**PROBLEM STATEMENT**

We find There are two types of text occurring in a video.

Natural text.

Superimposed text.

Natural Text:

**Natural text is the text that appears in the video while it is being recorded. These texts are part of a video recorded scene**

Example: Flat number, Vehicle number Plate.

Figure 1 Natural Text

Superimposed text: Superimposed text is text that is added to video after it has been shot but before it was actually part of the scene.

Ex: Text that appears in a news video.

Figure 2 Superimposed Text

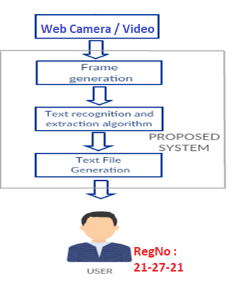
Superimposed text contains information that is extremely important, whereas natural text is not very useful because it contains information of lesser worth. The primary goal of the suggested approach is to identify instances of overlay text in the video.**SYSTEM OVERVIEW**

The proposed system has three main components:

Frame Generation

Text Recognition and Extraction

Text File Generation

Figure : System Overview

*Frame Generation:* The video is turned into frames in this stage. Images from a certain point in a video are called frames. The frames are generated at regular intervals so that text in the following frame doesn't repeat itself a lot. Any picture format can be used to save these frames.

When converting video to frames, the users will have two choices. The first is to convert the whole video, and the second is to convert a particular section of the video. People will choose option one if they want text from the entire video. People will choose the second option, which allows them to choose the start time and finish time for text extraction, if they just require text from a specific time period. The chosen video clip will be converted into photos and kept in a separate folder for convenient access while a text extraction technique is applied to it.

*Text Recognition and Extraction:* *This process is carried out for each frame. Using the algorithm explained in the following section, the text region is found in this phase. To improve the effectiveness of text extraction, the discovered text sections are then refined. The text extraction algorithm is used on the areas that have been detected. The effectiveness of text detection is affected by font colour, text size, backdrop colour, and video resolution.*

*Text File Generation:* A text file is used to hold the retrieved text. The created text is added to the preceding text in the text file and stored for each frame. People will be provided the path to the result file after all of the photos' text has been extracted. The text file is considerably smaller in size than the movie. As a result, information may be accessed more quickly and memory is conserved.

**Methodology**

The suggested framework is described in detail in this section and is illustrated in Fig. 3.1 The text detector, script identifier, and text recognizer are the three key components that make up the entire system. A wide range of systems, such as content summarization, key-word-based alert generation, and indexing and retrieval, can be built on top of these modules at the application layer. The first module, text detector, is required to locate and recognise every piece of text in a frame. Since text can appear in many scripts (within the same frame), the text portions that have been found are provided to the script identification module, which splits the text lines according to the script (English and other Indian Language being the two scripts considered in the present study). The text is then forwarded to the respective recognition engines of each script, where they convert the images of the text lines into strings that can be used to a range of situations.

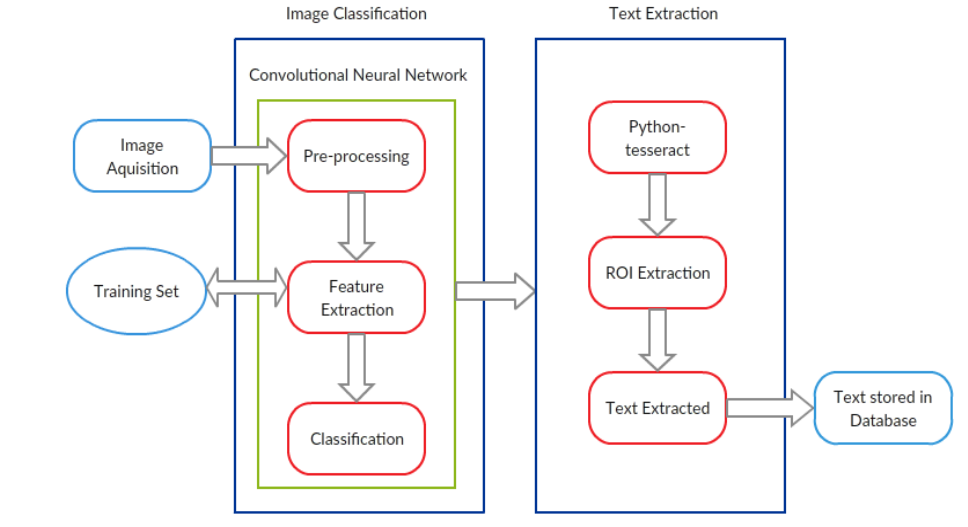
**Text detection**

The proposed framework's first step is the identification of potential text sections in the retrieved video frames. Modern object detectors based on convolutional neural networks (CNN) have been used to recognize textual information in a given frame. Despite the fact that many object detectors are highly accurate in detecting and recognizing various objects and have been trained on tens of thousands of class examples, they cannot be used to directly recognize text regions in photos. These models must be adjusted for the particular issue of separating text from non-text regions. These models' convolutional bases can be trained entirely from scratch or pre-trained models can be improved by giving them practice on regions with and without text.

