VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Machhe, Belagavi, Karnataka 590018

PROJECT REPORT

on

"TEXT EXTRACTION FROM IMAGES USING CONVOLUTIONAL NEURAL NETWORK"

Submitted in partial fulfillment of the requirement
for the award of the degree of

Bachelor of Engineering
in
Information Science and Engineering
by

Sudarshan Rao M	**(1BG15IS062)**
Bharati V	**(1BG15IS063)**
Aditi J	**(1BG15IS064)**
S G Aditya Bharadwaj	**(1BG15IS066)**

Under the Guidance of
Mrs.S.Srividhya
Assistant Professor
Dept. of Information Science and Engineering

Vidyaya Amrutham Ashnuthe

B.N.M. Institute of Technology

Approved by AICTE, Affiliated to VTU, Accredited as grade A Institution by NAAC.
All UG branches – CSE, ECE, EEE, ISE & Mech.E accredited by NBA for academic years 2018-19 to 2020-21 & valid upto 30.06.2021

Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph: 91-80- 26711780/81/82 Email: principal@bnmit.in, www. bnmit.org

Department of Information Science and Engineering
2018 – 2019

B.N.M. Institute of Technology

Approved by AICTE, Affiliated to VTU, Accredited as grade A Institution by NAAC.
All UG branches – CSE, ECE, EEE, ISE & Mech.E accredited by NBA for academic years 2018-19 to 2020-21 & valid upto 30.06.2021
Post box no. 7087, 27th cross, 12th Main, Banashankari 2nd Stage, Bengaluru- 560070, INDIA
Ph: 91-80- 26711780/81/82 Email: principal@bnmit.in, www. bnmit.org

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Vidyaya Amrutham Ashnuthe

CERTIFICATE

Certified that the project work entitled **"Text Extraction from Images Using Convolutional Neural Network"** is carried out by **Mr. Sudarshan Rao M** USN **1BG15IS062**, **Ms. Bharati V** USN **1BG15IS063**, **Ms. Aditi J** USN **1BG15IS064** and **Mr. S G Aditya Bharadwaj** USN **1BG15IS066** the bonafide students of **B.N.M Institute of Technology** in partial fulfillment for the award of **Bachelor of Engineering** in **Information Science & Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2018-2019. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Mrs.S.Srividhya	**Dr. Girisha G.S**	**Dr. Krishnamurthy**
Assistant Professor,	**Prof. & Head,**	**Principal**
Dept. of ISE	**Dept. of ISE**	**BNMIT**
BNMIT	**BNMIT**	

Name of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

We consider it a privilege to express through the pages of this report, a few words of gratitude to all those distinguished personalities who guided and inspired us in completion of project.

We would like to thank **Shri. Narayan Rao R. Maanay,** Secretary, BNMEI, Bengaluru for providing excellent academic environment in the college.

We would like to sincerely thank **Prof. T. J. Rama Murthy**, Director, BNMIT, Bengaluru, for having extended his support and encouraging me during the course of the work.

We would like to express our gratitude to **Prof. Eishwar N Maanay,** Dean Administration, BNMIT, Bengaluru, for his relentless support, guidance and assistance.

We would like to thank **Dr. Krishnamurthy G.N**, Principal, BNMIT, Bengaluru, for his constant encouragement.

We would like to thank **Dr. Girisha G.S**, Professor and Head of the Department of Information Science and Engineering, BNMIT, Bengaluru, who has shared his opinions and thoughts which helped us in the completion of project successfully.

We consider it is a privilege to express gratitude to my internal guide **Mrs.Srividhya.S**, Assistant Professor, Department of Information Science and Engineering, BNMIT, Bengaluru, who has given us all the support and guidance in completing the project work successfully.

We would also like to thank project coordinator **Mr. Manjunath G S,** Assistant Professor, Department of Information Science and Engineering, BNMIT, Bengaluru, for guiding in a systematic manner.

We would like to thank our parents for their support and guidance in completing the project work successfully. We also would like to place on the card for their constant support and guidance throughout our career.

Sudarshan Rao M (1BG15IS062)
Bharati V (1BG15IS063)
Aditi J (1BG15IS064)
S G Aditya Bharadwaj (1BG15IS066)

ABSTRACT

Today most of the valuable information is present on bills, photos, sign boards. People employed in data entry department face a challenge of entering the bill details in the computer by manually typing the content of the bill. This system extracts text from the bill when the image of the bill is given. And the extracted text is directly produced in a document form, where the employer can directly access. Text fonts similar to Times new roman such as Calibri, Cambria, Arial can be extracted with hundred percent accuracy. Other fonts can be extracted with 97% accuracy. This system also has a feature of searching for a text in the image content. User can search for the keywords, if the keyword is present in the image, it gets highlighted in the extracted text. Multiple occurrences of the same keyword are also highlighted. Ultimately, this system extracts text from different types images and saves the extracted text in a document form, where it can be edited.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

Today, most of the information is available either on paper or in the form of photographs or videos. Large information is stored in images. Any text appearing in an image can provide useful information for the task of automatic image annotation and other related problems. One of the primary ways through which people share and consume information in social networks such as Facebook is through visual media such as photos and videos. In the last several years, the volume of photos being uploaded to social media platforms has grown exponentially to the order of hundreds of millions every day, presenting technological challenges for processing increasing volumes of visual information. One of the challenges in image understanding is related to the retrieval of textual information from images, also called Optical Character Recognition (OCR), which represents a process of conversion of electronic images containing typed, printed, painted or scene text into machine encoded text. Obtaining such textual information from images is important as it facilitates many different applications, e.g. search and recommendation of images.

Following state-of-the-art systems, OCR system is divided into a text detection stage and a text recognition stage. In the text detection approach, based on Faster-RCNN model, is responsible of detecting regions of the image that contain text. After that, in text recognition approach, that uses a fully-convolutional character-based recognition model, processes the detected locations and recognizes the text they contain. Text detection is an important preliminary step before text can be recognized in unconstrained image environments. The approach is based on convolutional neural networks to detect and localize horizontal text lines from raw color pixels. The network learns to extract and combine its own set of features through learning instead of using hand-crafted ones. Learning was also used in order to precisely localize the text lines by simply training the network to reject badly-cut text and without any use of tedious knowledge-based postprocessing.

Images are divided into three types:

- Document image
- Scene image
- Born-digital image

Document images: Document images are the image-format of the document. It is a way of transforming paper-based documents into image format for electric read. Document images are the focus of the text extraction in its early stage.

Scene images: Scene images contain the text, such as the advertising boards, banners, which is captured naturally when the scene images are taken by the camera, therefore scene text is embedded in the background as a part of the scene.

Born digital images: It is a digital image that never physically existed before becoming a digital file. The most common example is an image created with digital camera Born-digital images is generated by computer software and are saved as digital images.

1.1 Motivation

- Today the most information is available either on paper or in the form of photographs. Large information is stored in images. The current technology is restricted to extracting text against clean backgrounds. Thus, there is a need for a system to extract text from general backgrounds.
- Text Extraction and recognition in Images has become a potential application in many fields like Image indexing, Robotics, Intelligent transport systems etc.
- For example capturing license plate information through a video camera and extracting license number in traffic signals
- However, variations of text due to differences in size, style, orientation, and alignment, as well as low image contrast and complex background make the problem of automatic text extraction extremely challenging.

1.2 Problem Statement

Extracting text from images more efficiently using neural networks. Storing the extracted text in document format. Implementing the search technique to identify specific content from the extracted text.

This project extracts text from more than a billion public Facebook and Instagram images and video frames (in a wide variety of languages), daily and in real time, and inputs it into a text recognition model that has been trained on classifiers to understand the context of the text and the image together.

1.3 Objectives

- To get an accurate and relevant search results when one searches for an image.
- To enable people in the industry, to directly get the text this is on a bill rather than typing it out manually.

1.4 Summary

Extracting text and realizing it on a document using the state-of-the-art algorithms

Such as Convolutional neural networks and the techniques that follow it. It is going to be very helpful for those who are in data entry environment who can get the job done of some photos of the bills and invoices. Using the current technology to solve such problems in the real world with machine possible solutions is one of the greatest goals of the project.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

The literature survey contains a brief background to the existing text extraction system using various techniques, limitations of the existing systems, the proposed system and its advantage.

2.1 Background

Workers in various fields such as pharmacy and accounts work on large number of files which is tedious regarding fetching relevant information from the entire document or prescription bills. This leads to more man power.

There are platforms that do only text extraction but do not have any other features that can help the user to manipulate them.

Therefore the proposed system is an application which counters all the disadvantages of the existing platforms and enhances them.

2.2 Existing System and its limitations

Fedor Borisyuk et.al.,[1] had proposed a text extraction system called Rosetta, Facebook's scalable OCR system. It is implemented and deployed in production and that powers downstream applications within Facebook Approach, that uses a fully-convolutional character based recognition model, processes the detected locations and recognizes the text they contain. Faster-RCNN simultaneously performs detection and recognition by learning a fully-convolutional CNN that can represent an image as a convolutional feature map. Rosetta service is deployed within Facebook at scale, offers a cloud API for text extraction from images and processes a large volume of images uploaded to Facebook every day.

Manolis Delakis et.al.,[2] had proposed a text detection system using convolution neural network. The use of convolutional neural networks, for text detection directly from raw color pixels. In this advanced MLP architecture, feature extraction and classification are performed jointly in one step. The relative features and their combinations are learned by examples via the back-propagation algorithm. In addition to the automatic feature extraction, this also put emphasis on the good localization of the text lines by the network itself, instead of applying a set of tedious geometric constraints and local image processing. This was achieved by training the network to reject badly localized text. Even though it was used

computer-generated examples in the training corpus, the present results in real-world settings and compare with other methods. Before feeding the network, the image is decomposed to its three-color channels, forming the actual network input, which is three 64×36 intensity image planes. Their pixel values are linearly scaled between -1 and +1.

Andreas Veit et.al., [3] had proposed a dataset called COCO. This paper describes the COCO-Text dataset. In recent years large-scale datasets like SUN and Imagenet drove the advancement of scene understanding and object recognition. The goal of COCO-Text is to advance state-of-the-art in text detection and recognition in natural images. The dataset is based on the MS COCO dataset, which contains images of complex everyday scenes. To reflect the diversity of text in natural scenes, annotate text with (a) location in terms of a bounding box, (b) fine-grained classification into machine printed text and handwritten text, (c) classification into legible and illegible text, (d) script of the text and (e) transcriptions of legible text.

The first task for annotating the dataset is detecting text regions present in each image. Annotate each text region with an enclosing bounding box. For legible text the is aim is to have one bounding box per word, i.e. an uninterrupted sequence of characters separated by a space, and for illegible text the aim is one bounding box per continuous text region, e.g. a sheet or paper. The text detection step has four parts: Incorporating Photo OCR input. First, use the input provided by collaborating photo OCR approaches. In particular, use the detection results. Treat each detection as if it were a human annotator. False positive detections are handled in a subsequent stage after collecting the human detections. To reduce bias towards specific approaches, use only OCR input, where at least one human annotator agrees with the OCR input. This step contributes about 20% of our text annotations.

Next, classify detected text regions according to three attributes: First, legibility in terms of legible and illegible text. Legibility can be understood as indicator whether text can be read. Second, collect the script of the text. Collect the transcriptions. The collection has three iterations and each consists of two steps. First, collect transcriptions and second, check with majority vote whether they are correct. In the first iteration take transcriptions provided by the OCRs and ask workers to check for correctness. In the second and third iteration human annotators transcribe and check the text regions.

Shaoqing Ren et.al., [4] presented a project where, recent advances in object detection are driven by the success of region proposal methods and region-based convolutional neural networks (RCNNs). Although region-based CNNs were computationally expensive as originally developed, their cost has been drastically reduced thanks to sharing convolutions

across proposals. Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search, one of the most popular methods, greedily merges super pixels based on engineered low-level features. The algorithm changes-computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation.

The R-CNN method trains CNNs end-to-end to classify the proposal regions into object categories or background. R-CNN mainly plays as a classifier, and it does not predict object bounds. The fully-connected layer is then turned into a convolutional layer for detecting multiple class specific objects. The MultiBox methods generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the "singlebox" fashion of Over Feat. These class-agnostic boxes are used as proposals for R-CNN.

In detection system, called Faster R-CNN, is composed of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector. A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score. Model this process with a fully convolutional network, which is described in this section. The ultimate goal is to share computation with a Fast R-CNN object detection network, assume that both nets share a common set of convolutional layers. To generate region proposals, slide a small network over the convolutional feature map output by the last shared convolutional layer.

Sliding window-based methods proposed by Uma B. Karanje et.al.,[5] also known as region-based methods. This method uses a sliding window to search for possible texts in the image and then use machine learning techniques to identify the text. Disadvantages: These methods are slow, because the image has to be processed in multiple scales. These methods limit the search of text to a subset of image rectangles. So, it reduces the number of subsets checked for the presence of text. Connected component (CC) based methods, extracts the character candidates from an image by connected component analysis, followed by grouping character candidates into text; additional checks may be performed to remove false positives. Advantages: Achieves state of the art result. The complexity does not depend on the properties of text like orientation, font style etc. Disadvantages: Fails in some natural scene images which have very poor contrast text and strong illumination. Maximally Stable Extremal Region (MSER) is one of recent method used for text detection from natural scene

images. It gives advantages over other region detectors like Harrisaffine, Hessian-affine, edge-based regions, intensity extrema, and salient regions. MSER detects near about 2600 regions for image and gives greater results in region size, blur, viewpoint change, scale change and light change. MSER based methods are categorized under connected component based methods.

Shailendra Singh Kathait et.al.,[6] proposed that pre-processing of image involves removal of noise from image. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing. With ICR technology the text is directly entered to database post classifying all the segments in whole document after doing proper character recognition through OCR. ICR operates by capturing hand written text from image files and converting them into text searchable files there by giving users the ability to search through the files with text strings and capture information from them by using the copy/paste function. Pre-processing of image involves removal of noise from image. It is a common name for operations with images at the lowest level of abstraction, both input and output of which are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing. Different categories of image preprocessing methods exist according to the size of the pixel neighborhood that is used for the calculation of new pixel brightness: pixel brightness transformations, geometric transformations, pre-processing methods that use a local neighborhood of the processed pixel, and image restoration that requires knowledge about the entire image.

2.3 Methodologies

- **Text Detection model:**

Perform text extraction on an image in two independent steps: detection and recognition. In the first step, detect rectangular regions that potentially contain text. In the second step, perform text recognition, where, for each of the detected regions, use a convolutional neural network (CNN) to recognize and transcribe the word in the region.

