it is able to generalize the representation of an object. In a task where YOLO was trained on normal picture and was tested on fine arts, the algorithms beats the best in game slgorithms like DPM and crushes the prediction made by RCNN. This generalizability of the algorithm makes it more averse to separate when applied to new areas or applied to aread it has never seen before. In the

matters of precision thoigh, it still lags behind. It is not able to identify smaller objects which is dicussed at the end of the discussion for this research paper.

They bind together the different segments of article identification into a solitary neural system.

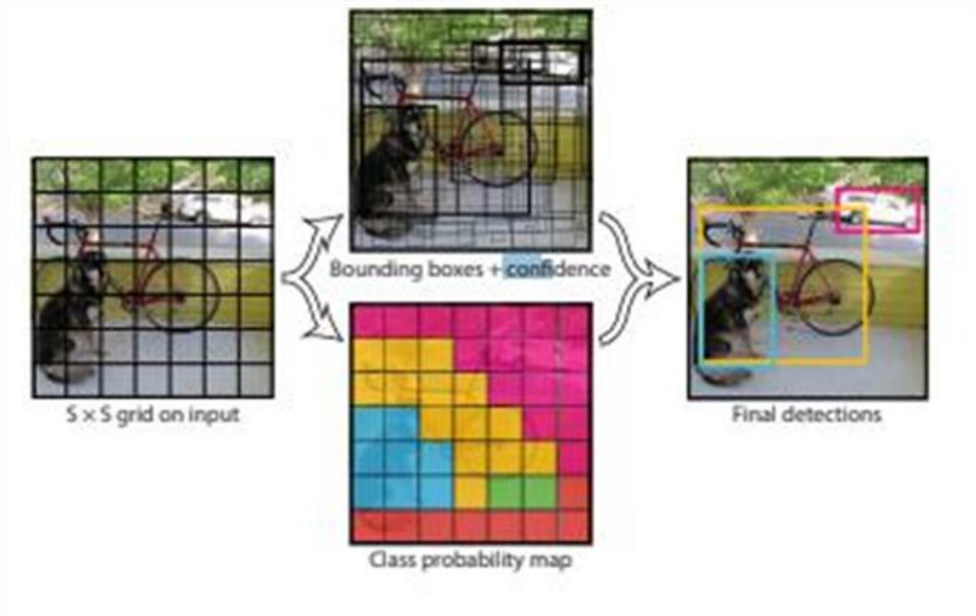


Fig 2.5: Flow Control for YOLO

Assuming the cell does not have any object, the certainty tend to be zero. Else need for the certainty arises to the convergence over association (IOU) between the anticipated boxes furthermor. Every bouncing boxe comprises of 5 forecasts. The (x, y) organizes speak to the inside of the case comparative with the limits of the lattice cell. The width what's more, tallness are anticipated comparative with the entire picture. At long last the certainty expectation speaks to the IOU of the anticipated box and other box.

Every network cell just predicts two boxes what's more, can just have one class. Thise spatiale imperative constrainse thee quantiety of close by objects.It is not able to detect properlu a cluster of small objects like a a flock of bords flying in the sky of some picture or the small parts of a mutual funds hoarding. Downsampling is used at a major scale in the model therefore upsampling is used to produce the final outpu. The errors are treated well in the output. No Max suppression and other techniques like that are used to obatin the best output. Some confusions can occur when multiple objects have to be detected in a smaller sqaure. What algorithms does is it divides the picture into many smaller sqaures and the

CNN is run over the image that is for each and every smaller boxes. Earlier every box was run in the network one by one which took a lot of time. So here what happens is these smaller boxes are given in all at the same time to the network along with the anchor boxes. Now the disadvantage is that we no other object which diverts from the behavious of the anchor boxes will be detected in the process. And also, if there are multiple objects in one smaller square, that raises another problem for the algorithms. The solution to this is to divide the images into more smaller boxes. Therefore, this is the reason why cluster of objects sometime go unidentified in the process of object detectoin.

Research Paper 3:

Past methodologies for scene content location have just accomplished promising exhibitions across different benchmarks. Be that as it may, they as a rule miss the mark when managing testing situations, in any event, when furnished with profound neural system models, in light of the fact that the general execution is controlled by the interchange of different stages and parts in the pipelines. The pipeline proposed by the paper is simple yet groundbreaking. What they have done is that they have removed all the useless intermediate steps and in return they have introduced a model that is en-to-end trainable. It was born as a result of the competition which was ICDAR 2015. The dataset also goes by the same name. The east model is mainly divided into three parts, the first part is an adaptation from the PVANet the second part accumulates the weights while upsampling them to produce the last part which is output of the model. The ouput is 4 different object used to build the bounding box around the detected text.

Existing techniques, either traditional or profound neural system based, generally comprise of a few phases and parts, which are most likely problematic and timeconsuming. Along these lines, the exactness and productivity of such strategies are still a long way from agreeable. propose a quick and precise scene content identification pipeline that has just two phases. fully convolutional network (FCN) pipeline model that legitimately delivers word or content line level expectations, barring repetitive and moderate middle advances. The delivered content expectations, which can be either pivoted square shapes or quadrangles, are sent to Non-Maximum Suppression to yield conclusive outcomes. Contrasted and existing strategies, the proposed calculation accomplishes essentially

improved execution, while running a lot quicker, as indicated by the subjective and quantitative analyses on standard benchmarks.

The keye componenet ofe thee proposede algorithme ise ae neurale networke modele, whiche ise trainede toe directlye preedicte thee existencee ofe texet instancese and theire geometriese frome fuell imagees. The model is a completely convolutional neural system adjusted for content recognition that yields thick per-pixel forecasts of words or content lines. This disposes of middle of the road steps, for example, up-and-comer proposition, content locale development also, word segment. The post-preparing steps just incorporate thresholding and NMS on anticipated geometric shapes. The finder is named as EAST, since it is an Efficient and Accurate Scene Text detection pipeline.

An elevated level diagram of our pipeline is delineated in. The calculation followse thee generale structure of DenseeBox, a picture is taken care of into the FCN and various channels of pixel-level content score guide and geometry are created. One of the anticipated

Fig 2.6: Output by the EAST on various datasets

channels is a score map whose pixel values are in the scope of [0, 1]. The rest of the channels speak to geometries that encases the word from the view of every pixel. The score represents the certainty of the geometry shape anticipated at a similar area. We have tried different things with two geometry shapes for content districts, turned box (RBOX) and quadrangle (QUAD), what's more, structured distinctive misfortune capacities for every geometry. Thresholding is then applied to each anticipated district, where the geometries

whose scores are over the predefined edge is viewed as legitimate and put something asid for later non maximum-concealment. Results after NMS are thought of the final output of the pipeline.