**Determining Bounding Boxes**

The individual characters will initially be combined into a single connected component before the bounding box of the text region is computed. To eliminate any outliers, this can be done by morphologically closing and then opening.

***Text Extraction***The Tesseract OCR package, which includes an optical character recognition (OCR) engine called libtesseract and a command-line tool called Tesseract, is used to implement text extraction. Long Short-Term Memory (LSTM) based OCR engine, which concentrates on line identification and also identifies character pattern, is a new neural network included in Tesseract. The building blocks of recurrent neural networks are the LSTM network. An optical character recognition (OCR) tool in Python called the Python-Tesseract is used to extract text. This tool can "read" text that is embedded in photos and recognise it. This application reads all image kinds, including JPEG, PNG, GIF, bmp, and others. It is a wrapper for Google's Tesseract-OCR Engine. The Python Imaging Library is capable of displaying these images. Once the text information has been retrieved from the image files utilising

Fig.2. Process of Image Classification and Text Extraction

**Description of Deep Learning Model Used**

**Optical Character Recognition (OCR)**

OCR systems transform the image or text of a two-dimensional text frame that may contain text into machine-readable text. OCR normally consists of several sub-processes in order to be carried out as accurately as is practical.

The sub processes are:

Preprocessing of the Image

Text Localization

Character Segmentation

Character Recognition

Post Processing

The sub-processes in the list above of course can differ, but these are roughly steps needed to approach automatic character recognition. In OCR software, it’s main aim to identify and capture all the unique words using different languages from written text characters.

Optical Character Recognition process (Courtesy)

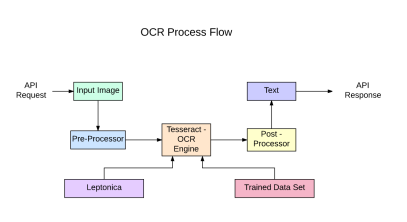
Google Tesseract

The most modern OCR engine, Google Tesseract, offers great accuracy for many different languages, including English. We used Tesseract version 5.2, a recent release from Google, in our system. Deep neural networks, more specifically recurrent neural networks with extended short-term memory architecture, are used to create version 5.2. The recognition engine receives the English text lines as input and outputs the relevant textual strings.

In HP Labs, Bristol, Tesseract was first developed as a Ph.D. research project. Between 1984 and 1994, HP created it and it became well-known. Tesseract was made available as open-source software by HP in 2005. It has been created by Google since 2006.

Tesseract-OCR

Available under the Apache 2.0 licence is the Tesseract-OCR Engine. To extract printed text from photos, use this technique. Many different languages are supported. Tesseract doesn't come with a built-in GUI, but there are a number of them on third-party websites. Tesseract works with a wide range of frameworks and computer languages. It can be used in conjunction with the current layout analysis to identify text within a huge document or with an outside text detector to identify text from a picture of a single text line.

OCR Process Flow to build API with Tesseract from a [blog post](https://medium.com/@balaajip/optical-character-recognition-99aba2dad314)

OCR Process Flow from a blog article to develop API using Tesseract

A new neural network subsystem outfitted as a text line recognizer is part of Tesseract 4.00. It was modified for Tesseract in C++ and is based on the Python-based LSTM implementation from OCRopus. Although Tesseract's neural network architecture predates TensorFlow, it is compatible with it because TensorFlow also has a network description language called Variable Graph Specification Language (VGSL).

Typically, we utilise a Convolutional Neural Network to identify an image with a single character (CNN). Character sequences make up text of any length, and RNNs—of which LSTM is a well-liked variant—can be used to address such problems. To find out more about LSTM, read this article..

OCR with Pytesseract and OpenCV

Pytesseract or Python-tesseract is an OCR tool for python that also serves as a wrapper for the Tesseract-OCR Engine. It can read and recognize text in images and is commonly used in python ocr image to text use cases.