Faster R-CNN has two networks: region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects. The main different here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN

shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects.
Detectron is Facebook AI Research's (FAIR) software system that implements state-of-the-art object detection algorithms, including Mask R-CNN. It is written in Python and powered by the Caffe2 deep learning framework.

At FAIR, Detectron has enabled numerous research projects, including: Feature Pyramid Networks for Object Detection, Mask R-CNN, Detecting and Recognizing Human-Object Interactions, Focal Loss for Dense Object Detection, Non-local Neural Networks, Learning to Segment Every Thing, and Data Distillation: Towards Omni-Supervised Learning. The goal of Detectron is to provide a high-quality, high-performance codebase for object detection research. It is designed to be flexible in order to support rapid implementation and evaluation of novel research.

Region Proposal Network

The output of a region proposal network (RPN) is a bunch of boxes/proposals that will be examined by a classifier and regressor to eventually check the occurrence of objects.

ROI Pooling

After RPN, proposed regions with different sizes are obtained. Different sized regions mean different sized CNN feature maps. It's not easy to make an efficient structure to work on features with different sizes. Region of Interest Pooling can simplify the problem by reducing the feature maps into the same size. Unlike Max-Pooling which has a fix size, ROI Pooling splits the input feature map into a fixed number (let's say k) of roughly equal regions, and then apply Max-Pooling on every region. Therefore the output of ROI Pooling is always k regardless the size of input.

- ## Text Recognition model:

The text recognition model is a CNN based on the ResNet18 architecture, as this architecture led to good accuracies while still being computationally efficient. To train the model, cast it as a sequence prediction problem, where the input is the image containing the text to be recognized and the output is the sequence of characters in the word image. Use the connectionist temporal classification **(CTC)** loss to train sequence model. Casting the

problem as one of sequence prediction allows the system to recognize words of arbitrary length and to recognize out-of-vocabulary words (i.e., words that weren't seen during training).

Preserving the spatial location of the characters in the image may not matter for other image classification problems, but it is very important for word recognition. Because of this, have two modifications to the ResNet18 architecture:

- Remove the global average pooling layer at the end of the model and replace the fully connected layer with a convolutional layer that can accept inputs of different lengths.
- Reduce the stride of the last convolutional layers to better preserve the spatial resolution of the features.

Both changes help obtain good accuracies. Furthermore, use long short-term memory (LSTM) units to further improve the accuracy of our models

- **Search Model :**

  There are two components:

  1. Identifying the keywords
  2. Highlighting them on the page using JavaScript.

Take the form input (e.g. "search engine keywords") and convert it into a JavaScript regular expression (e.g. "b(search| engine| keywords)\b/"). This regular expression will match any of the entered words where they appear in the content area of the page.

2.4 Summary

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous researches. Shallow and Deep Learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. Deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

CHAPTER 3

SYSTEM REQUIREMENTS & SPECIFICATIONS

CHAPTER 3

SYSTEM REQUIREMENT AND SPECIFICATION

The following section provides an overview of requirements necessary for the project. The requirements are divided as user requirement, hardware requirement, software requirement, Functional requirement, Non-functional requirement and operational requirement.

3.1 Software Requirements

- Windows 10
- Anaconda Navigator
- Python 3.0
- Flask

3.2 Hardware Requirements

- Intel i5 7^{th} gen and above
- 8 GB RAM
- GPU RTX 2080

3.3 Functional requirement

The functional requirements indicate the functionalities that the proposed system exhibit. Those include:

- Taking the desired text image
- Recognition of the text
- Search the text on the image
- Highlight the text after the search.

3.4 Non-functional requirement

- **Performance:** Response time of the proposed system should be fast enough for the user to get better experience which should be achieved by rigorous training of the model.
- **Scalability:** Should be scaled to minimum 5 fonts.

- **Capacity:** Capacity indicates the amount or size of input data the proposed system can process without much deterioration in performance.
- **Usability:** The application shall be used friendly and doesn't require any guidance to be used. In other words, the application has to be as simple as possible, so its users shall use it easily.
- **Reliability:** The application should not have any unexpected failure. In order to avoid any failure's occurrence, the specifications have been respected and followed correctly. The only problem that may occur in some cases is that the application do not get 100% of the characters in the picture.

Summary

This chapter contains the functional and non-functional requirements, Software and Hardware Requirements that are needed for efficient working of the proposed system.

CHAPTER 4

COST ESTIMATION OF THE PROJECT

CHAPTER 4

COST ESTIMATION OF THE PROJECT

4.1 Description of COCOMO model

COCOMO (Constructive Cost Estimation Model) was proposed by Boehm [1981]. According to Boehm, software cost estimation should be done through three stages. Basic COCOMO, intermediate COCOMO, and complete COCOMO.

The basic COCOMO model gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort applied (E) = a_b(KLOC)b_b[man-month]

Development Time (D) = c_b(Effort Applied) d_b[months]

People required (P) = Effort Applied/Development Time [count]

- KLOC is the estimated size of the software product expressed in kilo lines of code.
- a1, a2, b1, b2 are constants for each category of software products.
- D is the estimated time to develop the software, expressed in months.
- Effort is the total effort required to develop the software product, expressed in person months (PMs).

Boehm postulated that any software development project can be classified into one of the following three categories based on the development complexity:

- Organic: A development project can be considered of organic type, if the project deals with developing a well understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.
- Semi-detached projects: A developing project can be considered of semidetached type, if the development consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.
- Embedded projects: A development project is considered to be of embedded type, if the software being developed is strongly coupled to complex hardware, or if the strongest regulations on the operational procedures exist.

Table 4.1: Coefficient for different categories of software projects

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

4.2 Cost Estimation

- The type of project that is being implemented is a semi-detached project.
- The value of the co-efficient are: a=3.0, b=1, c=2.5 and d=0.35 as shown in table 4.1. The value is KLOC=6.
- Effort applied (E) is found to be 22.31 man months.
- Development time is found to be 7.4 months.
- Minimum number of people required is found to be 4 people.
- The effort applied and development time as shown in the circulation above is in compliance with the actual effort applied and development time taken.

CHAPTER 5

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

5.1 Architectural design

Requirements of the software should be transformed into an architectural that describes the software's top-level structure and identifies its components. This is accomplished through architectural design (also called system design), which acts as a preliminary blueprint from which software can be developed. IEEE architectural design as the process of defining a collection of hardware and software components and their interface to establish the framework for the development of a computer system .This framework is established by examining the software requirement document and designing a model for providing implementation details .These details are used to specify the components of the system along with their inputs, outputs, functions, and the interaction between them. An architectural design performs several functions.

Figure 4.1 shows the system architecture diagram with different components of the proposed system .Feature such as convolutional neural network is used to pre-process the image and store is as a text document which is in an editable format.

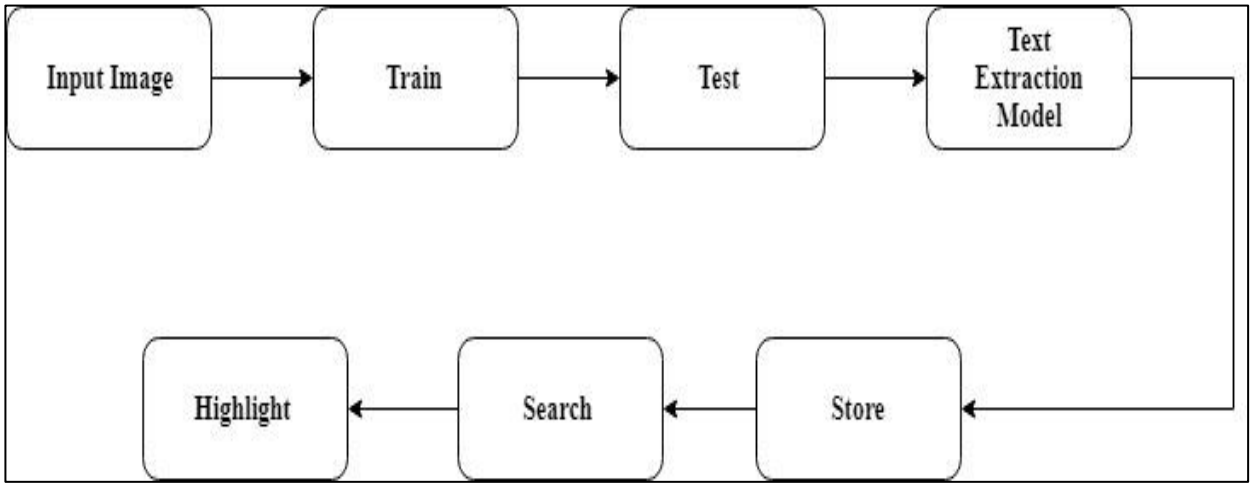


Fig 5.1: Architectural Design

5.2 Data Flow Diagram:

A **data-flow diagram** (DFD) is a way of representing a flow of a data of a process or a system (usually an information system) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow; there are no decision rules and no loops. Figure 4.2 shows the data flow diagram of the proposed system where each activity is identified.

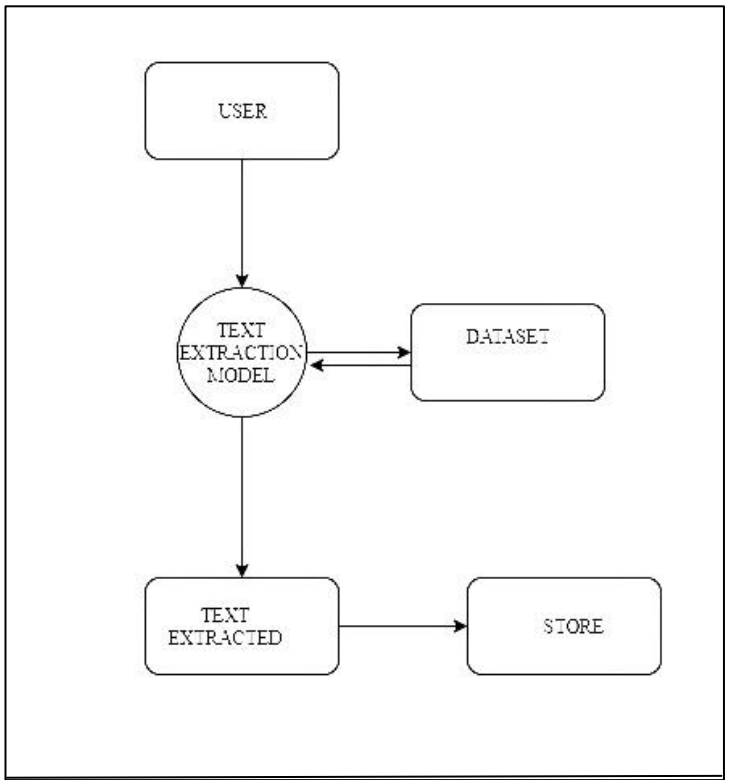


Fig 5.2: Data Flow Diagram

5.3 Modules:

5.3.1 Pre-processing of image.

5.3.2 Text detection and extraction

5.3.3 Searching for the text in the image.

5.3.1 Pre-Processing of image

The input given to the recognition engine is a scanned image in jpg/png/tif format, where scanning needs to be done at good resolution for maximum accuracy. The root step is the pre- processing of this image as shown in figure 4.3.

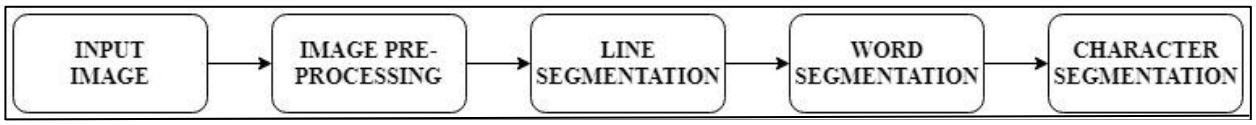


Fig 5.3: Image Processing Methodology

Pre-processing of image involves removal of noise from image. It is a common name for operations with images at the lowest level of abstraction, both input and output of which are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing. Different categories of image pre-processing methods exist according to the size of the pixel neighborhood that is used for the calculation of new pixel brightness: pixel brightness transformations, geometric transformations, pre-processing methods that use a local neighborhood of the processed pixel, and image restoration that requires knowledge about the entire image. The different pre-processing filters (median, symmetric etc.) can be used to improve the quality of the image and reduce distortion. After the pre-processing operations have been performed on the image, it is now ready for processing further.

Scaling to the Right Size

Ensure that the images are scaled to the right size which usually is of at least 300 DPI (Dots per Inch) .Keeping DPI lower than 200 will give unclear and incomprehensible results while keeping the DPI above 600 will unnecessarily increase the size of the output file without improving the quality of the file. Thus, a DPI of 300 works best for this purpose.

Increase Contrast

Low contrast can result in poor recognition. Increase the contrast and density before carrying out the detection process. This can be done in the scanning software itself or in any other image processing software. Increasing the contrast between the text/image and its background brings out more clarity in the output.

Binarize Image

This step converts a multi-coloured image (RGB) to a black and white image. There are several algorithms to convert a color image to a monochrome image, ranging from simple thresholding to more sophisticated zonal analysis. Another advantage of binarizing your images before sending them to your detection engine is the reduced size of your images.

Deskew

This may also be referred to as rotation. This means de-skewing the image to bring it in the right format and right shape. The text should appear horizontal and not tilted in any angle. If the image is skewed to any side, deskew it by rotating it clockwise or anti clockwise direction.

The figure 4.4 shows the final image which we can send to the detection engine. The image was binarized; de-skewed and scanning artifacts (black border) were removed.

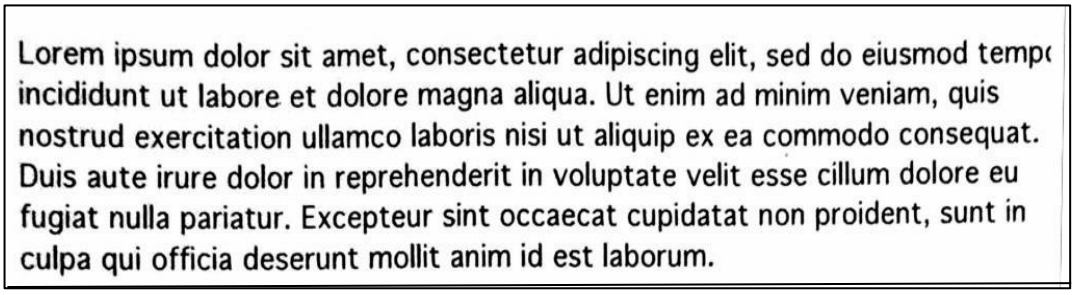


Fig 5.4: Pre-Processed Image

5.3.2 Text Detection and extraction

Text Detection model

Perform text extraction on an image in two independent steps: detection and recognition. In the first step, detect rectangular regions that potentially contain text. In the second step, perform text recognition, where, for each of the detected regions, use a convolutional neural network (CNN) to recognize and transcribe the word in the region.