Research Paper 4:

Recurrent neural systems with Long Short-Term Memory (which we will succinctly allude to as LSTMs) have developed as a powerful and versatile model for a few learning issues identified with successive information. Prior techniques for assaulting these issues have been custom either fitted towards a particular issue or then again did not scale to long time conditions. LSTMs on the other hand are both general and powerful at catching long-term transient conditions. They do not experience the ill effects of the improvement leaps that plague straightforward intermittent systems SRNs) and have been utilized to propel the state-of the-craftsmanship for some troublesome issues. This incorporates handwriting recognition and generation, language displaying what's more, interpretation, acoustic modelling of speech, speech synthesis, protein optional structure expectation, investigation of sound, and video information among others. The focal thought behind the LSTM engineering is a memory cell, which can keep up its state after some time, and

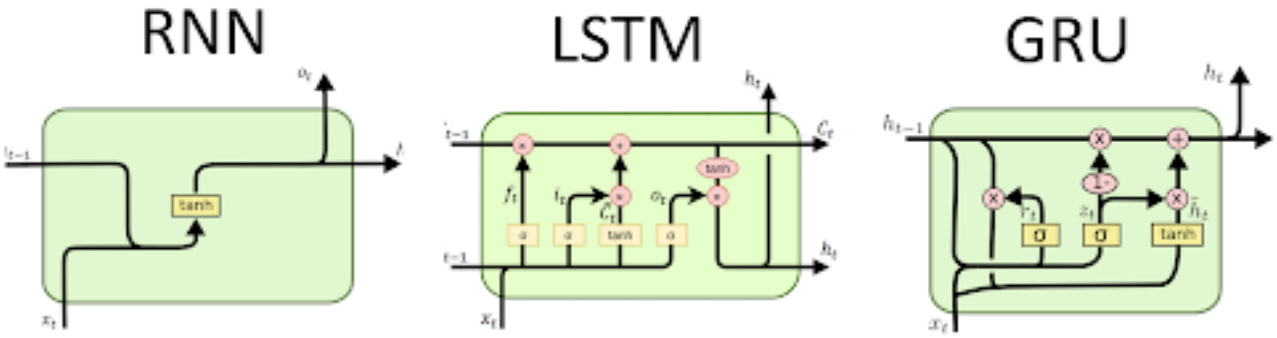


Fig 2.7: A comparison between al the types of RNN cells

non-straight gating units which manage the data stream into and out of the cell. Most current examinations join numerous enhancements that have been made to the LSTM design since its unique plan. Be that as it may, LSTMs are presently applied to many

learning issues that vary essentially in scale andnature from the issues that these upgrades were at first tried on. A deliberate investigation of the utility of different computational

parts that involve LSTMs was absent. This paper fills that hole and deliberately addresses the open inquiry of improving the LSTM engineering.

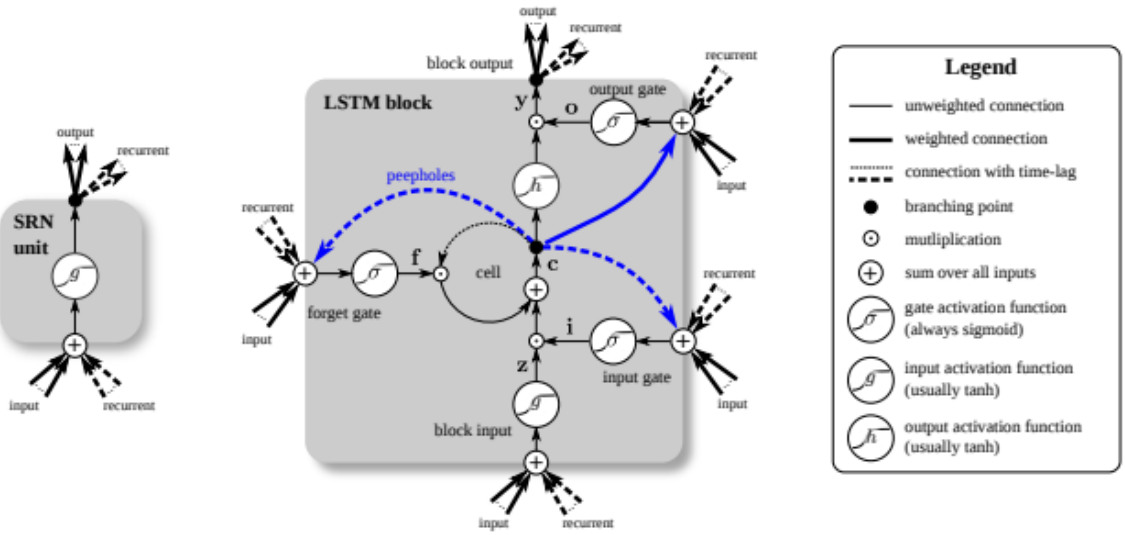


Fig 2.8: LSTM as represented in paper

We start with the forget gate. It tell what input to keep and what input to discard. It will be learned as we move along the training, that is the exact value the forget gate should have. The forget gate inputs the previous cell states and decides what to do with eventually. Wheter it os some use or not in predicting the sequence.

The gates are just a combination of no linearities of tanh and softmax. These gates mainly help in back propagation by preventing the gradient explosion or preventing the diminishing gradient.

What happened earlier was, when the cell state was absent, the back propagatin was done through the hidden state. Therefore, weight was collected at every time step and in the end the weight matrix was multiplied by hidden state as the gradient comes out to be the multiplication of both the matrices. Now this multiplication would have resulted in two cases. Either the gradient would have vanished or it would have exploded.

But in LSTM, the cell state creates a highway for the gradient to flow. This prevents both of the phenomena mentioned above. This way LSTM learns the nect state.

Now we ought to have enough data to compute the cell state. To start with, the cell state gets pointwise duplicated by the overlook vector. This has a chance of dropping qualities in the cell state on the off chance that it gets duplicated by values almost 0. At that point we take the yield from the information door and do a pointwise expansion which refreshes the cell state to new qualities that the neural system finds pertinent. That gives us our new cell state.

Last we have the output gate. The yield door chooses what the following shrouded state ought to be. Recall that the shrouded state contains data on past sources of info. The concealed state is additionally utilized for expectations. To begin with, we pass the past shrouded state and the present contribution to a sigmoid capacity. At that point we pass the recently changed cell state to the tanh work. We duplicate the tanh yield with the sigmoid yield to choose what data the shrouded state should convey. The yield is the shrouded state. The new cell state and the new hidden state is carried over to the next time step.

