TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PASHCHIMANCHAL ENGINEERING CAMPUS

A
FINAL YEAR PROJECT REPORT
ON

"iListen: Smart Reader for Visually Impaired"

Submitted To
Department of Electronics and Computer Engineering

Submitted By:
Santosh Poudel (BEX 071-415)
Kshitiz Aryal (BEX 071-420)
Prakriti Pahari (BEX 071-424)

CERTIFICATION

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled " iListen: Smart Reader for Visually Impaired" submitted by **Prakriti Pahari**, **Kshitiz Aryal** and **Santosh Poudel** in partial fulfilment of the requirements for the Bachelor's degree in Electronics and Communication Engineering.

…………………………………………………….

Supervisor,
Er. Sharan Thapa,
Assistant Professor,
Department of Electronics and
Computer Engineering

…………………………………………………….

External Examiner,
Er. Ram Sharan Baral

Engineer at Nepal Telecom

…………………………………………………….

Head of Department,

Hari Prasad Baral,

Department of Electronics and Computer Engineering

COPYRIGHT

Head of Department,
Department of Electronics and Computer Engineering,
Pashchimanchal Campus,
Lamachaur-16, Pokhara,
Nepal

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the major project undertaken during 4th year Electronics and Communication Engineering. We owe special debt of gratitude to the supervisor of our project, Er. Sharan Thapa for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us.

Our deep sense of gratitude is accorded to HOD of Electronics and Computer Department, Er. Hari Prasad Baral for his encouragement and motivation for our project work. We also wish to express our sincere thanks to Er. Hari KC, Er. Homnath Tiwari, Er. Laxmi Pd. Bastola for the enthusiasm they transmitted, as well as for the richness of their guidelines and invaluable suggestions throughout the project. Our thanks and appreciations also goes to Er. Aashish Adhikari and Er. Prakash Pandey for their supervision in developing the project and helping us out with their abilities.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members and lab-Instructors of the department for their kind assistance and cooperation during the development of our project. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in completing this project as well as in preparation of this report.

Any suggestion for further improvement of our project are hearty welcomed.

ABSTRACT

According to World Health Organization (WHO), an estimated 253 million people live with vision impairment: 36 million are blind and 217 million have moderate to severe vision impairment worldwide among which 90% people live in developing countries. Written document is an appearing form of information which is unapproachable by a lot of visually impaired people except it is symbolized in a non-visual form like Braille. The fact that most of the visually impaired are not enlightened with the knowledge of Braille is still unknown. For this, the use of Smart Reading devices is essential. Although there are many available solutions to assist them to read, however none of them provide a reading experience similarly as that of sighted population. In particular, there is the need of portable, affordable and reliable text reader for blind community. Our project proposes the smart reader for the virtually challenged people using Raspberry Pi which addresses the integration of complete Text Read-out system. This system captures the page of printed text through the webcam interfaced with raspberry pi. The OCR (Optical character Recognition) installed in Raspberry pi scans the captured image into a digital document which is then pre-processed using skew correction, segmentation, etc. which is now subjected to feature extraction for classification. Once classification is completed, the text is converted to speech by a Text-To-Speech conversion unit kept in Raspberry pi. In a nutshell, this system converts the text written in paper to the editable format and which is then read out as speech.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

AI: Artificial Intelligence

OCR: Optical Character Recognition

NLP: Natural Language Processing

TTS: Text to Speech

OpenCV: Open Source Computer Vision

SMPS: Switch Mode Power Supply

USB: Universal Serial Bus

GPIO: General Purpose Input Output

SOC: System On Chip

LAN: Local Area Network

HDMI: High-Definition Multimedia Interface

AMBA: Advanced Microcontroller Bus Architecture

DPI: Dots Per Inch

RGB: Red, Green and Blue

OpenGL: Open Graphics Library

CNN: Convolutional Neural Network

PWM: Pulse Width Modulation

VI: Visually Impaired

CHAPTER ONE: INTRODUCTION

1.1 BACKGROUND

In today’s world, we have seen a huge demand of storing texts of books, newspapers, etc. into digital formats. An existing way of doing this is by scanning the desired text, which will be stored as an image that won't help for further processing. This means if we store the scanned text image, we cannot further edit or modify the content; neither can we read the text, line by line or word by word. It shows the need of Optical Character Recognition (OCR).