For text detection, adopt an approach based on **Faster R-CNN**, a state-of-the-art object detection network. In a nutshell, Faster R-CNN simultaneously performs detection and recognition by:

1. Learning a CNN that can represent an image as a convolutional feature map.
2. Learning a region proposal network (RPN), which takes that feature map as input and produces a set of proposed regions (or bounding boxes) that are likely to contain text, together with their confidence score.
3. Extracting the features from the feature map associated with the spatial extent of each candidate box, and learning a classifier to recognize them (in our case, the categories are text and no text). The proposals are sorted by their confidence scores, and non- maximum

suppression (NMS) is used to remove duplicates or overlaps and choose the most promising proposals. Additionally, bounding box regression is typically used to improve the accuracy of the produced regions by refining the proposals.

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, a Region Proposal Network (RPN) is introduced that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection .Further merge RPN and Fast R-CNN into a single network by sharing their convolutional features---using the recently popular terminology of neural networks with 'attention' mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model, the detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image

The whole detection system (feature encoding, RPN, and classifiers) is trained jointly in a supervised, end-to-end manner. The text detection model uses Faster R-CNN but replaces the ResNet convolutional body with a ShuffleNet-based architecture for efficiency reasons. ShuffleNet is significantly faster than ResNet and showed comparable accuracy on our data sets. The anchors are modified in RPN to generate wider proposals, as text words are typically wider than the objects for which the RPN was designed. In particular, seven aspect ratios and five sizes are used, so the RPN generates 35 anchor boxes per region. To train the end-to-end detection system, bootstrap the model with an in-house synthetic data set (more on that below) and then fine-tune it with human-annotated data sets so that it learns real-world characteristics. For training, use the recently open-sourced Detectron framework powered by Caffe2.

Detectron is Facebook AI Research's (FAIR) software system that implements state-of-the- art object detection algorithms, including Mask R-CNN. It is written in Python and powered by the Caffe2 deep learning framework.

At FAIR, Detectron has enabled numerous research projects, including: Feature Pyramid Networks for Object Detection, Mask R-CNN, Detecting and Recognizing Human-Object

Interactions, Focal Loss for Dense Object Detection, Non-local Neural Networks, Learning to Segment Every Thing, and Data Distillation: Towards Omni-Supervised Learning. The goal of Detectron is to provide a high-quality, high-performance codebase for object detection research. It is designed to be flexible in order to support rapid implementation and evaluation of novel research.

Faster R-CNN has two networks: region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects. The main different here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects. Figure 4.2 explains the working of faster R-CNN.

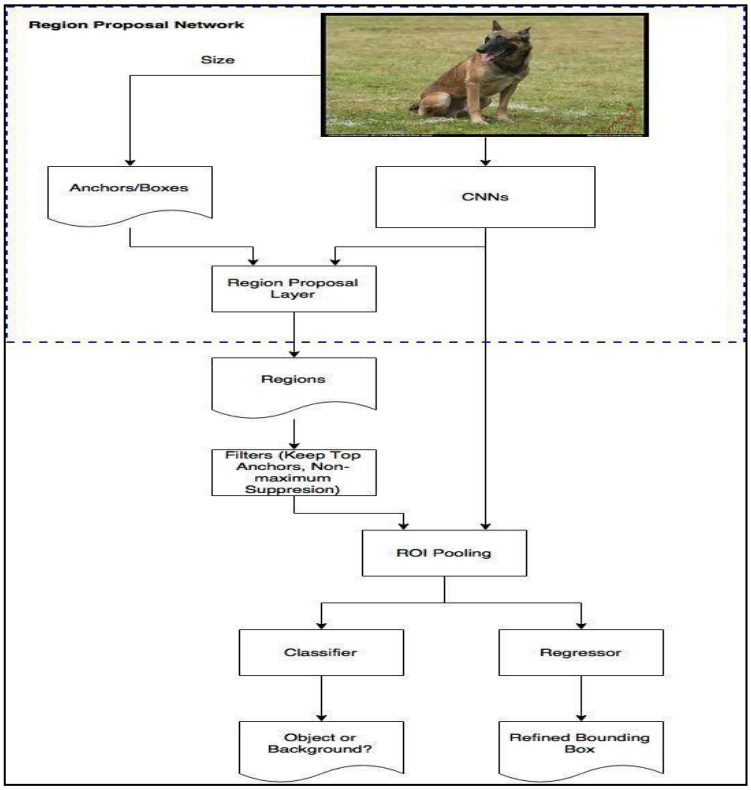


Fig 5.5: Working of R-CNN

Region Proposal Network

The output of a region proposal network (RPN) is a bunch of boxes/proposals that will be examined by a classifier and regress or to eventually check the occurrence of objects. To be more precise, RPN predicts the possibility of an anchor being background or foreground, and refine the anchor .Figure 4.3 describes the working to RPN.

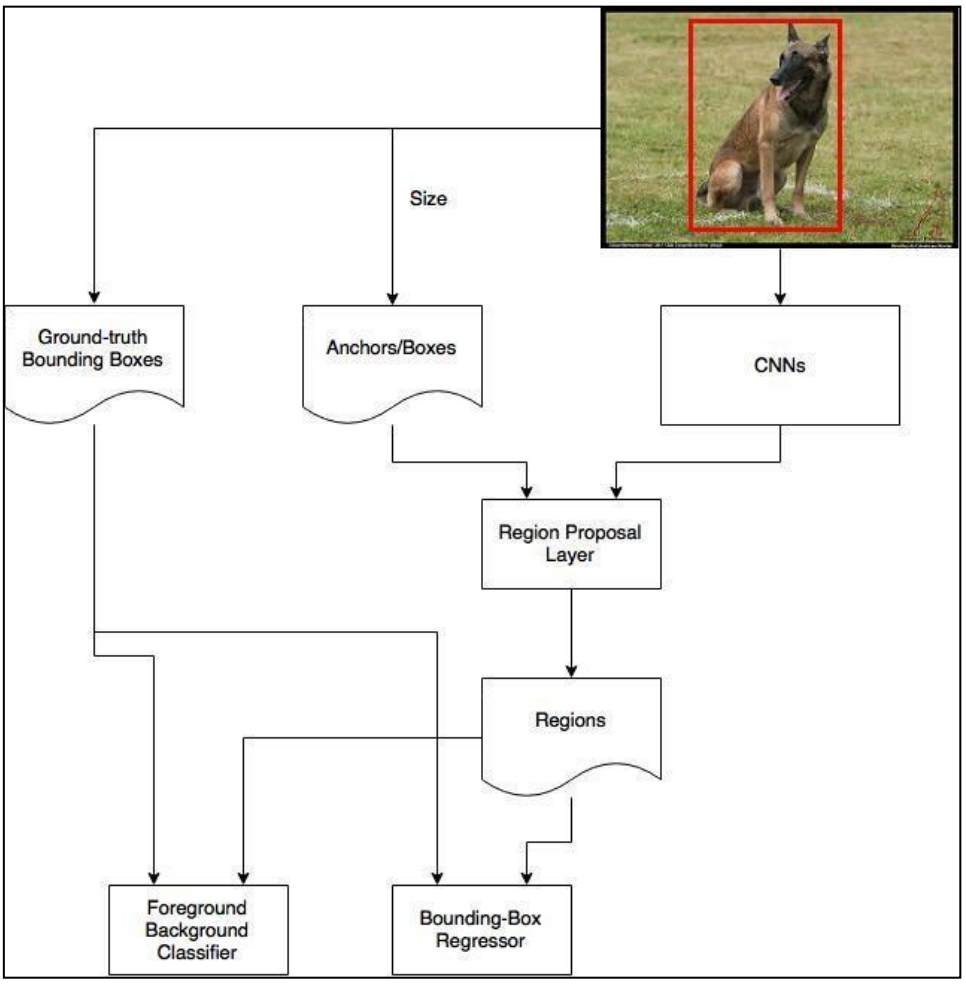


Fig 5.6: Working of region proposal network

The Classifier of Background and Foreground

The first step of training a classifier is to make a training dataset. The training data is the anchors that get from the above process and the ground-truth boxes. The problem to be solved here is how to use the ground-truth boxes to label the anchors. The basic idea here is to label the anchors having the higher overlaps with ground-truth boxes as foreground, the ones with lower overlaps as background. Apparently, it needs some tweaks and compromise separate foreground and background.

The second question here is what features of the anchors are: Let's say the 600x800 image shrinks 16 times to a 39x51 feature map after applying CNNs. Every position in the feature map has 9 anchors, and every anchor has two possible labels (background, foreground). If the depth of the feature map is 18 (9 anchors x 2 labels), every anchor will have a vector with two values (normal called logit) representing foreground and background. If the logit is fed into a softmax/logistic regression activation function, it will predict the labels. Now the the training data is complete with features and labels.

In the architecture of Overfeat, it only uses non-overlapping convolutional and pooling filters to make sure every position in the feature map cover its own receptive field without overlapping others. In Faster R-CNN, receptive fields of different anchors often overlap each other. It leaves the RPN to be position-aware.

The Regressor of Bounding Box

Pick out the anchors based on the similar criteria for the regressor to refine. One point here is that anchors labeled as background shouldn't include in the regression. The depth of feature map is 32 (9 anchors x 4 positions). The paper uses smooth-L1 loss on the position (x ,y) of top-left the box, and the logarithm of the heights and widths, which is as the same as in Fast R-CNN.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

ROI Pooling

After RPN, proposed regions with different sizes are obtained .Different sized regions means different sized CNN feature maps. It's not easy to make an efficient structure to work on features with different sizes. Region of Interest Pooling can simplify the problem by reducing the feature maps into the same size. Unlike Max-Pooling which has a fix size, ROI Pooling splits the input feature map into a fixed number (let's say k) of roughly equal regions, and then apply Max-Pooling on every region. Therefore the output of ROI Pooling is always k regardless the size of input. Here is a good explanation about ROI Pooling.

Text Recognition model

The text recognition model is a CNN based on the ResNet18 architecture, as this architecture led to good accuracies while still being computationally efficient. To train the model, cast it as a sequence prediction problem, where the input is the image containing the text to be recognized and the output is the sequence of characters in the word image .Use the **connectionist temporal classification (CTC)** loss to train our sequence model. Casting the problem as one of sequence prediction allows the system to recognize words of arbitrary length and to recognize out-of-vocabulary words (i.e., words that weren't seen during

training).

Preserving the spatial location of the characters in the image may not matter for other image classification problems, but it is very important for word recognition. Because of this, have two modifications to the ResNet18 architecture:

- Remove the global average pooling layer at the end of the model and replace the fully connected layer with a convolutional layer that can accept inputs of different lengths.
- Reduce the stride of the last convolutional layers to better preserve the spatial resolution of the features.

Both changes help obtain good accuracies. Furthermore, use long short-term memory (LSTM) units to further improve the accuracy of our models. Figure 4.4 describes the working of text recognition.

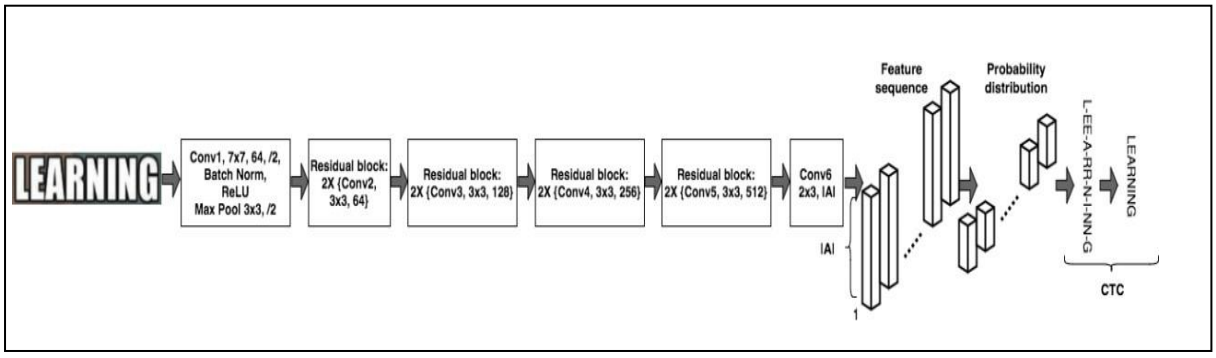


Fig 5.7: Text Recognition Model

Curriculum Learning for CTC text recognition model:

Training models with a sequence loss such as CTC is notably more difficult than training them with standard classification losses. For example, with a long word, one needs to predict all the characters of the word and in the right order, which is intuitively more difficult than just predicting a single label for an image. In practice, it is observed that low learning rates led to under fit models, while higher learning rates led to model divergence. To address this problem, draw inspiration from curriculum learning, where one first trains a model with a simple(r) task and increases the difficulty of the problem as training progresses. As a result, training procedure is modified in two ways:

- Start training the model using only short words, with up to five characters. Once all the five-or-fewer-character words are seen, start training with words of six or fewer characters, then seven or fewer, etc. This significantly simplifies the problem.

- Use a learning rate scheduling, starting with a very low learning rate to ensure that the model doesn't diverge, and progressively increase the learning rate during the first few epochs to ensure that the model reaches a good, stable point. Once this point is reached, start reducing the learning rate, as is standard practice when learning deep models.

5.4 Algorithms:

Algorithm for text recognition model:

Algorithm 1 Curriculum Learning for Text Recognition Model

Input: Training dataset D; Number of warm-up epochs W; Number of epochs N; Initial maximum length of words l_0; Warm up word width w_w; Initial word width w_0; Learning rates α and β; Learning rate decay period t

Output: Trained CTC model.

Note: The values of learning rates α and β are determined empirically by observing training convergence. α is the highest learning rate that allows one to train the model after random initialization without diverging, while β is the highest learning rate that allows one to train the model after initializing with pretrained weights.
Set initial learning rate $lr = \alpha$.
Set log-increment $\Delta = \frac{\log\beta - \log\alpha}{N}$.
Set image width preprocessing to w_w
Warmup training:
Set $l = l_0$.
for $(i = 1; i \leq W; i++, l++)$ **do**
    Generate filtered training dataset T by keeping only training examples from D with labeled word length equal or smaller than l characters.
    Train CTC model for one epoch using filtered training dataset T and learning rate lr.
    Increase learning rate $lr = \alpha + 10^{i\Delta}$
end for
Post-warmup training:
Set $w = w_0$
for $(i = 1; i \leq N; i++, w+ = 8)$ **do**
    Set image width preprocessing to w. Generate filtered training dataset T by keeping only training examples from D with labeled word length equal or smaller than the size of the feature map when using width w.
    Train CTC model for one epoch using filtered training dataset T and learning rate lr.
    Set $lr = lr * 10^{-floor(i/t)}$
end for
Return model.