The essential thought of bidirectional recurrent neural nets is to introduce each preparation grouping advances and in reverse to two separate intermittent nets, the two of which are associated with a similar yield layer. This implies for each point in a given arrangement, the BRNN has total, consecutive data pretty much all focuses when it. Additionally, in light of the fact that the net is allowed to use so a lot or as meagre of this setting as fundamental, there is no compelling reason to discover a (task-subordinate) time-window or target defer size.

For transient issues like discourse acknowledgment, depending on information on the future appears from the outset sight to damage causality … How would we be able to base our comprehension of what we've heard on something that hasn't been said at this point? In any case, human audience members do precisely that. Sounds, words, and even entire sentences that from the start amount to nothing are found to bode well in the light of future setting.

Research Paper 5:

Optical character acknowledgment is the mechanical or electronic transformation of pictures of composed, written by hand or printed content into machine-encoded content [8].It is the easiest method to digitize printed and handwritten texts so that they can be easily searched, more compactly stored, displayed and edited, and used in various other preparing undertakings, for example, language interpretation and content mining .

Thresholding is used as the initial step which turns the picture into a tweo folde forme. Subsequente stage is page format examination, thaet is utilized to separate contente squarese

In the next step the baseline of each line is traced and the text is divided into words using fixed space and fuzzy space.

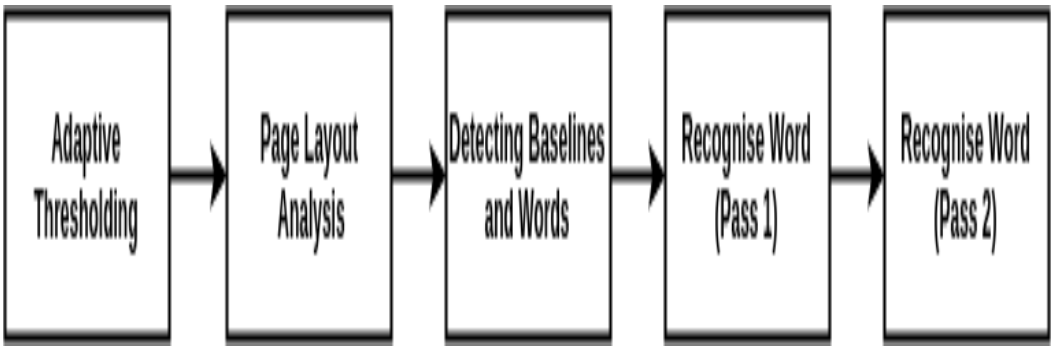


Figure 2.9: Architecture of Tesseract OCR engine.

In the next step, the character outline is extracted from the words. The recognition of the text is then initiated as a two-pass process. The first password is validated using static classifiers. Each word passed satisfactorily is given to an adaptive classifier in the form of training data . A new pass is run on the page, using newly learned adaptive classifiers in which words that were not well recognized are re-identified.

Format investigation of the page in Tesseracte depends on recognizing tab-stops in an archive picture. It has the accompanying steps:

The morphological preparing from Leptonica distinguishes the vertical lines and the pictures. These components are expelled from the information picture before passing the cleaned picture to associated part investigation.

The competitor tab-stop associated parts that seem as though they might be at the edge of a book area are found and afterward assembled into tab-stop lines.

Examining the associated segments from left to right and top to bot-tom, runs of comparably characterized associated segments are assembled into Column Partitions (CP), subject to the requirement that no CP may cross a tab stop line.

Chains of content CPs are additionally isolated into gatherings of uniform line-dividing, which make content squares. Presently each chain of CPs speaks to an up-and-comer area. The perusing request of the areas are controlled by some heuristic principles.

Words are fragmented into characters here.

Content line are tested by tesseracte to test if they have a fixed pitch that is having a consistent separator among the words. Tesseeract hacks the words into characters utilizing the pitch for the content sporting a fixed pitch. The remainder of the word acknowledgment step applies just to non-fixed-pitch content.
The outcomes are introduced to a lexicon after characterizing each piece of text to search to discover a word in the blends of a classifieer decisions for eeach mass in the word. While the word result is unacceptable, Tesseeract cleaves the mass with most exceedingly awful certainty from the character classifier.

Up-and-comer slash focuses are found from sunken vertices of a polygonal estimate of the blueprint.

In this part the identified words are fragmented into characters.

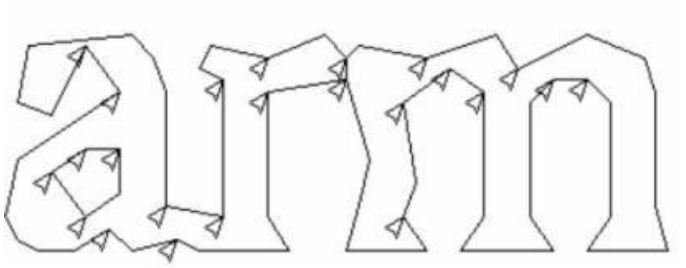


Figure 3.0: Candidate cut points.

Some part of tesseracte has already been published in a research paper. The algorithm is designed to find the line so that the skewed page can be identified without de-skew, thus saving image quality loss. The major parts of the process are droplet filtering and line construction. Accepting that page format examination has just given content territories of generally uniform content size, a straightforward rate tallness channel evacuates drop-tops and vertically contacting characters. It is safe to drop those characters having a height more than the median height or the average height simce they are more likely to be punctuations and special symbols.After the negatives of the parts if the images have been filtered they can be moved back to the baseline. They moved from the baseline so that they can be processes upon and correct text can be recognized. The last advance of the line creation process covers at any rate half on a level plane, assembling the exceptional bases with the right base and accurately affiliated pieces of some messed up characters.

Curved baselines can also be found or detected by the algorithm since it now has the text lines that are fitted into the base lines. These curved baselines can be found everywhere in the real world like book binding, water bottles, soft drink bottle stickers, hoardings in the wild, posters of movies, psychedelic posters, armbands, headbands, wristbands. In short any of the surface you can find to print text upon. Fitted are baselines by dividing groups into groups with reasonably constant displacement for the original straight baseline. A quadratic frame is fitted with the most populated division (considered the baseline) by a least squares fit. The advantage in quadratic Spanish is that this calculation is reasonably stable, but inconvenience may arise when multiple split segments are required. A more traditional cubic sline may work better.