Scanned documents help us archive stacks of written documents into folder, occupying less storage and making it easier to organize, edit, and copy. However, the process of finding the required content inside hundreds of scanned documents is troublesome. Opening each of them and read it by ourselves can eat up a lot of time as well. For this, we could let our smart device to turn the scanned image into text and searching or editing our scanned documents as easily as we search through other editable text documents. This is what OCR (Optical Character Recognition) does. It uses our computer's intelligence to recognize letter shapes in an image or scanned document, and turn them into digital text we can copy and edit as needed.

Optical character recognition (OCR) is the conversion of captured images of printed text into machine encoded text. It is a process that associates a symbolic meaning with objects (letters, symbols an number) with the image of a character. Optical Character Recognition is also proved to be significant for visually impaired people who cannot read paper texts, but need to access the content of the Text documents. OCR is used to digitize the texts that have been produced with non-computerized system which also reduces storage space. It is very much time consuming and labor intensive to edit and reprint the text document on paper. So, OCR makes it possible to apply techniques such as machine translation, text-to-speech and text mining to the capture / scanned page and help in efficient conversion.

On the other part, 285 million are visually impaired out of whom 39 million people are completely blind, i.e. have no vision at all, and 246 million have mild or severe visual impairment (WHO, 2011). It has been predicted that by the year 2020, these numbers will rise to 75 million blind and 200 million people with visual impairment [1]. There have been numerous discoveries in this area to help visually impaired to read without much difficulties. The existing technologies use a similar approach as our system, but they have certain drawbacks. Firstly, the input images taken in previous works have no complex background, i.e. the test inputs are printed on a plain white sheet. It is easy to convert such images to text without pre-processing, but such an approach will not be useful in a real-time system.[2][3][4] Visually impaired people face numerous difficulties while reading printed texts using existing technologies, including problems with alignment, focus, accuracy, mobility and efficiency.

In our project, we present a fully integrated camera based device on Raspberry Pi board that assists the visually impaired which effectively and efficiently reads paper-printed text. This is a text read out system for the visually challenged which has a camera as an input device to feed the printed text document for digitization and the scanned document is processed by a software module called the OCR (optical character recognition engine). As part of the software development, the Open CV (Open source Computer Vision) libraries is utilized for image capturing of text, pre-processing and character segmentation while Tensorflow model is used to train and recognize the characters. The edited text document is fed to the output devices (i.e. headset) which is connected to the raspberry pi board or a speaker which can spell out the text document aloud.

Most of these technologies built for visually impaired people are built on the two basic building blocks of OCR software and Text-to-Speech (TTS) engines.

1.2 PROBLEM STATEMENT

Books and paper documents are the real sources of knowledge. But this knowledge can only be accessed by people with clear vision. However, our society includes a number of people who are visually challenged. For this group of population, world is like a black illusion. This shows that acquiring the knowledge by reading documents for visually impaired people is cumbersome. According to the survey, the population of blind is shown below:

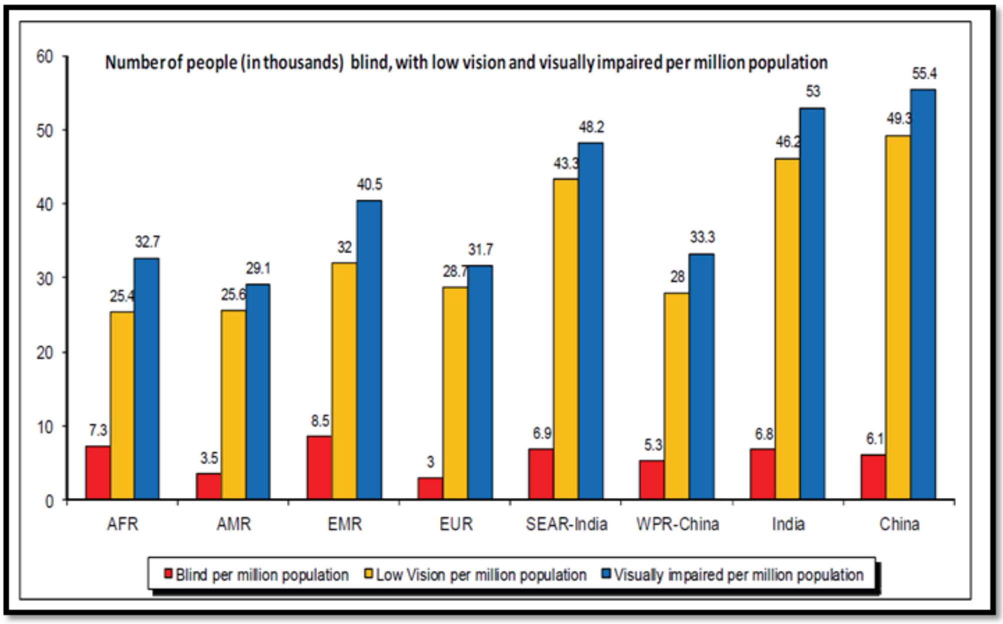


Figure 1. Survey of Visually Impaired Population

Table 1: Prevalence of blindness (per thousand) as per the estimate of 2015

Types of Blindness	Urban	Rural	Total
Total Blindness	4.43	5.99	5.40
Economic Blindness	11.14	15.44	13.83
One eye blindness	8.23	7.00	7.46

Blindness = Visual acuity less than 3/60 in better eye with spectacle correction

Economic blindness = Visual acuity less than 6/60 in the better eye with spectacle correction

One eye blindness = Visual acuity less than 3/60 in one eye and better than 6/60 in the other eye with spectacle correction

Braille is one of the methods which is used to read a book or document. In this method, any document has to be converted to braille format to become understandable to a blind. The problem arises due to the fact that, this is an expensive procedure and many times not available easily. Other solutions for Smart Reader were also discovered however, they only support the black colored texts with white background. They also did not produce the accurate results due to the errors in image capturing and image quality degradation. The solution is rather simple, introduce a smart device with a multimodal system that can convert any document to the interpreted form to a blind accompanied with pre-processing operations for accurate results.

1.3 OBJECTIVES

The project is expected to fulfill the following objective on the completion of this project:

General Objective:

a) Improving the quality of captured image and its orientation

b) Detecting the text in image followed by audio output

Specific Objective:

To help the visually impaired in reading texts from paper documents in English and Nepali Language

CHAPTER TWO: LITERATURE REVIEW

Ray Kurzweil et.al [5] proposes A K-Reader Mobile number of portable reading assistants are designed specifically for the visually impaired. “K-Reader Mobile” is a mobile application which allows the user to read mail, receipts, fliers, and many other document.But these systems/ devices fail to give an economic solution to the problem and are available on specific platforms.

AthiraPanicker et.al [6] proposes smart shopping assistant label reading system with voice output for blind using raspberry pi.— This project aims at The document to be read must be nearly flat, placed on a clear, dark surface and contain mostly black text printed on white background and it does not read from complex backgrounds.

MarutTripathi et.al [7] proposes A Navigation System for blind people to navigate safely and quickly, in the system obstacle detection and recognition is done through ultrasonic sensors and USB camera. This system detects the obstacles up to 300 cm via ultrasonic sensors and sends feedback in the form of beep sound via earphone to inform the person about the obstacle.

DimitriosDakopoulos et.al [8] proposes A Wearable Obstacle Avoidance Electronic Travel Aids for Blind that presents a comparative survey among portable/wearable obstacle detection/avoidance systems (a subcategory of ETAs) in an effort to inform the research community and users about the capabilities of these systems and about the progress in assistive technology for visually impaired people.

X.Chen et.al [9] proposes Automatic detection and recognition of signs from natural scenes we present an approach to automatic detection and recognition of signs from natural scenes and its application to a sign translation task. We have applied the approach in developing a Chinese sign translation system, which can automatically detect and recognize Chinese signs as input from a camera, and translate the recognized text into English but its only work in Chinese language.

William A. Ainsworth [10] proposes a system for converting English text into speech the feasibility of converting English text into speech using an inexpensive computer and a small amount of stored data has been investigated but it’s not suitable for all memory range of computers.

ZoranZivkovic et.al [11] proposes Improved Adaptive Gaussian Mixture Model for Background Subtraction Background subtraction is a common computer vision task. We analyze the usual pixel-level approach. We develop an efficient adaptive algorithm using Gaussian mixture probability density. Recursive equations are used to constantly update the parameters and but also to simultaneously select the appropriate number of components for each pixel.