Convolutional Neural Network

Convolutional neural networks are deep artificial neural networks that are used primarily to classify images, cluster them by similarity, and perform object recognition within scenes. Convolutional networks perform optical character recognition (OCR) to digitize text and make natural-language processing possible on analog and hand-written documents, where the images are symbols to be transcribed

Convolutional networks perceive images as volumes; i.e. three-dimensional objects, rather than flat canvases to be measured only by width and height. That's because digital colour images have a red-blue-green (RGB) encoding, mixing those three colours to produce the colour spectrum humans perceive. A convolutional network ingests such images as three

separate strata of colour stacked one on top of the other as shown is figure 4.5.

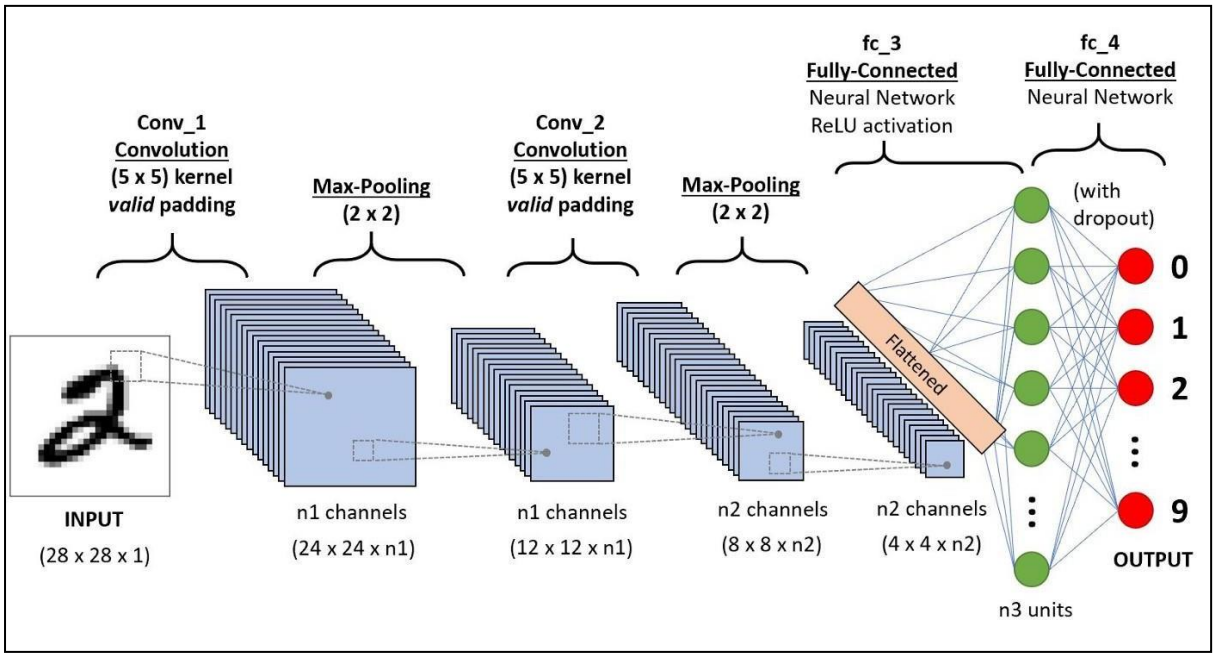


Fig 5.8: Convolutional Neural Network

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

Input Image

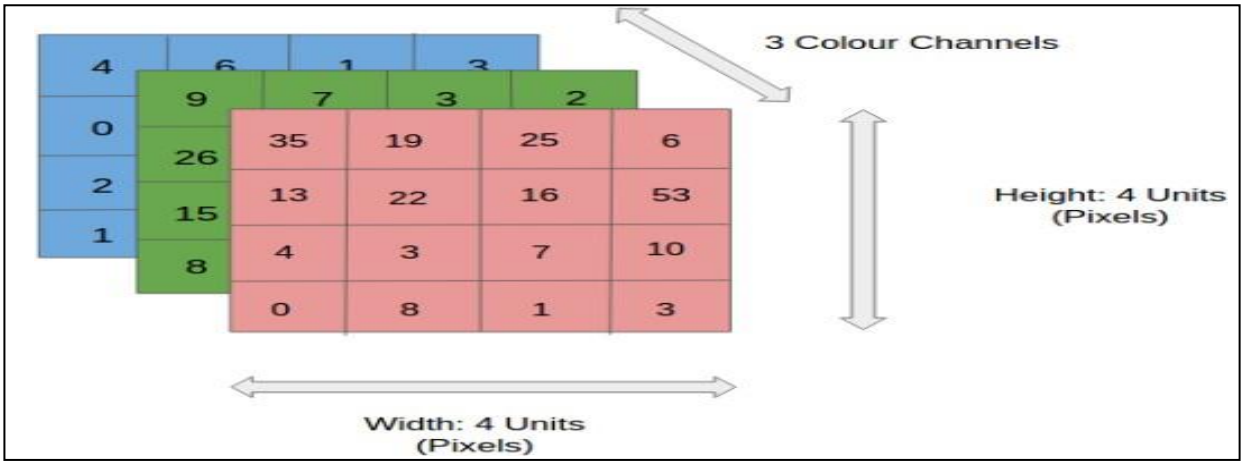


Fig 5.9: RGB image

In the figure 4.6, there is RGB image, which has been separated by its three colour planes—Red, Green, and Blue. There are a number of such colour spaces in which images exist—Grayscale, RGB, HSV, CMYK, etc. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good

prediction. This is important to design an architecture which is not only good at learning features but also is scalable to massive datasets.

Convolution Layer—the Kernel

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

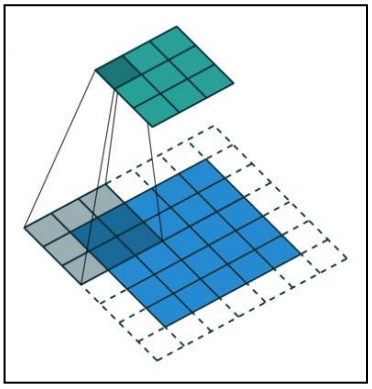


Fig 5.10: Convolution Operation with Stride Length = 2

Pooling Layer

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

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.as shown in Fig. 4.9

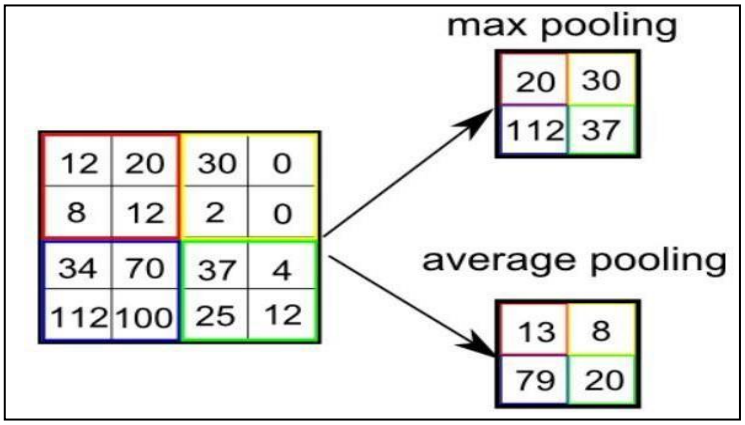


Fig 5.11: Pooling Layer

Classification—Fully Connected Layer (FC Layer)

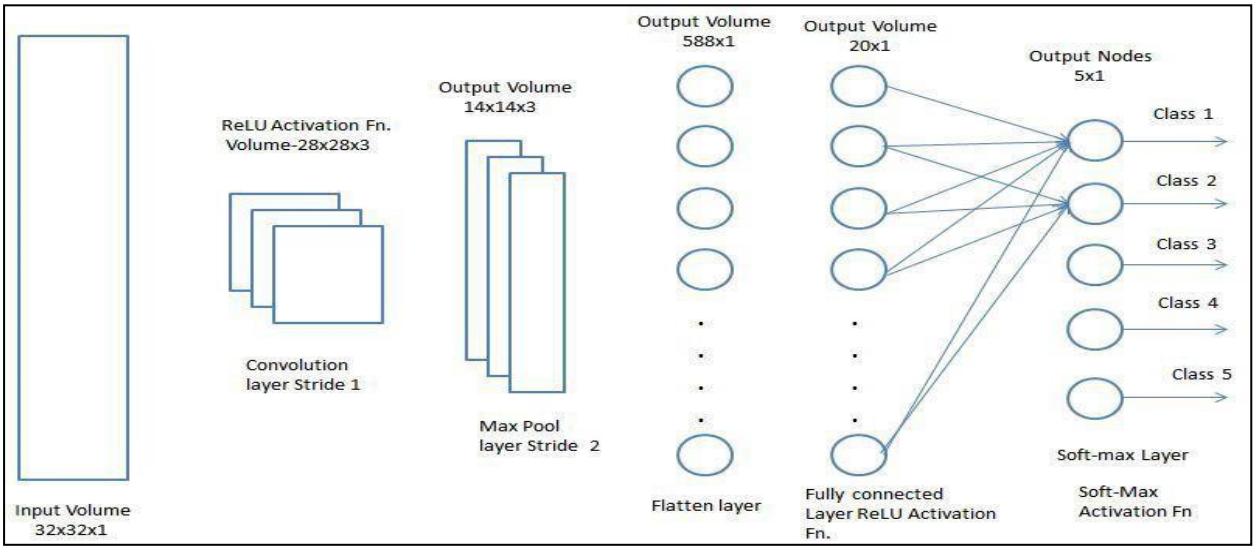


Fig 5.12: Classification—Fully Connected Layer

Adding a Fully-Connected layer is a cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space as shown in fig 4.10.

Flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and back propagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

Searching for text

Pattern searching is an important problem in computer science. When searched for a string in notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

There are two components:

1. Identifying the keywords.
2. Highlighting them on the page using JavaScript.

Take the form input (e.g. “search engine keywords”) and convert it into a JavaScript regular expression (e.g. “/\b(search| engine| keywords)\b”). This regular expression will match any of the entered words where they appear in the content area of the page.

When the apply() method is called it generates a regular expression from the keywords, clears any existing highlighting on the page, and then calls the hiliteWords() method passing a reference to the selected start node. The hiliteWords() method examines the first node to see if it has any child nodes in which case it calls itself recursively once for each child. When a text node is encountered its contents are tested against the regular expression to see if any of our keywords are present. If there are one or more matches the text node is cut in two at the point where the first match was found. The matched word itself is wrapped in an MARK tag that also specifies the background colour. The process then continues. As the script encounters and highlights different keywords the colour used for each is remembered so that the same keyword can be highlighted consistently down the page.

CHAPTER 6

IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

Implementation is the phase of project where the theoretical outline is transformed into a working framework. Project implementation is the phase where visions and plans become reality. If implementation is not consciously arranged and controlled, it can bring about confusion and mystification.

6.1 List Of Modules

The project has been implemented by dividing the entire project into four modules. They are:

- Preprocessing of image
- Text detection and extraction
- Searching for text in the image

6.2 Module Description

The section below describes each of the modules

6.2.1 Preprocessing of image

The code below demonstrates how the image is being preprocessed using

Thresh filter:

```
if  preprocess == "thresh": gray = cv2.threshold(gray, 0,

255,cv2.THRESH_BINARY|cv2.THRESH_OTSU)[1]
```

GausianBlur:

```
gray = cv2.medianBlur(gray, 3)
```

ScaleUp:

```
gray=cv2.resize(img, None, fx=2, fy=2,interpolation=cv2.INTER_LINEAR)
```

Deskew:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.bitwise_not(gray)

# threshold the image, setting all foreground pixels to
# 255 and all background pixels to 0
```

```
 thresh = cv2.threshold(gray, 0, 255,
 cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

 #grab the (x, y) coordinates of all pixel values that
 # are greater than zero, then use these coordinates to
 # compute a rotated bounding box that contains all
 # coordinates
 coords = np.column_stack(np.where(thresh > 0))
 angle = cv2.minAreaRect(coords)[-1]

 # the `cv2.minAreaRect` function returns values in the
 # range [-90, 0); as the rectangle rotates clockwise the
 # returned angle trends to 0 -- in this special case we
 # need to add 90 degrees to the angle
 if angle < -45:
                angle = -(90 + angle)

# otherwise, just take the inverse of the angle to make
# it positive
else:
                angle = -angle

# rotate the image to deskew it
(h, w) = image.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, angle, 1.0)
rotated = cv2.warpAffine(image, M, (w, h),
flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)

# draw the correction angle on the image so we can validate it
cv2.putText(rotated, "Angle: {:.2f} degrees".format(angle),
(10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
 # show the output image
 print("[INFO] angle: {:.3f}".format(angle))
 cv2.imshow("Input", image)
```

```
cv2.imshow("Rotated", rotated)
cv2.waitKey(0)
```

6.2.2 Text detection and extraction

The text detection and extraction from images are done using CNN which has the following code:

```
def setupCNN(self):
        "create CNN layers and return output of these layers"
        cnnIn4d = tf.expand_dims(input=self.inputImgs, axis=3)
# list of parameters for the layers
 kernelVals = [5, 5, 3, 3, 3]
 featureVals = [1, 32, 64, 128, 128, 256]
 strideVals = poolVals = [(2,2), (2,2), (1,2), (1,2), (1,2)]
 numLayers = len(strideVals)
# create layers
 pool = cnnIn4d # input to first CNN layer
 for i in range(numLayers):
      kernel = tf.Variable(tf.truncated_normal([kernelVals[i], kernelVals[i],
     featureVals[i], featureVals[i + 1]], stddev=0.1))
 conv = tf.nn.conv2d(pool, kernel, padding='SAME',  strides=(1,1,1,1))
 conv_norm = tf.layers.batch_normalization(conv, training=self.is_train)
 relu = tf.nn.relu(conv_norm)
 pool = tf.nn.max_pool(relu, (1, poolVals[i][0], poolVals[i][1], 1), (1, strideVals[i][0],
 strideVals[i][1], 1), 'VALID') self.cnnOut4d = pool
def setupRNN(self):
        "create RNN layers and return output of these layers"
        rnnIn3d = tf.squeeze(self.cnnOut4d, axis=[2])
```

```
# basic cells which is used to build CNN

numHidden = 256

cells = [tf.contrib.rnn.LSTMCell(num_units=numHidden, state_is_tuple=True) for _ in
        range(2)] # 2 layers