Tesseracte tests the content lines to decide if they are fixed pitch. Where it finds fixed pitch content, Tesseeract slashes the words into characters utilizing the pitch, and debilitates the

chopper and associator on these words for the word acknowledgment step. Fig. 2 shows a commonplace case of a fixed-pitch word.

Variable-pitch or relative content dividing is a profoundly non-paltry undertaking. outlines some regular issues. The hole betweeen the tens and units of '12.0%' is a comparative size to the general space, and is unquestionabley bigger than the kernel. not even hole at all between the jumping boxes 'of' and 'money related'. Tesseeract tackles the majority of these issues by estimating holes in a restricted vertical range betweeen the standard and mean line. Spaces that are near the limit at this stage are made fluffy, with the goal that a ultimate choice can be made after word acknowledgment.

CHAPTER – 3 SYSTEM DEVELOPMENT

The System to make the project is developed upon a machine with i5 fifth generation quad core processor and eight Gigabytes of RAM(Random Access Memory). The machine has one terabytes of hard disk. Upon this disk, two operating systems have been installed namely, Windows 10 and the Linux base operating system Ubuntu 20.04(Focal Fossa). The Ubuntu partition is allotted almost 150 Gigabytes of the hard disk and is logically divided into two parts, root and home. The machine is also installed with a web camera which can record videos as 720p at high frame rates.

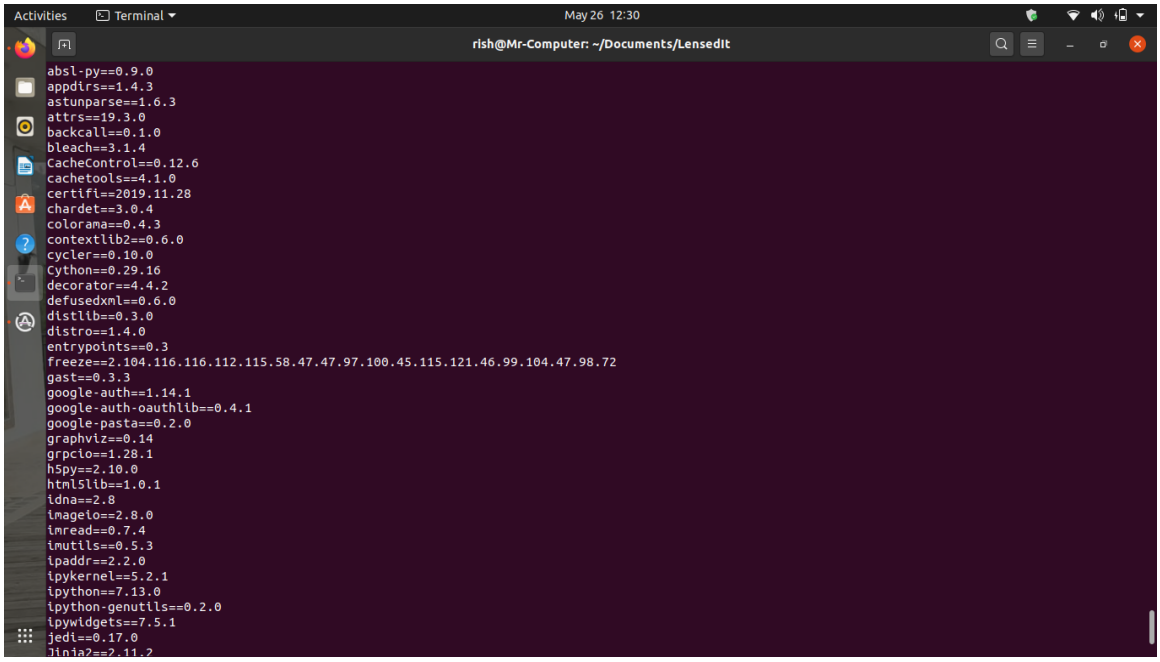


Fig 3.1: Packages used-I

For the purposes of the development of the project, Ubuntu is used. A virtual machine is set up which includes all the packages used for the project. Virtualenv package was installed into the system first which was used to establish the virtual environment. A project should always be made on a virtual environment because it provides absolutely different area to work upon. This means that, for example, even though python comes pre-installed with Ubuntu, I will not be able to use it on my virtual machine. I will have to install each an every package sperately. Though, many packages like wheel and Werzeug get installed with the virtualenv package.

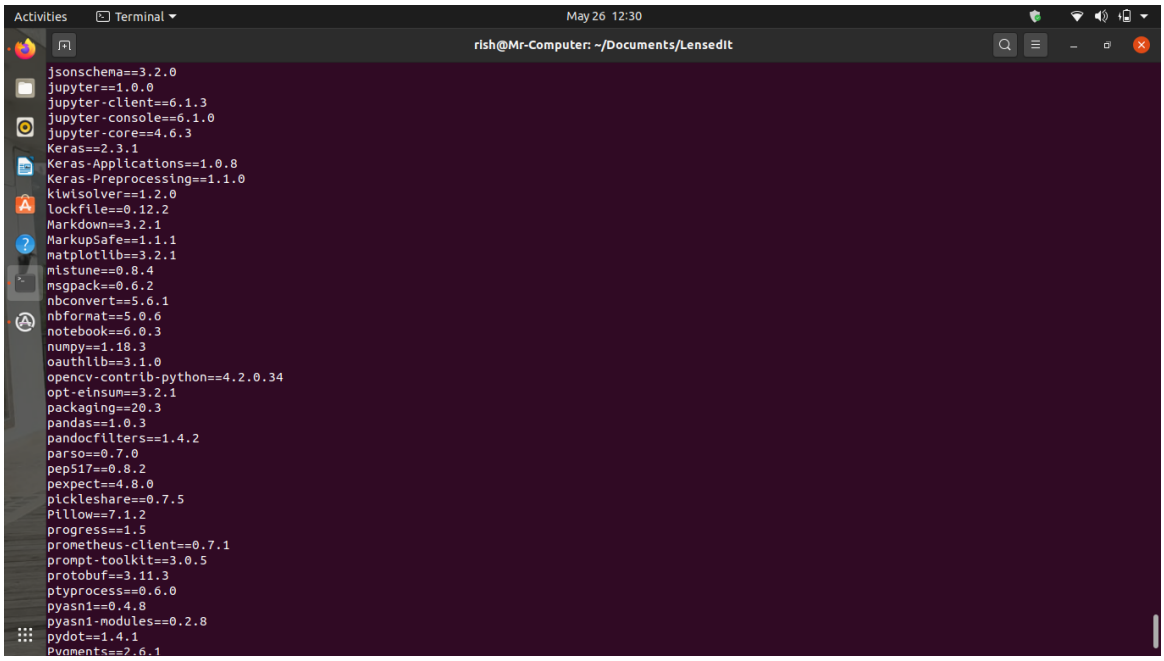


Fig 3.2: Packages used-II

After establishing the virtual environment, I started installing basic packages; python(python 3.6.0), numpy(numpy1.18.3), scipy(scipy 1.4.1), both of which are used for complex calculations to be done in an easier looking way, seaborn(seaborn 0.10.1), used for represented the data using graphs, scikiit-learn(scikit-learn 0.23.1) and sklearn, that are used for including pre-trained models sometime and pandas(pandas 1.0.3) from loading and manipulating mainly csv(comma separated values) file data.