S. B. Shrote et.al [12] proposes Assistive Translator for Deaf & Dumb People Communications between deaf-mute and a normal person have always been a challenging task. The project aims to facilitate people by means of a glove-based deaf-mute communication interpreter system.

Michael McEnancy et.al [13] Finger Reader Is audio reading gadget for Index Finger We present a finger-worn device that assists the visually impaired with effectively and efficiently reading the paper-printed text. But the blind people can't aim the letters accurately.

Vasanthi.G et.al [14] proposes Vision Based Assistive System for Label Detection with Voice Output. A camera-based assistive text reading framework to help blind persons read text labels and product packaging from a hand-held object in their daily resides are proposed

CHAPTER THREE: METHODOLOGY

3.1. Block Diagram of Proposed Method:

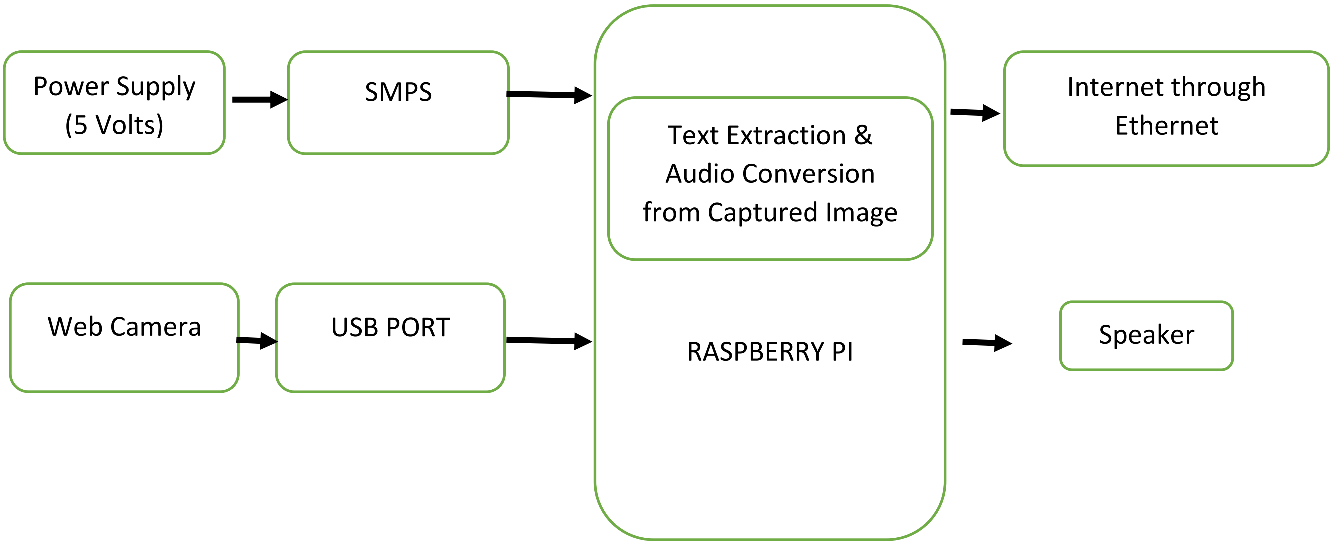

Power Supply (5 Volts)
SMPS
Web Camera
USB PORT
Text Extraction & Audio Conversion from Captured Image
RASPBERRY PI
Internet through Ethernet
Speaker

Figure 2: Block diagram of Proposed System

The framework of the proposed project is the raspberry pi board. The raspberry pi B+ is a single board computer having 4 USB ports, an Ethernet port for internet connection, 40 GPIO pins for input/ output, CSI camera interface, HDMI port, DSI display interface, SOC (system on a chip), LAN controller, SD card slot, audio jack, and RCA video socket and 5V micro USB connector.

The power supply is given to the 5V micro USB connector of raspberry pi through the Switched Mode Power Supply (SMPS). The SMPS converts the 230V AC supply to 5V DC. The web camera is connected to the USB port of raspberry pi. The raspberry pi should have an OS installed named RASPBIAN which process the total conversions. The audio output is taken from the audio jack of the raspberry pi. The converted speech output is amplified using an audio amplifier (i.e. Speaker). The page to be read is placed on a base and the camera is focused to capture the image. The captured image is processed by the OCR software using Python and Open CV library installed in raspberry pi. The captured image is converted to text by the recognition function which is built on Tensorflow and is previously trained from 12000 sample images. The text is converted into speech by the TTS engine. The final output is given to the audio amplifier from which it is connected to the speaker.