    # stack basic cells

stacked = tf.contrib.rnn.MultiRNNCell(cells, state_is_tuple=True)

# bidirectional RNN

# BxTxF -> BxTx2H

 ((fw, bw), _) = tf.nn.bidirectional_dynamic_rnn(cell_fw=stacked, cell_bw=stacked,
       inputs=rnnIn3d, dtype=rnnIn3d.dtype)

 # BxTxH + BxTxH -> BxTx2H -> BxTx1X2H  concat tf.expand_dims(tf.concat([fw,
      bw], 2), 2)

 # project output to chars (including blank): BxTx1x2H -> BxTx1xC -> BxTxC

kernel = tf.Variable(tf.truncated_normal([1, 1, numHidden * 2, len(self.charList) + 1],
       stddev=0.1))

self.rnnOut3d = tf.squeeze(tf.nn.atrous_conv2d(value=concat, filters=kernel, rate=1,
       padding='SAME'), axis=[2])

def setupCTC(self):

          "create CTC loss and decoder and return them"

          # BxTxC -> TxBxC

          self.ctcIn3dTBC = tf.transpose(self.rnnOut3d, [1, 0, 2])

          # ground truth text as sparse tensor

          self.gtTexts = tf.SparseTensor(tf.placeholder(tf.int64, shape=[None, 2]) ,
tf.placeholder(tf.int32, [None]), tf.placeholder(tf.int64, [2]))

# calc loss for batch

self.seqLen = tf.placeholder(tf.int32, [None])

self.loss = tf.reduce_mean(tf.nn.ctc_loss(labels=self.gtTexts, inputs=self.ctcIn3dTBC,
```

```
sequence_length=self.seqLen,ctc_merge_repeated=True))

# calc loss for each element to compute label probability

self.savedCtcInput = tf.placeholder(tf.float32, shape=[Model.maxTextLen, None,
len(self.charList) + 1]) self.lossPerElement = tf.nn.ctc_loss(labels=self.gtTexts,
inputs=self.savedCtcInput, sequence_length=self.seqLen, ctc_merge_repeated=True)

# decoder: either best path decoding or beam search decoding

if self.decoderType == DecoderType.BestPath:

self.decoder = tf.nn.ctc_greedy_decoder(inputs=self.ctcIn3dTBC,
sequence_length=self.seqLen)

elif self.decoderType == DecoderType.BeamSearch:

self.decoder = tf.nn.ctc_beam_search_decoder(inputs=self.ctcIn3dTBC,
sequence_length=self.seqLen, beam_width=50, merge_repeated=False)

elif self.decoderType == DecoderType.WordBeamSearch:

# import compiled word beam search operation (see
https://github.com/githubharald/CTCWordBeamSearch)

word_beam_search_module = tf.load_op_library('TFWordBeamSearch.so')

# prepare information about language (dictionary, characters in dataset, characters
forming words)

chars = str().join(self.charList)

wordChars = open('../model/wordCharList.txt').read().splitlines()[0]

corpus = open('../data/corpus.txt').read()

# decode using the "Words" mode of word beam search

    self.decoder=word_beam_search_module.word_beam_search(tf.nn.softmax(self.ct
    cIn3dTBC, dim=2), 50, 'Words', 0.0, corpus.encode('utf8'), chars.encode('utf8'),
    wordChars.encode('utf8'))

def setupTF(self):

"initialize TF"

 print('Python: '+sys.version)
```

```
 print('Tensorflow: '+tf.__version__)
sess=tf.Session() # TF session
saver = tf.train.Saver(max_to_keep=1) # saver saves model to file
        modelDir = '../model/'
            latestSnapshot = tf.train.latest_checkpoint(modelDir) # is there a saved model?
# if model must be restored (for inference), there must be a snapshot
        if self.mustRestore and not latestSnapshot:
raise Exception('No saved model found in: ' + modelDir)
        # load saved model if available
        if latestSnapshot:
            print('Init with stored values from ' + latestSnapshot)
            saver.restore(sess, latestSnapshot)
        else:
            print('Init with new values')
            sess.run(tf.global_variables_initializer())
return (sess,saver)
def toSparse(self, texts):
                    "put ground truth texts into sparse tensor for ctc_loss"
                    indices = []
                    values = []
                    shape = [len(texts), 0] # last entry must be max(labelList[i])
# go over all texts
                    for (batchElement, text) in enumerate(texts):
# convert to string of label (i.e. class-ids)
                    labelStr = [self.charList.index(c) for c in text]
```

```
# sparse tensor must have size of max. label-string
                if len(labelStr) > shape[1]:
                        shape[1] = len(labelStr)
# put each label into sparse tensor
                        for (i, label) in enumerate(labelStr):
                                indices.append([batchElement, i])
                                values.append(label)
                return (indices, values, shape)
def decoderOutputToText(self, ctcOutput, batchSize):
            "extract texts from output of CTC decoder"
# contains string of labels for each batch element
                encodedLabelStrs = [[] for i in range(batchSize)]
# word beam search: label strings terminated by blank
                if self.decoderType == DecoderType.WordBeamSearch:
                        blank=len(self.charList)
                        for b in range(batchSize):
                        for label in ctcOutput[b]:
                                if label==blank:
                                        break
                                encodedLabelStrs[b].append(label)
# TF decoders: label strings are contained in sparse tensor
                        else:
# ctc returns tuple, first element is SparseTensor
                                decoded=ctcOutput[0][0]
# go over all indices and save mapping: batch -> values
```

```
                idxDict = { b : [] for b in range(batchSize) }
                for (idx, idx2d) in enumerate(decoded.indices):
                  label = decoded.values[idx]
                  batchElement = idx2d[0] # index according to [b,t]
                  encodedLabelStrs[batchElement].append(label)
# map labels to chars for all batch elements
                return [str().join([self.charList[c] for c in labelStr]) for labelStr in
                encodedLabelStrs]
def trainBatch(self, batch):
    "feed a batch into the NN to train it"
    numBatchElements = len(batch.imgs)
    sparse = self.toSparse(batch.gtTexts)
    rate = 0.01 if self.batchesTrained < 10 else (0.001 if self.batchesTrained < 10000
    else 0.0001) # decay learning rate
    evalList = [self.optimizer, self.loss]
feedDict = {self.inputImgs : batch.imgs, self.gtTexts : sparse , self.seqLen :
[Model.maxTextLen] * numBatchElements, self.learningRate : rate, self.is_train: True}
    (_, lossVal) = self.sess.run(evalList, feedDict)
    self.batchesTrained += 1
    return lossVal
def inferBatch(self, batch, calcProbability=False, probabilityOfGT=False):
        "feed a batch into the NN to recognize the texts"
# decode, optionally save RNN output
        numBatchElements = len(batch.imgs)
        evalList = [self.decoder] + ([self.ctcIn3dTBC] if calcProbability else [])
        feedDict = {self.inputImgs : batch.imgs, self.seqLen :
        [Model.maxTextLen] * numBatchElements, self.is_train: False}
```

```
                evalRes = self.sess.run([self.decoder, self.ctcIn3dTBC], feedDict)

                decoded = evalRes[0]

                texts = self.decoderOutputToText(decoded, numBatchElements)

# feed RNN output and recognized text into CTC loss to compute labeling probability

                probs = None

                if calcProbability:

                    sparse = self.toSparse(batch.gtTexts) if probabilityOfGT else
                    self.toSparse(texts)

                   ctcInput = evalRes[1]

                  evalList = self.lossPerElement

        feedDict = {self.savedCtcInput : ctcInput, self.gtTexts : sparse, self.seqLen :
 [Model.maxTextLen] * numBatchElements, self.is_train: False}

 lossVals = self.sess.run(evalList, feedDict)

 probs = np.exp(-lossVals)

 return (texts, probs)

def save(self):

 "save model to file"

 self.snapID += 1

 self.saver.save(self.sess, '../model/snapshot', global_step=self.snapID)
```

6.2.3 Searching for text from image

```
 function Hilitor(id, ag)
 {
   // private variables
   var targetNode =
 document.getElementById(id) ||
 document.body;
   var hiliteTag = tag || "MARK";
   var skipTags = new RegExp("^(?:" +
```

```
 hiliteTag + "|SCRIPT|FORM|SPAN)$");
  var colors = ["#ff6", "#a0ffff", "#9f9",
 "#f99", "#f6f"];
  var wordColor = [];
  var colorIdx = 0;
  var matchRegExp = "";
  var openLeft = false;
  var openRight = false;
 // characters to strip from start and end
of the input string
 var endRegExp = new
RegExp('^[^\\w]+|[^\\w]+$', "g");
 // characters used to break up the input
string into words
 var breakRegExp = new
RegExp('[^\\w\'-]+', "g");
 this.setEndRegExp = function(regex) {
  endRegExp = regex;
  return endRegExp;
 };
 this.setBreakRegExp = function(regex) {
  breakRegExp = regex;
  return breakRegExp;
 };
 this.setMatchType = function(type)
 {
  switch(type)
  {
    case "left":
      this.openLeft = false;
      this.openRight = true;
      break;
    case "right":
      this.openLeft = true;
                              this.openRight =
```

```
                        false;
                            break;

      case "open":
        this.openLeft = this.openRight =
true;
        break;

      default:
        this.openLeft = this.openRight =
false;
    }
  };
  this.setRegex = function(input)
  {
    input = input.replace(endRegExp, "");
    input = input.replace(breakRegExp,
"|");
    input = input.replace(/^\||\|$/g, "");
    if(input) {
      var re = "(" + input + ")";
      if(!this.openLeft) re = "\\b" + re;
      if(!this.openRight) re = re + "\\b";
      matchRegExp = new RegExp(re, "i");
       return matchRegExp;
      }
      return false;
    };
    this.getRegex = function()
    {
      var retval = matchRegExp.toString();
    retval =
retval.replace(/(^\/(\\b)?|\(|\)|(\\b)?\/i$)/g,
"");
    retval = retval.replace(/\|/g, " ");
```

```
    return retval;
  };
  // recursively apply word highlighting
  this.hiliteWords = function(node)
  {
    if(node === undefined || !node) return;
    if(!matchRegExp) return;
    if(skipTags.test(node.nodeName))
return;
    if(node.hasChildNodes()) {
      for(var i=0; i <
node.childNodes.length; i++)
       this.hiliteWords(node.childNodes[i]);
    }
    if(node.nodeType == 3) { //
NODE_TEXT
      if((nv = node.nodeValue) && (regs =
matchRegExp.exec(nv))) {

if(!wordColor[regs[0].toLowerCase()]) {
         wordColor[regs[0].toLowerCase()]
= colors[colorIdx++ % colors.length];
        }
        var match =
document.createElement(hiliteTag);

match.appendChild(document.createText
Node(regs[0]));
        match.style.backgroundColor =
wordColor[regs[0].toLowerCase()];
        match.style.color = "#000";
        var after =
node.splitText(regs.index);
        after.nodeValue =
after.nodeValue.substring(regs[0].length);
```