As we moved forward with the project we installed libraries like opencv(opencv-contrib-python==4.2.0.34), used to handle the videos and image part of the project. Mainly used in initial stages to process the image and capture vides through webcam. VideoSteram library was used to record the video from the webcam and its attributes like FPS is used to find the frames per second. Multiprocessing library is used for the outsourcing of the processing part of the video so that the FPS can be increased. Some classes are also used off this library for the shared variables like thin white lies and best years. OpenCV is also used to load the EAST model and take out the names of the layers needed. Numpy is used fro preprocessing parts mainly. For making some numpy arrays that can be later fed to the model. Datetime packages is used to calculate the time taken by the preprocessing task

while detection. We used Cython(Cython==0.29.16) to speed up the process and increase the frame rate. A couple of classes and methods were converted to a cython code.

For the recognition phase of the project, pytesseract(pytesseract==0.3.4) and tesserocr(tesserocr==2.5.1) were used initially. Pytesseract is basically a wrapper of the package of tesserocr. This means that every functionality of tesseract is originally written

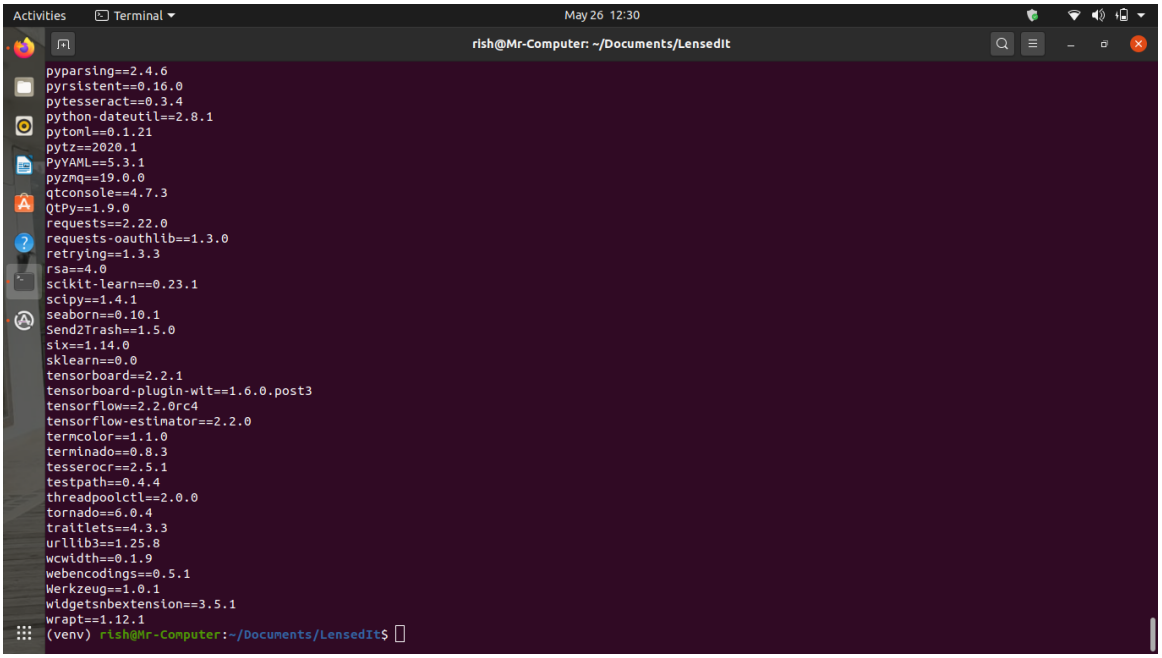


Fig 3.3: Packages used-III

in the library tesserocr and can be used directly by using tesserocr but pytesseract is used for a simpler and easier user interface. Since the predictions were bad, we moved on to

using CRNN. The CRNN model was made using the functional API of the tensorflow(tensorflow==2.2.0rc4). Inorder to make the model, Input, CNN, BidirectionalRNN, LSTM and Dense layers were used and thus were imported from tf.keras.layer. Model class defined the model and .save and load_model methods are used to save and load the model respectively.

Keras is used for translation puposes. Keras is nothing but a high level version of tensorflow. It is easier to use and understand and is especially useful for beginners. The layers LSTM, TimeDistribiuted, Dense, RepeatVector and Embedding are used. A sequential model is used in this case. The data used is for English to Hindi conversions of the recognized text. The model is trained upon 9216 examples and cross validates using 1024 examples on google colab notebook. Google colaboratory is a cloud platform that provides with 12 Gigabytes of RAM and 30 Gigabytes of disk space. The google colab notebook is free to use and also provides GPU(Graphics Processing Unit) and TPU(Tensor Processing Unit) support.

EAST was trained on ICDAR Dataset. The dataset incorporates around 1000 characteristic pictures, which are reaped from the Internet or taken by volunteers. 500 pictures are chosen for calculation advancement also, approval, and 484 picture for testing. For each picture, the polygons and substance of all the content lines inside it are commented on. It is permitted to utilize additional information for preparing in this rivalry. The content lines might be separated into one of the four classifications: (1) Tranpapernt English; (2) Non-Transparent English; (3) Transparent Other; (4) Non-Transparent Other. The classifications with "Translucent" show the nearness of translucent content, which might be utilized to encode site interface, name of the shop, contact data, and so forth.. Perusing the encoded content will help decide whether the content is in accordance with the counter spam approach of the site facilitating the pictures. The classes with "Other" demonstrate the nearness of multilingual content involving Chinese and English in common/Internet pictures. Whereas, the translation model was trained on a kaggle dataset. It consist of 150000 images form three different sources. IT is a CSV file and there are only two columns one containing Hindi sentences and the other containing English sentences. We

use the data collected from the source “ted” which contain nearly 40000 entries of the language couple. The translation model is trained on 10000 images.