It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

More info about Python approach read [here](https://github.com/madmaze/pytesseract). The code for this tutorial can be found in this [repository](https://github.com/NanoNets/ocr-with-tesseract).

***Convolutional Neural Network (CNN)***

In order to accomplish sophisticated image categorization, CNN, a straightforward deep learning-based method, transforms input data into a meaningful representation. As seen in Figure 4.2, these typically include convolutional layers, pooling layers, and fully linked layers (also known as dense layers). By giving each neuron, which connects to some of the preceding neurons, the proper weights and biases, convolutional layers learn a feature representation of the input image.

An activation function, such as Rectified Linear Unit, is passed the result (ReLU). The size of features is reduced in part by the pooling layer. With the use of probabilities determined by softmax, a fully connected layer combines all the learnt features of the preceding layers to provide predictions in the output layer (classification layer).

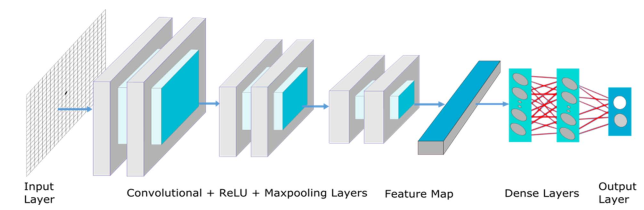
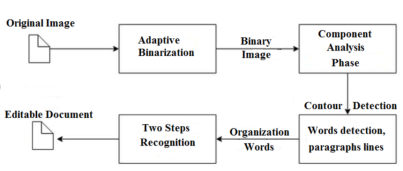


Fig. 3. Network architect of proposed CNN model

Long short-term memory (LSTM)

Although LSTMs are excellent at learning sequences, when there are a lot of states, they become quite sluggish. Empirical evidence suggests that asking an LSTM to learn a long sequence is preferable to asking it to learn a short sequence of many classes. Tesseract evolved from the Python-based OCRopus model, which was a clone of the C++-based LSMT known as CLSTM. The LSTM recurrent neural network model is implemented as CLSTM in C++, which utilises the Eigen toolkit for numerical calculations.



Tesseract 3 OCR process

Legacy Tesseract 3.x was dependent on the multi-stage process where we can differentiate steps:

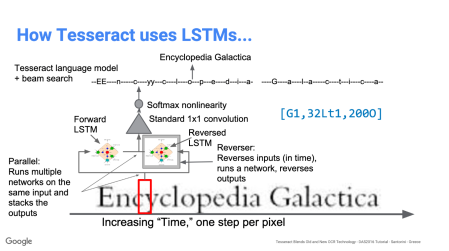
Word finding

Line finding

Character classification

Text lines were grouped into blobs, and the lines and regions were then examined for fixed pitch or proportional text in order to locate words. Depending on the type of character spacing, text lines are divided into words in a variety of ways. The process of recognition then continues in two passes. First, each word is tried to be recognised individually. An adaptive classifier receives each word that is acceptable as training data.

Code cleanup and the addition of a new LSTM model were efforts made to modernise the Tesseract tool. Line by line, boxes (rectangles) of the input image are analysed before being fed into the LSTM model and producing output. We can see how it functions in the figure below.



How Tesseract uses LSTM model [presentation](https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/6ModernizationEfforts.pdf)

Tesseract performs better following the addition of a new training tool and extensive data and font training for the model. However, it is still insufficient to work with handwritten writing and unusual fonts. Top layers can be adjusted or retrained for experimentation.

**Conclusion**

In the present work we have discussed our proposed method of recognizing and extracting text from video. The system automates the manual process of extracting text from videos and is therefore economical in terms of time and human effort. The system is implemented in the Python programming language. The system is primarily used for educational and news videos that contain information in text form.

In the present work, we present a comprehensive framework for text recognition and recognition in video frames containing text events in the English language. We used a dataset of video frames aggregated with ground truth information that allowed the evaluation of recognition and detection tasks. For the recognition of text regions, we used state-of-the-art deep learning-based object detectors and fine-tuned them to recognize text in multiple scripts. In a classification framework, the script of recognized text regions is recognized using CNNs. We used the English Language Net, a combination of CNN and bidirectional LSTMs, which reports high recognition rates for challenging video text in the English language.

In our further work on this problem, we want to develop a converter of the extracted text into the required language, which saves it in a text file and also gives a voice output, the extracted text is stored in a database.

**Results**