```
      node.parentNode.insertBefore(match,
after);
      }
    };
  };
  // remove highlighting
  this.remove = function()
  {
    var arr =
document.getElementsByTagName(hilite
Tag);
    while(arr.length && (el = arr[0])) {
      var parent = el.parentNode;
      parent.replaceChild(el.firstChild, el);
      parent.normalize();
    }
  };
  // start highlighting at target node
  this.apply = function(input)
                              {
                                this.remove();
                                if(input ===
                             undefined || !input)
                             return;
    if(this.setRegex(input)) {
      this.hiliteWords(targetNode);
    }
    return matchRegExp;
  };
}
```

CHAPTER 7

TESTING

CHAPTER 7

TESTING AND VALIDATION

Software Testing is the process of evaluating a system and its components with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Software Testing is an important factor of software quality assurance. Errors can be introduced at any stage in the development process. Even after checking and correcting the errors at every stage, there may still be some errors in the system. It can also be stated as the process of validating and verifying that a software program or application or product: meets the business and technical requirements that guided its design and development, works as expected and can be implemented with the same characteristic.

7.1 TESTING METHOD

There exists a host of methods that can be used to successfully reveal errors in a system. These are:

- **Unit Testing** is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed.
- **Integration Testing** is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
- **Functional Testing** is a software testing process used within software development in which software is tested to ensure that it conforms to all requirements. Functional testing is a way of checking software to ensure that it has all the required functionality that's specified within its functional requirements.
- S**ystem Testing** is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

- **White Box Testing** (also known as Clear Box Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs.
- **Black Box Testing,** also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

7.2 DIFFERENT TESTS DONE

Boundary value testing is applied on the texts which are dynamically input by the user. This testing can be used to test if the number of characters in text does not exceed 280 characters and go below 1. Hence the max, max-1, nom, min+1 and min values in the testing are 281, 280, 140, 1 and 0 respectively. The expected output that is the sentiment behind the text is obtained for max-1, nom and min+1 value. For max and min values should be handled by appropriate error messages.

Equivalence class testing is applied on datasets which are divided into two classes, test set and training set. The train dataset is used to model the classifier and classify the test dataset based on the sentiment it is associated with. A k-fold validation method is used, which divides the dataset into test and train sets. The k is set to 10, which means that the dataset is divided in different combinations 10 times. The best division is considered.

System testing is an end-to-end testing, which is navigating through all the features/modules and validating whether the end feature/module is works properly. The proposed system modules include data collection, data cleaning, feature extraction, training and testing the dataset, and computing performance evaluation measures such as accuracy, precision, recall and F1 score.

7.3 TEST CASE

Test cases are the documents that cover all possible scenarios for specific requirement. It contains serial number, Action/Description, Input, Expected output, actual output, status and comment sections.

Table 7.1: Test case for Pre-processing

TEST CASE	#1
Name of Test	Pre-processing
Items being Tested	Images passed by the user
Input	Image
Expected output	pre-processed image (ie., thresh, deskew, scale up, blur)
Actual output	Pre-processed image
Remarks	Pass
Comments	The pre-processing of image has been successful.

Table 7.2: Test case for Text Extraction

TEST CASE	#2
Name of Test	Text extraction
Items being Tested	Text-extraction model
Input	Pre-processed image
Expected output	Text extracted from image
Actual output	Text extracted for image and displayed on the screen
Status	Pass
comments	The texts are extracted as expected.

Table 7.3: Test case for search

TEST CASE	#3
Name of Test	search
Items being Tested	Extracted text
Input	Word to be searched
Expected output	Highlight the word when searched
Actual output	Highlight the word when searched
Remarks	Pass
Comments	Words are correctly searched and highlighted

Tables 7.1, 7.2 and 7.3 indicate the expected outputs and actual output of the given test cases. Most of them pass the given criteria successfully thus achieving higher accuracy.

CHAPTER 8

RESULTS

CHAPTER 8

RESULTS AND DISCUSSION

8.1 Snapshots and Description

Figure 8.1 shows the home page of the application.

Fig 8.1: Home Page

Figure 8.2 displays about the application. It gives the overall explanation of the application.

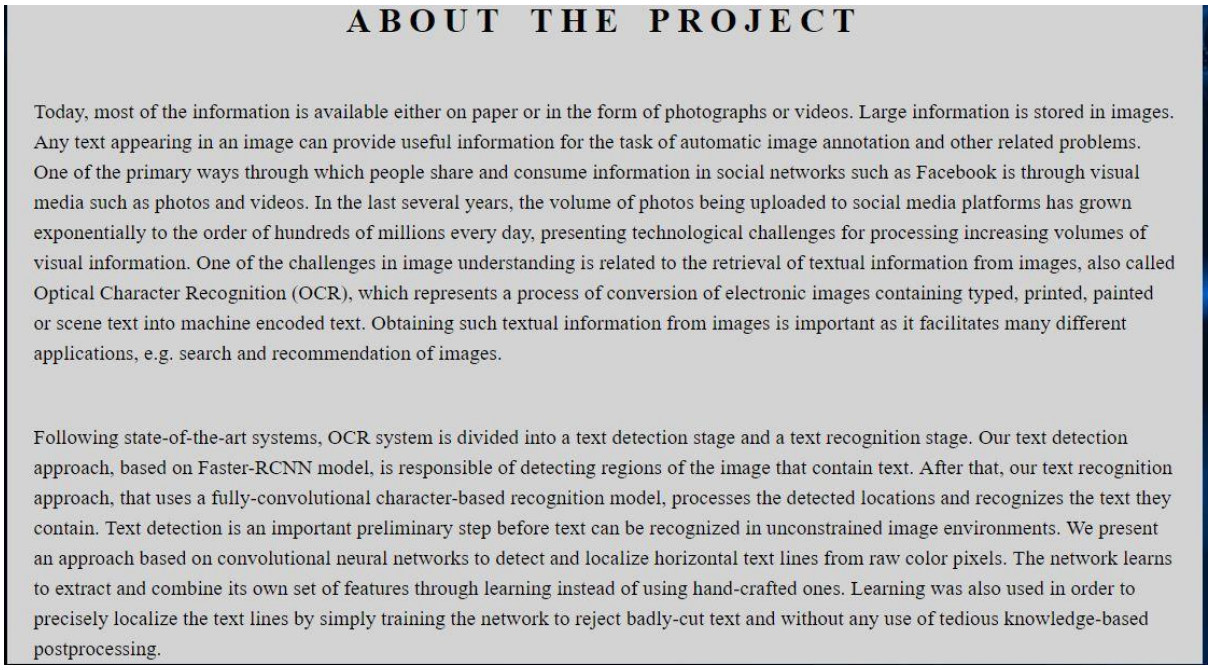


Fig 8.2: About the Project Page

Figure 8.3 shows different filters that are available in the current application and a brief description on each of them.

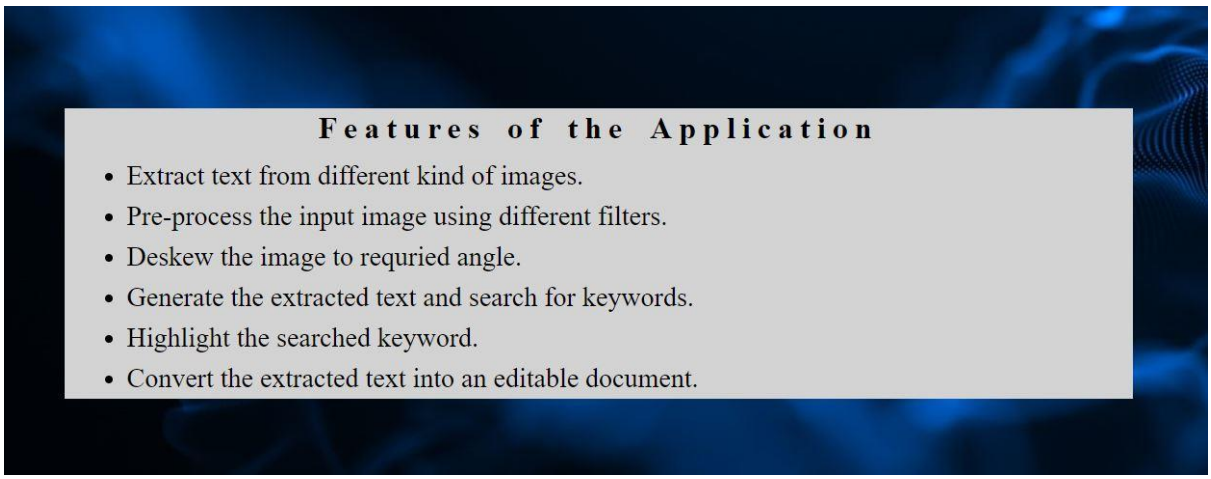


Fig 8.3: Features Of The Application

Figure 8.4 displays information about all the team members.

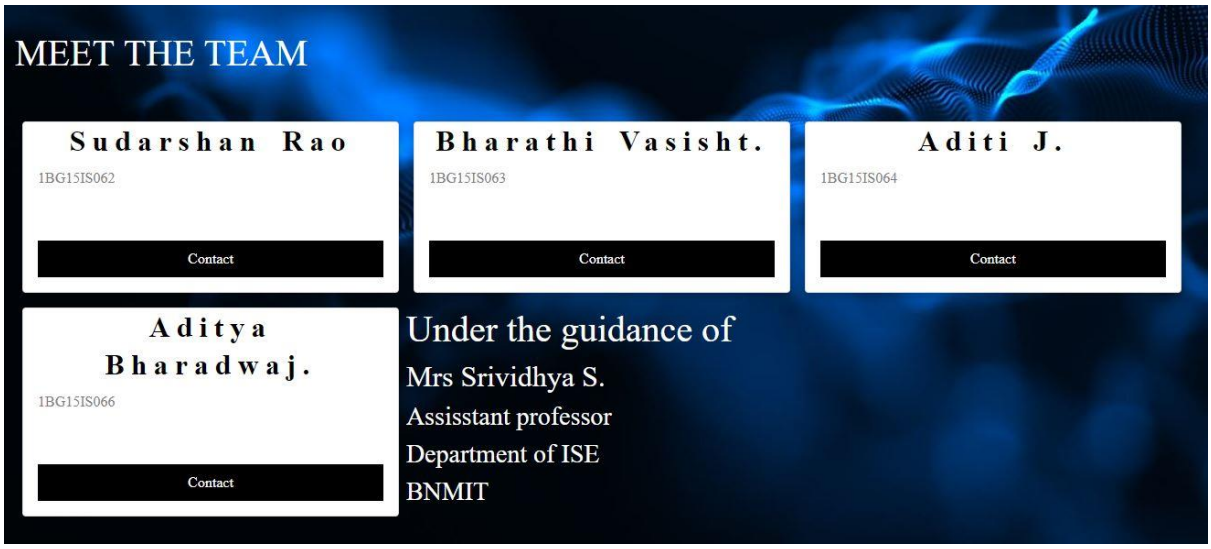


Fig 8.4: Team Members

Figure 8.5 shows the browsing of an image from the system files. Once the image is selected, the application gives a preview of an image.

Fig 8.5: Browsing an Image with regular test

Figure 8.6 shows the result after pre-processing an image. In this case we made use of median blur filter to pre-process an image.

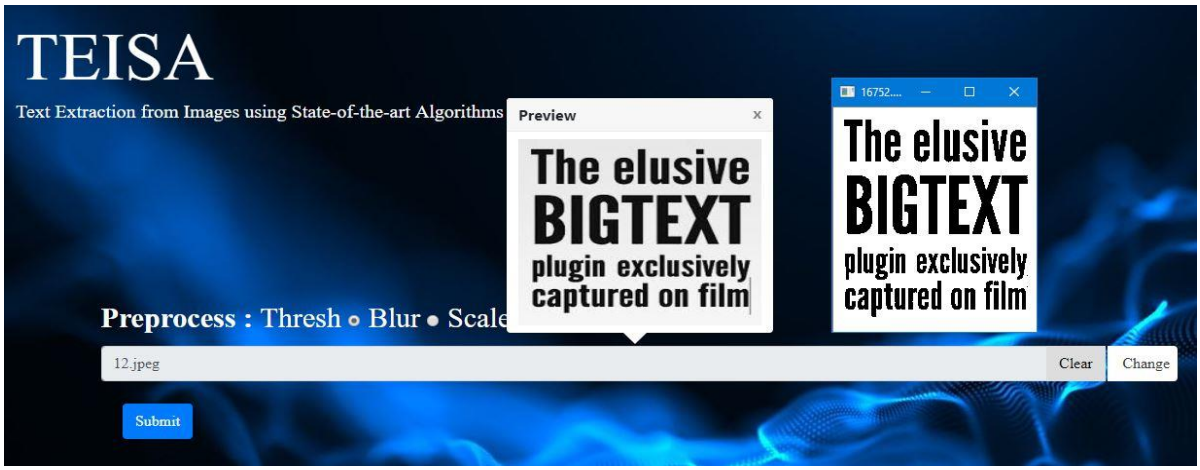


Fig 8.6: Pre-processed Image

Figure 8.7 shows the output of the text extracted from the image with 100% accuracy.

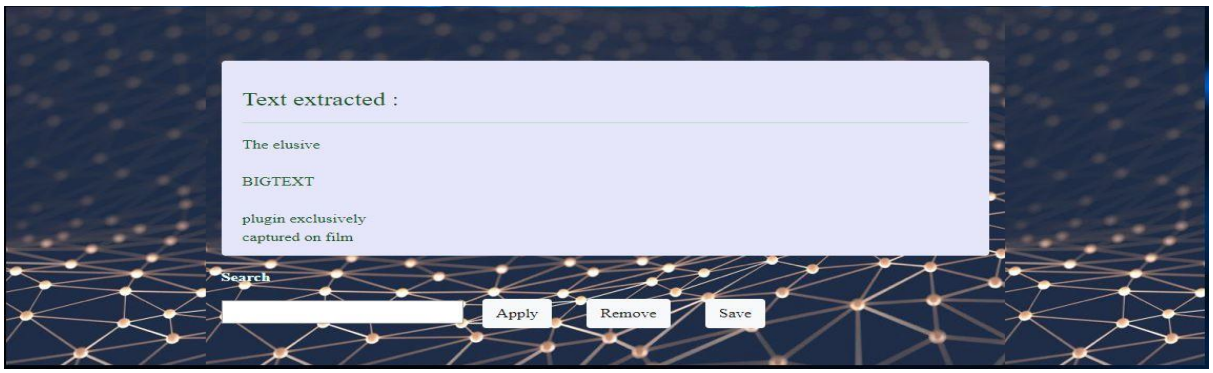


Fig 8.7: Text Extracted from Image

Figure 8.7 shows another example, where the text in the image is tilted at an angle. On applying de-skew filter, we can rotate the text and bring it to a horizontal position as shown in Figure 8.8.

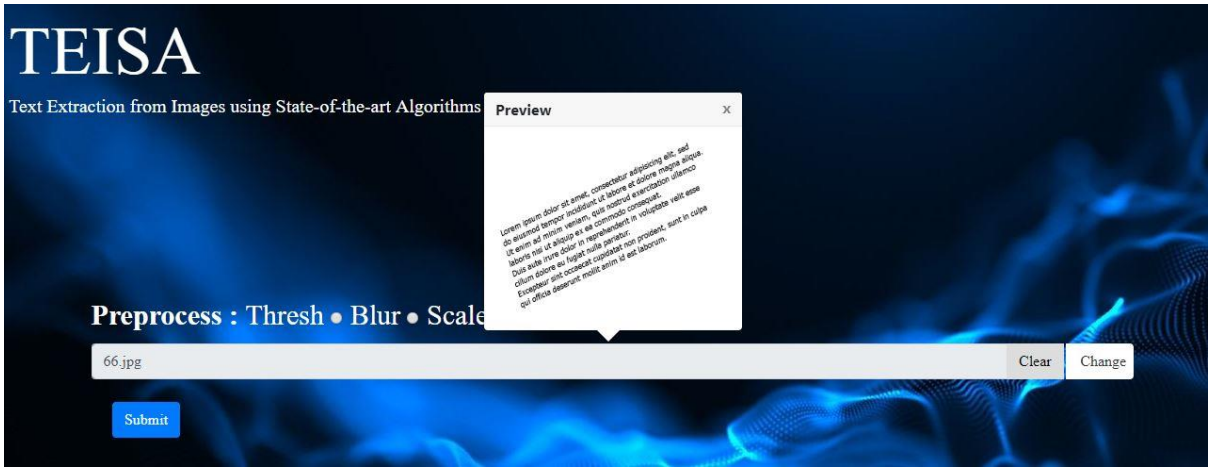


Fig 8.8: Browsing a Skewed Image

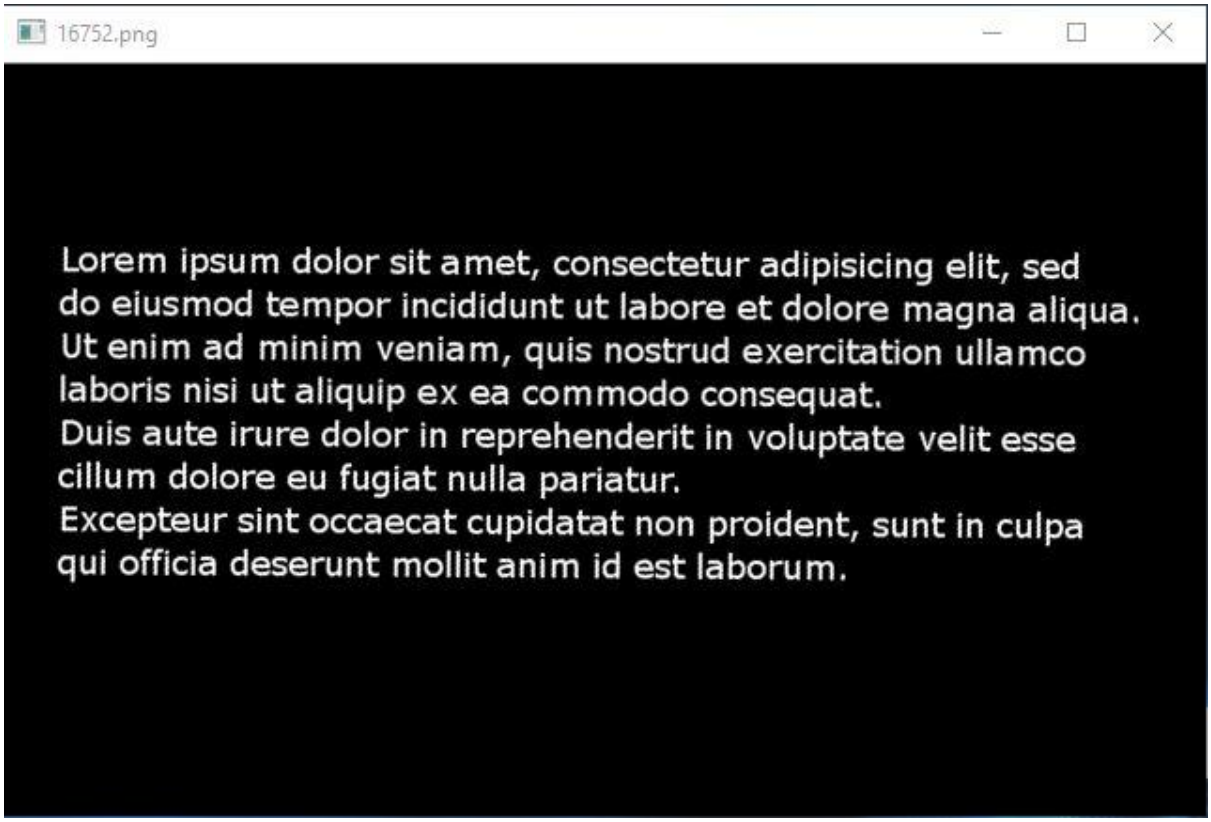


Fig 8.9: Output after De-Skewing an Image

Figure 8.10 shows the working of search Model. Here the search is case insensitive. The user can type the word to be searched and the keyword will be highlighted once found.

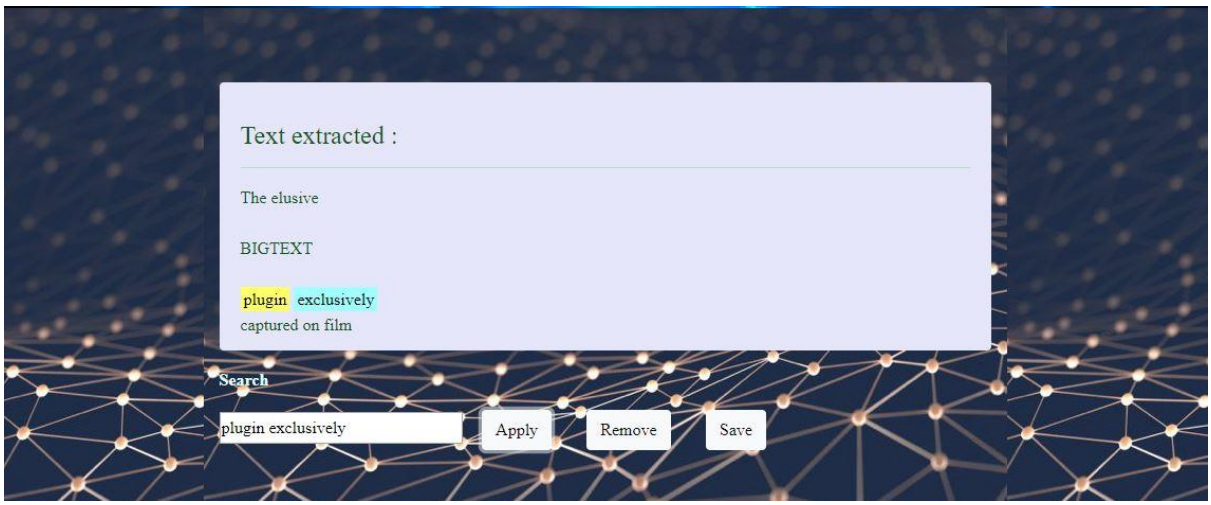


Fig 8.10: Search for a keyword

One of the key features about this application is that, the user can even store the text that is extracted in a document and convert it to an editable format. Figure 8.11 shows the text being displayed in a notepad.

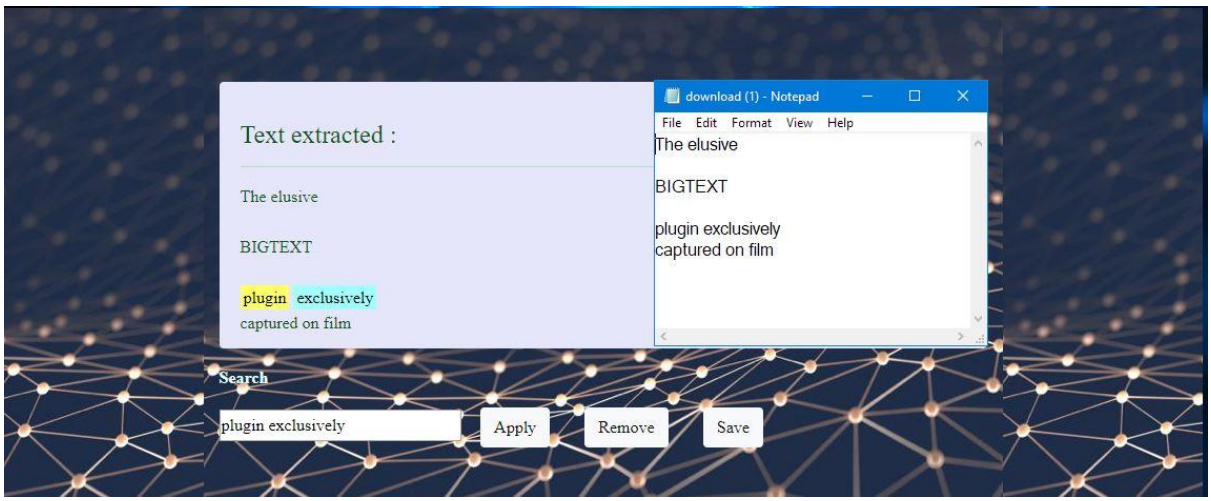


Fig 8.11: Storing the Text in Document

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENTS

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

The proposed system provides text extraction from images where the user will store the text in a document and it is in an editable format. The system counters most of the problems as specified in the internet. The main problems, such as extracting all the texts and storing the them has been successfully countered using Convolutional Neural Network. The browse option allows user to choose the image which contains text and the filters allow the images to be pre-processed to remove noise and make it easy for the system to detect and extract the text. Search is one of the different models implemented, that allows user to search for a keyword in the entire text. As the future heads in building a community with everything digital, such a system helps in making the work easier.

Future Enhancement

Future work of the project includes developing an application for the smart phones and making it available in cross platform and also improving the user interface. Aim is to even make it even multi-lingual.

REFERENCES

[1]. Fedor Borisyuk, Albert Gordo and Viswanath Sivakumar, "Rosetta: Large Scale System for Text Detection and Recognition in Images", KDD 2018, August 19-23, 2018, London, United Kingdom.

[2]. Manolis Delakis and Christophe Garcia, "Text Detection With Convolutional Neural Networks", VISAPP 2008 - International Conference on Computer Vision Theory and Applications

[3]. Andreas Veit, Tomas Mater, Lukas Neumann, Jirı Mata and Serge Belongie, "COCO-Text: Dataset And Benchmark For Text Detection And Recognition In Natural Images", arXiv:1601.07140v2 [cs.CV] 19 Jun 2016.

[4]. Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun,"Deep Residual Learning for image recognition",IEEE conference on computer vision and pattern recognition, 2016,page- 770-778.

[5]. Uma B Karanje and Rahul Dagade,"Survey On Text Detection,Segmentation and Recognition From Natural Scene Image", International Journal of Computer Applications,2014.

[6]. Shubhrita Tiwari Shailendra Singh Kathait," Application of Image Processing and Convolution Networks in Intelligent Character Recognition for Digitized Forms Processing", Int. J. Comput. App,2018.

REFERENCES

PUBLICATIONS

IEEE

The International Conference on Cloud Computing in Emerging Markets, 2019

Pre-conference Workshop