CHAPTER– 4 PERFOMANCE ANALYSIS

- We started with detection using EAST at 2.5 FPS

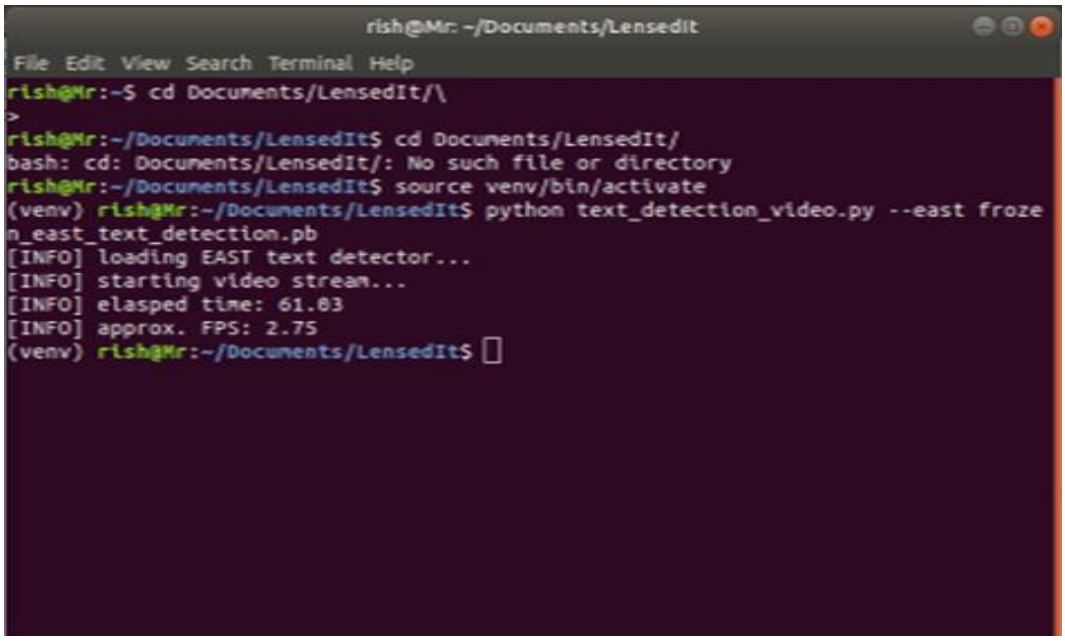


Fig 3.4: Result of text detection at initial stages

- Following are the detection results with the frame rate mentioned above:

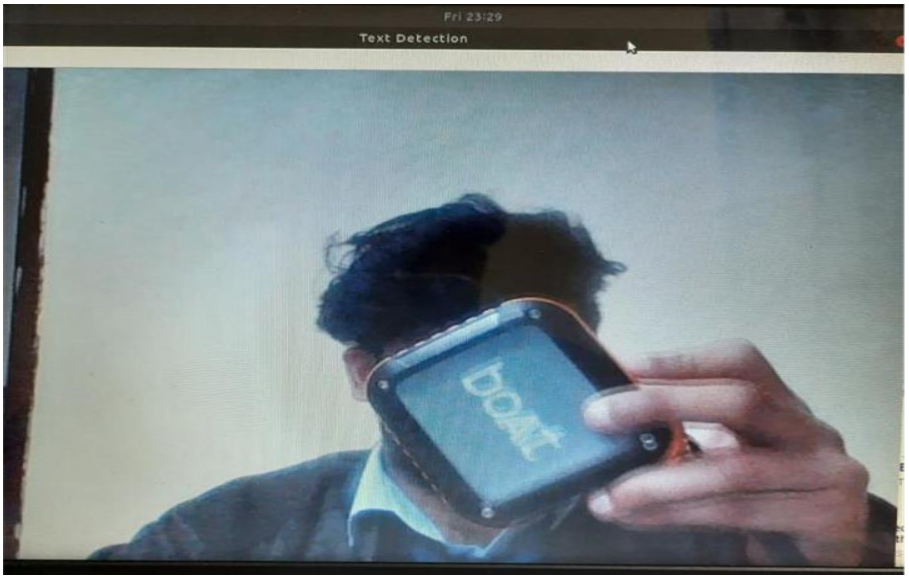


Fig 3.5: EAST text detection at skewed angles

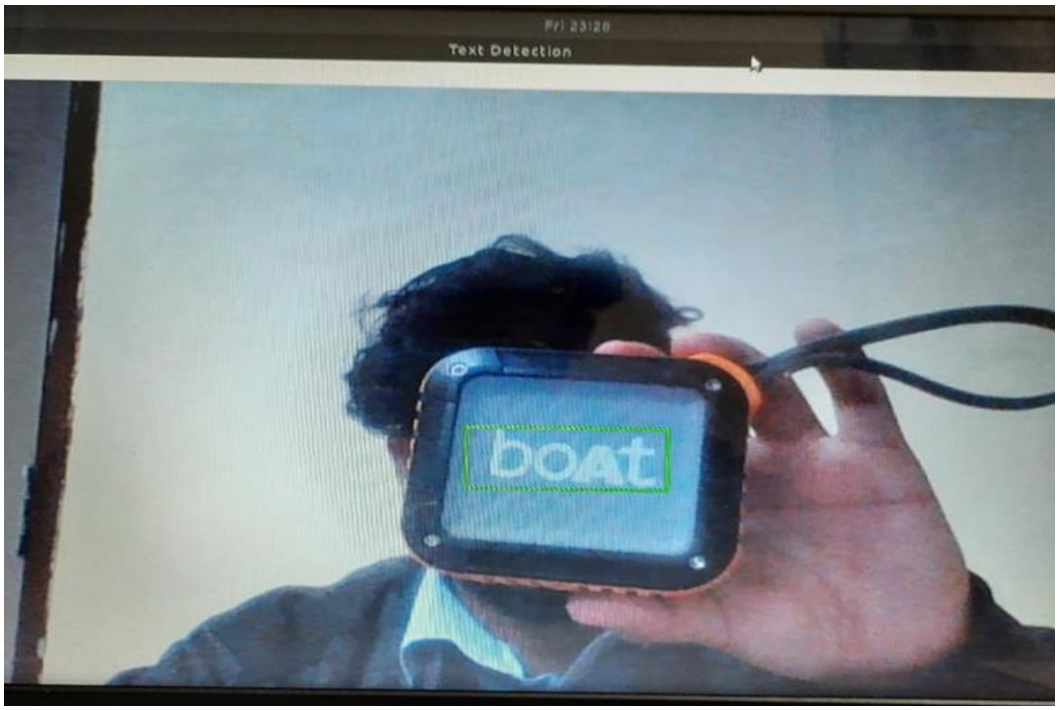


Fig 3.6: EAST text detection at horizontal angle

Fig 3.7: EAST text detection at vertical angle

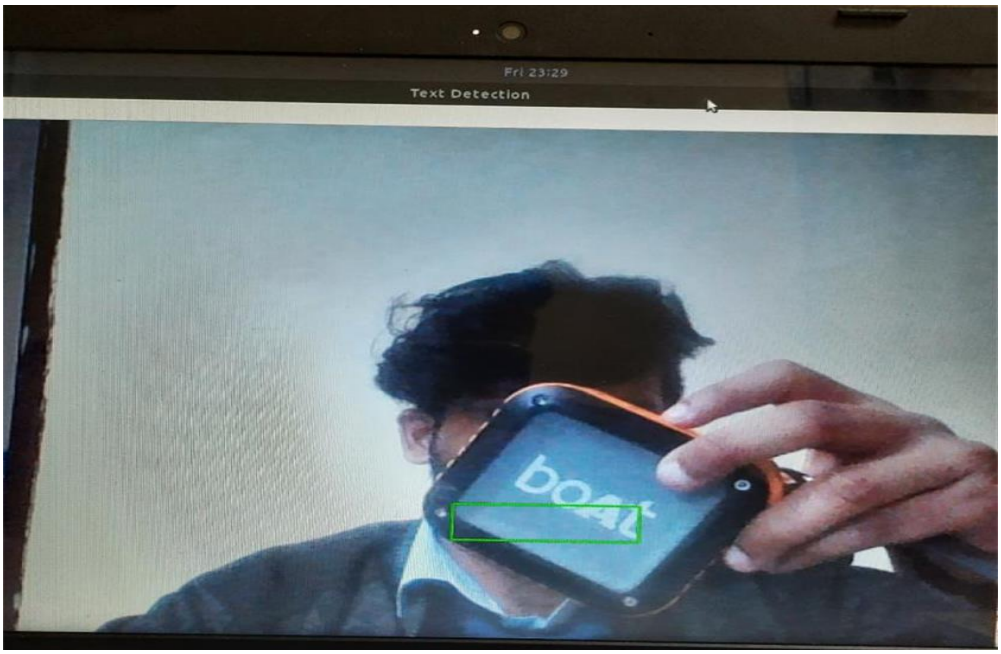


Fig 3.8: EAST text detection at skewed angle

- After incorporating text recognition and speeding up with cython along with multiprocessing, we climbed up to 13.2 FPS:

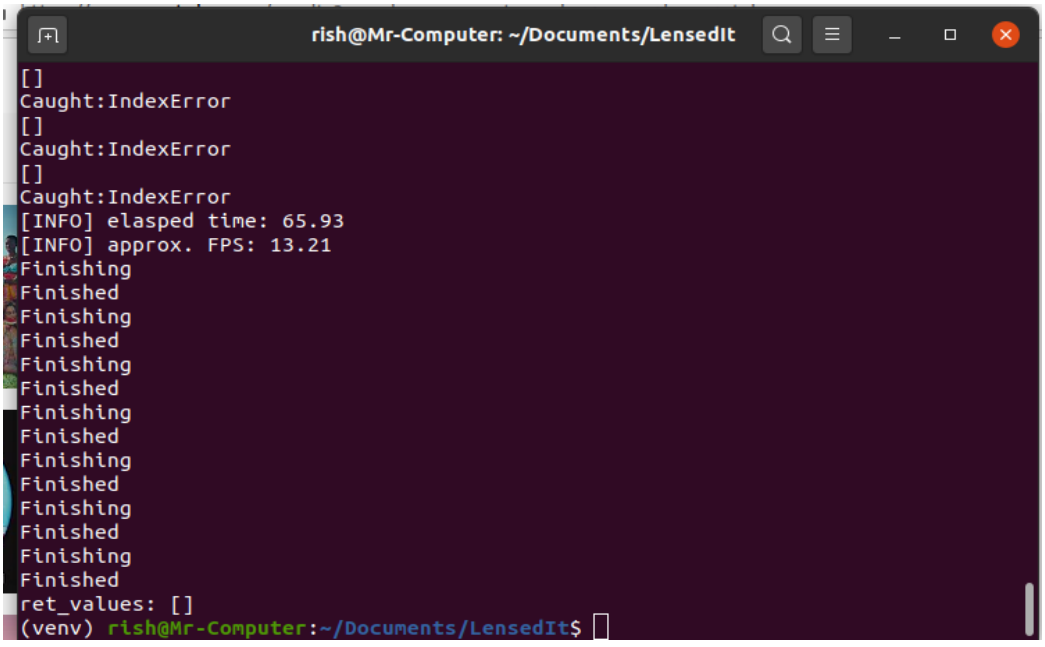


Fig 3.9: Indicating video recording rate

- We get the following results:

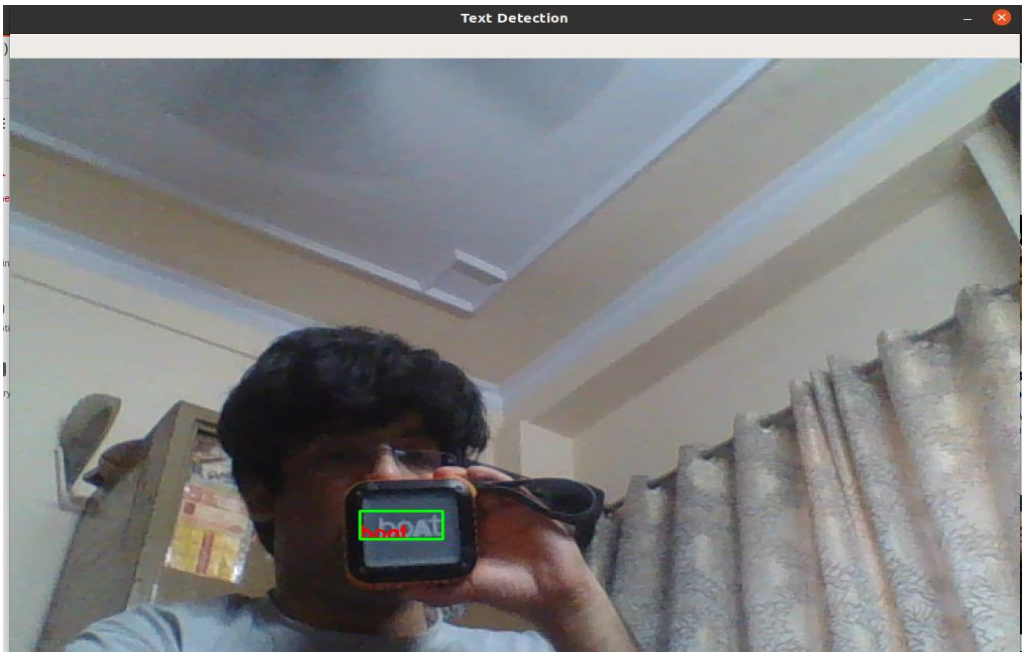


Fig 4.0: Text recognized as “boat”

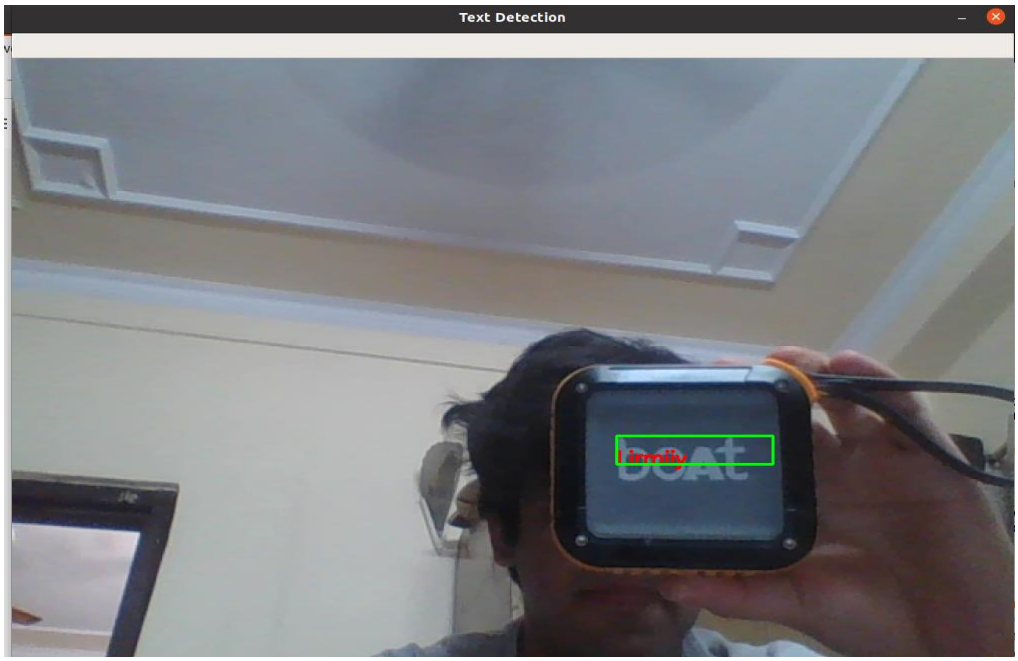


Fig 4.1: Text recognized as “l irmiiy”

- The translator for converting English to Hindi is still under development.

CHAPTER – 5 CONCLUSIONS:

5.1 Conclusions:

The project is divided into mainly 3 phases, text detection, text recognition and translation of the text obtained. The first phase of text detection works perfectly for horizontally and vertically aligned texts. Even for the texts that are relatively larger or smaller. Works also for curved surfaces and unclear text. The text detection goes haywire if the detected text is skewed. This is box the implementation of making a bounding box is such that only the sin and cosine part are take into consideration and not the angle the box as whole makes. Secondly, the implementation of Non-Max suppression is a bit different than the actual implementation since the actual implementation was using C++ libraries.

The seconds phase the project i.e. text recognition works perfectly fine for the texts that can be detected perfectly at a box height of 32. Thus, erroneous predictions are made for larger text. The recognizer fails to identify vertically aligned text and skewed text.

The third phase of the project i.e. translation is still under making. Major problem being faced is to replicate the sample model of the original model.

5.2 Future Scope:

- Better algorithm to bound the detected text can be made.
- A major limitation of the project is that it is unable recognize vertically aligned and skewed text.
- The parallel processing step makes a lot of unnecessary processes since it runs like busy waiting. Definitely a better method can be found to decrease these numbers of processes significantly. These process can use up a lot of CPU without doing anything of much significance thus wasting many cycles.
- Definitely translator model can be better trained. As of now 10000 sentences of both the languages are used thus the vocabulary sizes are equally small. Therefore, translator can be trained on more sentences so that vocabulary can be expanded.
- The translator can also be incorporated with many other languages which will increase the scope for this application
- There are three different pipelines being used in the project at the moment. One for text detection, one for text recognition and one for translation. Maybe one pipeline can be found for the task as a whole or the recognition and translation pipeline can be combined somehow(although, the loss functions used for both the tasks which means different kinds of outputs are being produced).
- A reader can also be integrated to the project that will read out loud the translations.

References:

- www.wikipedia.com
- www.arxiv.com
- www.coursera.com
- www.keras.io
- www.machinelearningmastery.com
- www.pyimagesearch.com
- www.vidhyanalytics.com
- www.medium.com
- www.towardsdatascience.com
- www.tensorflow.org
- docs.python.org
- docs.opencv.org
- cython.readthedocs.io
- cython.org
- noahsnail.com
- www.youtube.com

P

ORIGINALITY REPORT

10%	3%	7%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	noahsnail.com Internet Source	3%
2	Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, Jiajun Liang. "EAST: An Efficient and Accurate Scene Text Detector", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017 Publication	1%
3	R. Smith. "An Overview of the Tesseract OCR Engine", Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2, 2007 Publication	1%
4	"ICCCE 2018", Springer Science and Business Media LLC, 2019 Publication	1%
5	Submitted to Troy University Student Paper	1%
6	arxiv.org Internet Source	1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 17/07/2020

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

Name: Rishabh Malhotra Department: CSE Enrolment No 161263

Contact No. 8894027019 E-mail. rishabh98.juit@gmail.com

Name of the Supervisor: Dr. Pradeep Kumar Gupta

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

LENSED IT

_____

UNDERTAKING

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

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 52
- Total No. of Preliminary pages = 7
- Total No. of pages accommodate bibliography/references = 1

Rishabh

(Signature of Student)

FOR DEPARTMENT USE

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

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

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

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

Checked by
Name & Signature

Librarian

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