held on 1st March'19 at The Oterra, Electronics City, Bengaluru.

is pleased award this participation certificate, in the category of Student-Project-Showcase, to

Aditi J

(of B.N.M.Institute of Technology, Bengaluru)

in appreciation of her co-authoring and presenting a proposal titled:

'Text extraction from Images using Convolutional Neural network'

with Bharathi V, Aditya Bharadwaj and Sudarshan Rao.

Dr. Gopal Pingali	Sreekrishnan Venkiteswaran	Raj Tumuluri	Dr. T.S.Mohan
Steering Committee Chair	General Chair	General Chair	Steering Committee Chair

IEEE

The International Conference on Cloud Computing in Emerging Markets, 2019

Pre-conference Workshop

held on 1st March'19 at The Oterra, Electronics City, Bengaluru.

is pleased award this participation certificate, in the category of Student-Project-Showcase, to

Aditya Bharadwaj

(of B.N.M.Institute of Technology, Bengaluru)

in appreciation of his co-authoring and presenting a proposal titled:

'Text extraction from Images using Convolutional Neural network'

with Bharathi V, Sudarshan Rao and Aditi J.

Dr. Gopal Pingali	Sreekrishnan Venkiteswaran	Raj Tumuluri	Dr. T.S.Mohan
Steering Committee Chair	General Chair	General Chair	Steering Committee Chair

IEEE

The International Conference on Cloud Computing in Emerging Markets, 2019

Pre-conference Workshop

held on 1st March'19 at The Oterra, Electronics City, Bengaluru.

is pleased award this participation certificate, in the category of Student-Project-Showcase, to

Bharati V

(of B.N.M.Institute of Technology, Bengaluru)

in appreciation of her co-authoring and presenting a proposal titled:

'Text extraction from Images using Convolutional Neural network'

with Aditya Bharadwaj, Sudarshan Rao and Aditi J.

Dr. Gopal Pingali	Sreekrishnan Venkiteswaran	Raj Tumuluri	Dr. T.S.Mohan
Steering Committee Chair	General Chair	General Chair	Steering Committee Chair

The International Conference on Cloud Computing in Emerging Markets, 2019

Pre-conference Workshop

held on 1st March'19 at The Oterra, Electronics City, Bengaluru.

is pleased award this participation certificate, in the category of Student-Project-Showcase, to

Sudarshan Rao

(of B.N.M.Institute of Technology, Bengaluru)

in appreciation of his co-authoring and presenting a proposal titled:

'Text extraction from Images using Convolutional Neural network'

with Bharathi V, Aditya Bharadwaj and Aditi J.

Dr. Gopal Pingali	Sreekrishnan Venkiteswaran	Raj Tumuluri	Dr. T.S.Mohan
Steering Committee Chair	General Chair	General Chair	Steering Committee Chair

Text extraction from Images using Convolutional Neural network

Sudarshan Rao M
sudarshanmvrao@gmail.com
+91 9740365018
B.N.M Institute of technology

Bharati V
ambhrani.bharathi@gmail.com
+91 9742314744
B.N.M Institute of technology

S G Aditya Bharadwaj
sgaditya97@gmail.com
+91 9066635315
B.N.M Institute of technology

Aditi J
adi.j.acharya@gmail.com
+91 8762131538
B.N.M Institute of technology

Overview:

Extracting text and realizing it on a document using the state-of-the-art algorithms such as Convolutional neural networks and the techniques that follow it. It is going to be very helpful for those who are in data entry environment who can get the job done of some photos of the bills and invoices. Using the current technology to solve such problems in the real world with system possible solutions is one of the greatest goals of the project.

To get an accurate and relevant search results when one searches for an image To enable people in the industry, to directly get the text which is on a bill rather than typing it out manually.

Innovation presented:

- Extracting text from an image containing text and converting it into an editable format.
- Implementing the search technique to identify specific content on the image.
- The system is trained efficiently with all combination of inputs using the dataset
- Outputs the location of the words(co-ordinates) on the image.
- Highlight the text on the image.

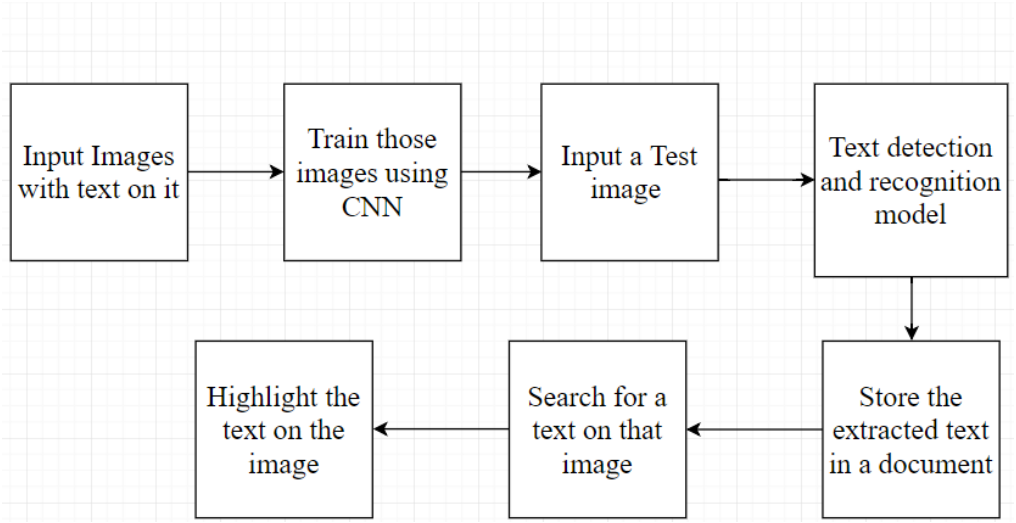
Text extraction model:

This model is done in two steps: detection and recognition. First, we detect rectangular regions in the image potentially containing text. In the second step we perform text recognition, where,for each of the detected regions, a CNN is used to recognize and transcribe the word in the region to

detect the text, we make use of faster RCNN. It represents an image into a convoluted feature map. This map is taken as input and produces bounding boxes that contains text. In the next stage, we recognize the text.

Text recognition model:

The text recognition model is a CNN based on the ResNet18 architecture. To train the model,cast it as a sequence prediction problem, where the input is the image containing the text to be recognized and the output is the sequence of characters in the word image. Use the connectionist temporal classification (CTC) loss to train our sequence model. Casting the problem as one of sequence prediction allows the system to recognize words of arbitrary length and to recognize out-of-vocabulary words (i.e., words that weren't seen during training).



Architecture of the proposed system

Technologies used for Text Recognition:

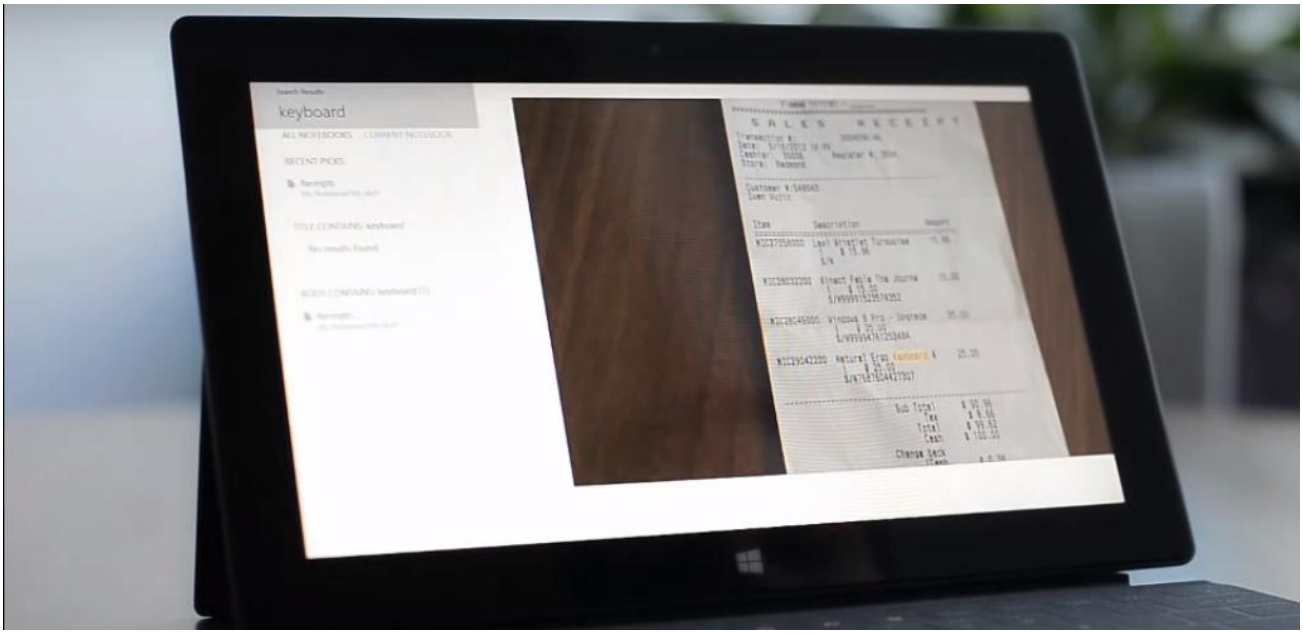
Convolution Neural Network

CNN is a class of deep, feed-forward artificial neural networks (where connections between nodes do *not* form a cycle) & use a variation of multilayer perceptrSons designed to require minimal pre-processing. These are inspired by animal visual cortex.

Recurrent Neural network

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic

temporal behavior for a time sequence. RNN is a sequence of neural network blocks that are linked to each other's like a chain. Each one is passing a message to a successor.



Expected output of highlighted text on the image

Conclusions:

- Today the most information is available either on paper or in the form of photographs. Large information is stored in images. The current technology is restricted to extracting text against clean backgrounds. Thus, there is a need for a system to extract text from general backgrounds.
- Text Extraction and recognition in Images has become a potential application in many fields like Image indexing , Robotics, Intelligent transport systems etc.
- For example capturing license plate information through a video camera and extracting license number in traffic signals
- However, variations of text due to differences in size, style, orientation, and alignment, as well as low image contrast and complex background make the problem of automatic text extraction extremely challenging.

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Bharati V

In recognition the publication of the manuscript entitled

Text Extraction from Images using Convolutional Neural Network

published in our Journal Volume 6 Issue 5 May 2019

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.211

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

S G Aditya Bharadwaj

In recognition the publication of the manuscript entitled

Text Extraction from Images using Convolutional Neural Network

published in our Journal Volume 6 Issue 5 May 2019

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.211

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Aditi J

In recognition the publication of the manuscript entitled

Text Extraction from Images using Convolutional Neural Network

published in our Journal Volume 6 Issue 5 May 2019

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.211

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

S Srividhya

In recognition the publication of the manuscript entitled

Text Extraction from Images using Convolutional Neural Network

published in our Journal Volume 6 Issue 5 May 2019

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.211

www.irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

Sudarshan Rao M

In recognition the publication of the manuscript entitled

Text Extraction from Images using Convolutional Neural Network

published in our Journal Volume 6 Issue 5 May 2019

Editor in Chief

E-mail : editor@irjet.net

Impact Factor : 7.211

www.irjet.net

Text extraction from Images using Convolutional Neural Network

Bharati V[1], Sudarshan Rao M [2], Aditi J[3] ,S G Aditya Bharadwaj[4],S Srividhya[5]

[1]*B.E Student, Dept. of Information Science and Engineering, BNM Institute Of Technology, Bengaluru-560070(Karnataka).*
[2]*B.E Student, Dept. of Information Science and Engineering, BNM Institute Of Technology, Bengaluru-560070(Karnataka).*
[3]*B.E Student, Dept. of Information Science and Engineering, BNM Institute Of Technology, Bengaluru-560070(Karnataka).*
[4]*B.E Student, Dept. of Information Science and Engineering, BNM Institute Of Technology, Bengaluru-560070(Karnataka).*
[5]Assistant Professor, Dept. of *Information Science and Engineering, BNM Institute Of Technology*, *Bengaluru-560070(Karnataka).*

Abstract - *In this paper, we present a text extraction model that is designed to process the images that are uploaded by the user. Images along with texts have become one of the common ways to exchange information; hence understanding these images plays an important role. We present efficient text detection and extraction models along with search.*

***Key Words*: Text detection, text recognition, CNN, Text Extraction, Pre-Processing.**

1. Introduction

Text detection [2] and extraction is used to get the extracted text in a document using the state-of-the-art algorithms such as Convolutional neural networks and the techniques that follow it. It is going to be very helpful for those who are in data entry department who can get the content of some photos of the bills and invoices directly on their screens rather than typing it out manually. Using the current technology to solve such problems in the real world with system possible solutions is one of the greatest goals of the project.

To get an accurate and relevant search results when one searches for a text in an image. This is to enable people in the industry, to directly get the image by searching for a keyword in that image.

1.1 Innovation Presented

- Browse for the image that contains text.
- Extracting text from the image.
- Implementing the search technique to identify keyword in the text.
- The text that is extracted can also be stored in a document, which is in an editable format.
- The system is trained efficiently with all combination of inputs using the datasets.

2. Text Extraction Model

This model is done in two steps: detection and recognition. [1] First, we detect those regions in the image potentially containing text. In the second step we perform text recognition, where, for each of the detected regions, a CNN is used to recognize and transcribe the word in the region to detect the text. It represents an image into a convoluted feature map. This map is taken as input producing bounding boxes that contain text [5]. In the last stage, we extract the text.

2.1 Comparison Of Different Technologies Used For Text Extraction

- Region based Method: Region-based method uses the properties of the color or gray scale in the text region or their differences to the corresponding properties of the background. They are based on the fact that there is very little variation of color within text and this color is sufficiently distinct from it's immediate background. Text can be obtained by thresholding the image at intensity level between the text color and that of its immediate background. This method is not robust to complex background and is further divided into two sub-approaches: connected component (CC) and edge based.
- CC based Method: CC-based methods use a bottom-up approach by grouping small components into successively larger components until all regions are identified in the image. A geometrical analysis is required to merge the text components using the spatial arrangement of those components so as to filter out non-text components and the boundaries of the text regions are marked. This method locate locates text quickly but fails for complex background.

- Edge based Method: Edges are a reliable feature of text regardless of color/intensity, layout, orientations, etc. Edge based method is focused on high contrast between the text and the background. The three distinguishing characteristics of text embedded in images that can be used for detecting text are edge strength, density and the orientation variance. Edge based text extraction algorithm is a general-purpose method, which can quickly and effectively localize and extract the text from both document and indoor/ outdoor images. This method is not robust for handling large size text.
- Texture based Method: This method uses the fact that text in images has discrete textural properties that distinguish them from the background. The techniques based on Gabor filters, Wavelet, Fast Fourier transform (FFT), spatial variance, etc. are used to detect the textual properties of the text region in the image . This method is able to detect the text in the complex background. The only drawback of this method is large computational complexity in texture classification stage.
- Morphological based Method: Mathematical morphology is a topological and geometrical based method for image analysis. Morphological feature extraction techniques have been efficiently applied to character recognition and document analysis. It is used to extract important text contrast features from the processed images. These features are invariant against various geometrical image changes like translation, rotation, and scaling. Even after the lightning condition or text color is changed, the feature still can be maintained. This method works robustly under different image alterations.

3. Pre- Processing

In this system, there are 4 different types of filters used to pre-process the image:

1. Binarize
2. Median blur
3. Scale up
4. De-skew

A combination of these filters is used in order to get the highest accuracy in extracting all the text present in the image.

4. Text Recognition Model

The text recognition model is a CNN based on the ResNet18 [4] architecture. To train [3] the model, cast it as a sequence prediction problem, where the input is the image containing the text to be recognized and the output is the sequence of characters in the word image. Use the connectionist temporal classification (CTC) loss to train the sequence model. Casting the issue as one of sequence prediction allows the system to recognize words of arbitrary length and to recognize out-of-vocabulary words (i.e., words that weren't seen during training).

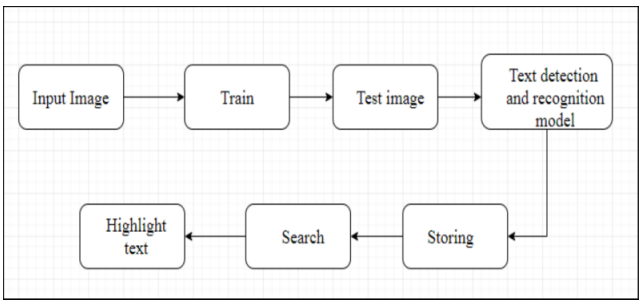


Fig -1: Architecture of the proposed system

4.1 Technologies used for Text Recognition

Convolution Neural Network

CNN is a class of deep, feed-forward artificial neural networks (where connections between nodes do not form a cycle) & use a variation of multilayer perceptrons designed to require minimal pre-processing. These are inspired by animal visual cortex. Convolutional neural network are used to find patterns in an image. We do that by convoluting over an image and looking for patterns. In the first few layers of CNNs the network can identify lines and corners, but we can then pass these patterns down through our neural net and start recognizing more complex features as we get deeper.

Recurrent Neural network

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. RNN is a sequence of neural network blocks that are linked to each other's like a chain. Each one is passing a message to a successor.

5. Searching Model

There are two stages that are implemented in search:

- Identifying the keyword.
- Highlighting the keyword in the text once found.

First take an input from the text field, For example: "Good Evening". This is converted into a JavaScript regular expression- "/\b (Good| Evening) \b/". This regular expression will match any of the entered words where they appear in the content area of the page. The apply() method is called, it generates a regular expression from the keywords, clears any existing highlights on the page, and then calls

the highlightWords() method passing a reference to the selected start node.

6. Output

Fig -2: Browse the Image

Fig -3: Pre-process the image

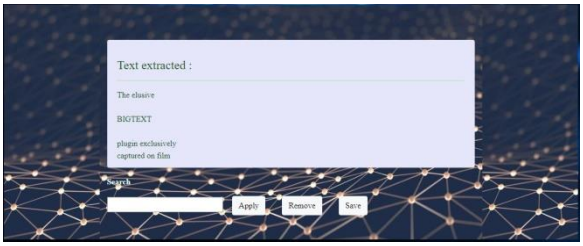


Fig -4: Text extracted

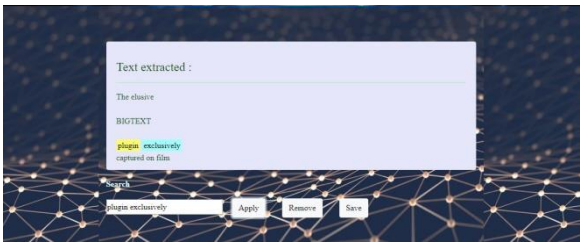


Fig -5: Search for keyword

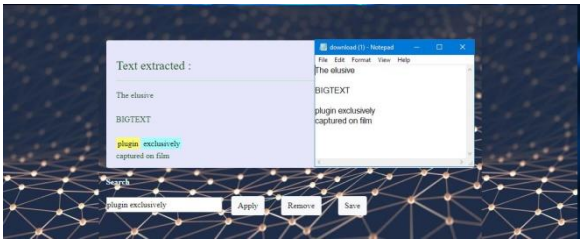


Fig -6: Storing the text in Document

7. Conclusion

Today most of the information is available either on paper or in the form of photographs. The current technology is restricted to extracting text against clean backgrounds. Thus, there is a need for a system to extract text from general backgrounds. Text Extraction and recognition in Images has become a potential application in many fields like [6] Image indexing, Robotics, Intelligent transport systems etc. For example capturing license plate information through a video camera and extracting license number in traffic signals. However, variations of text due to differences in size, style, orientation, and alignment, as well as low image contrast and complex background make the hitch of automatic text extraction extremely challenging.

The current system can extract the text from images and perform search action for a particular keyword and also store the extracted text in a document which is in an editable format.

8. Future Enhancement

Future work of the project may include developing an application for the smart phones and making it available in cross platform and also improving the user interface. Make it multi-lingual text extraction. In Military, the code maybe written in such a way that, white text is on white background and extracting such text might be critical at the situation. Detecting watermarks.

References

[1] Fedor Borisyuk, Albert Gordo and Viswanath Sivakumar, "Rosetta: Large Scale System for Text Detection and Recognition in Images", KDD 2018, August 19-23, 2018, London, United Kingdom.

[2] Manolis Delakis and Christophe Garcia, "Text Detection With Convolutional Neural Networks", VISAPP 2008 - International Conference on Computer Vision Theory and Applications.

[3] Andreas Veit, Tomas Mater, Lukas Neumann, Jirı Mata and Serge Belongie, "COCO-Text: Dataset And Benchmark For Text Detection And Recognition In Natural Images", arXiv:1601.07140v2 [cs.CV] 19 Jun 2016.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun,"Deep Residual Learning for image recognition",IEEE conference on computer vision and pattern recognition, 2016,page- 770-778.

[5] Uma B Karanje and Rahul Dagade,"Survey On Text Detection,Segmentation and Recognition From Natural Scene Image", International Journal of Computer Applications,2014.

[6] Shubhrita Tiwari Shailendra Singh Kathait," Application of Image Processing and Convolution Networks in Intelligent Character Recognition for Digitized Forms Processing", Int. J. Comput. App,2018.