3.2. Architecture of the System:

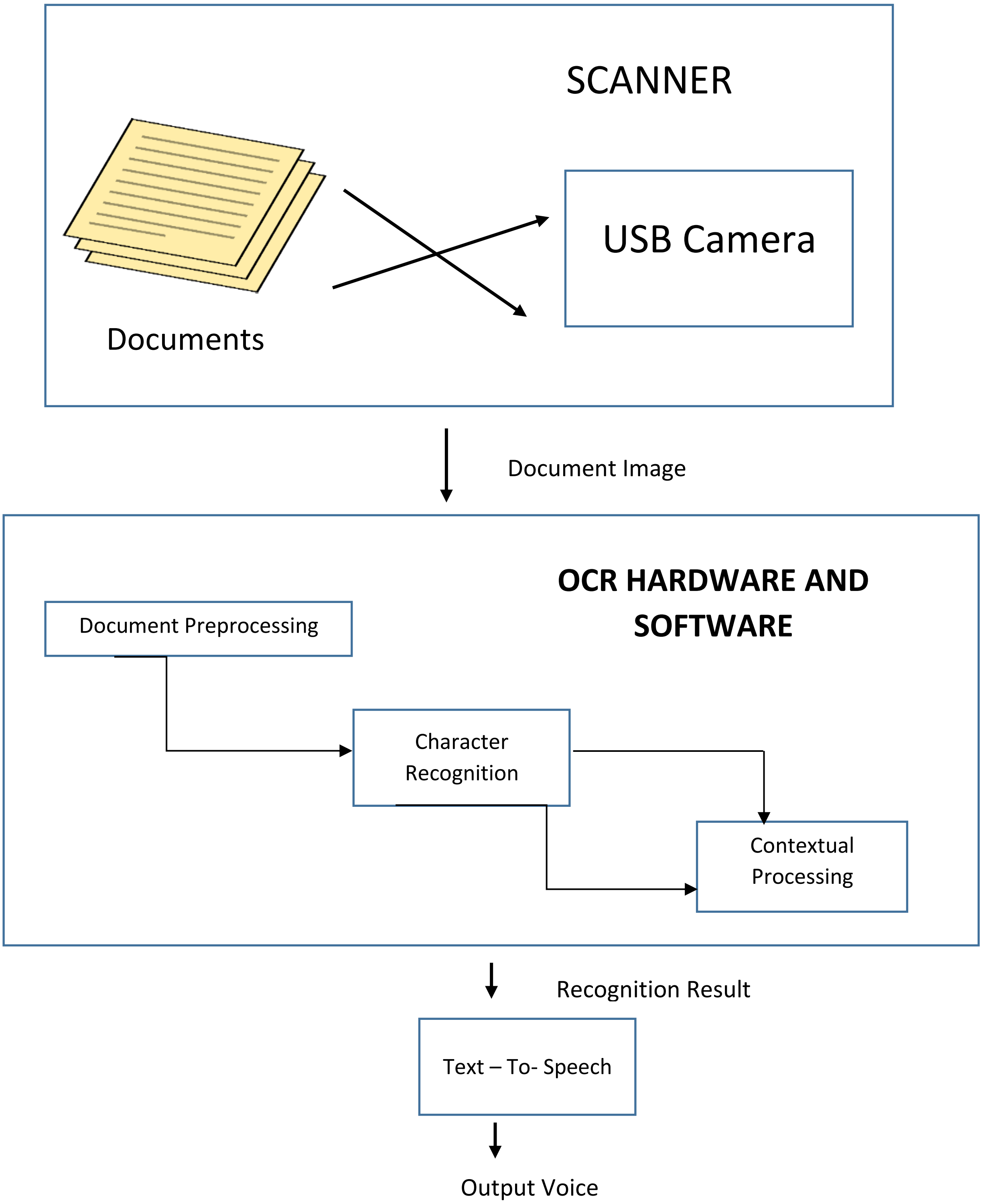


Figure 3: Architecture of iListen System

3.3. System Specifications:

3.3.1. Hardware Specifications:

3.3.1.1. Raspberry Pi

Raspberry pi is a SoC (System on Chip), that integrates several functional components into a single chip or chipset. The SoC used in Raspberry Pi 2 is the Broadcom BCM2836 SoC Multimedia processor. The CPU of the Raspberry Pi contains an ARM Cortex-A7 900MHz processor which makes use of the RISC Architecture and low power draw. It is not compatible with traditional PC software. Therefore it has to be connected to a monitor separately and thus it is called as a mini computer. Raspberry pi has an on-chip DSP processor which is used to perform the floating point operations. The raspberry pi uses AMBA (Advanced Microcontroller Bus Architecture) which is an on-chip interconnect specification for the connection and management of functional blocks in system-on-chip (SoC) designs. It facilitates development of multi-processor designs with large numbers of controllers and peripherals. The GPIO pins of the Pi differ by the model. There are 40 pins, out of which there are 4 power pins and 8 ground pins. Rest of the pins is used as GPIO's. The networking capabilities of the Pi can be used as a wired Ethernet (IEEE 802.3) or the wireless IEEE 802.11 Wi-Fi. Raspberry pi has an internal memory of 1GB RAM and external memory is extendable up to 64GB. HD webcam or raspberry pi camera which has a 5MP HD camera with a resolution of 1920x1200 can be used to capture the images. The speech output is given through the earphones connected to the raspberry pi's 3.5mm audio port.

Raspberry PI parts	**Specifications**
Chip	Broadcom BCM2837 SoC
CPU	ARM11
Memory	1.2 GHZ Low Power ARM1176JZFS Applications Processor - Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode - Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Operating System	Boots from Micro SD card, running a version of the Linux operating system
Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4) - Composite RCA (PAL and NTSC)
Audio Output	3.5mm jack, HDMI
USB	4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip - Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply line

Table 2: Raspberry PI & its Classifications

3.3.1.2. USB Camera:

USB Cameras are imaging cameras that use USB 2.0 or USB 3.0 technology to transfer image data. USB Cameras are designed to easily interface with dedicated computer systems by using the same USB technology that is found on most computers. The accessibility of USB technology in computer systems as well as the 480 Mb/s transfer rate of USB 2.0 makes USB Cameras ideal for many imaging applications. USB Camera used in the project captures the image of book pages and sends to Raspberry pi through USB connection. Once the button is pressed, camera takes the picture and processing tasks begins at raspberry.

3.3.2. Software Specifications:

Operating System: Raspbian (Debian)
Raspbian is the operating system for normal use on a Raspberry pi. Raspbian is a free operating system based on Debian, optimized for the Raspberry Pi hardware. Raspbian comes with over 35,000 packages, precompiled software bundled in a nice format for easy installation in Raspberry Pi. Raspbian is loaded into the SD card of Raspberry Pi.

Language: Python3

Platform:

- OpenCV: OpenCV, which stands for Open Source Computer Vision, is a library of functions that are used for real-time applications like image processing, and many others. [14] Open CV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, Open CV makes it easy for businesses to utilize and modify the code.
- Tensorflow: Tensorflow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

Library: OCR engine, TTS engine

3.4. MECHANISM

3.4.1 Image Acquisition:

In this step the device is moved over the paper with text and its image is captured using Raspberry Pi camera or an HD webcam with high resolution. The image quality must be high so as to have accurate and clear recognition due to the high-resolution camera. We are using the HD WebCam 4.8MP camera. This acquired image is then applied to the image pre-processing step for reduction of unwanted noise.

3.4.2 Image Preprocessing:

This step consists of rescaling, color to gray scale conversion, edge detection, noise removal, warping and cropping and thresholding. The image is converted to gray scale as many OpenCV functions require the input parameter as a gray scale image. Noise removal is done using bilateral filter. Canny edge detection is performed on the gray scale image for better detection of the contours. The warping and cropping of the image are performed according to the contours. This enables us to detect and extract only that region which contains text and removes the unwanted background. In the end, Thresholding is done so that the image looks like a scanned document. This is done to allow the OCR to efficiently convert the image to text.

3.4.2.1. Rescaling:

Rescaling refers to resizing the image. The size of the image can be specified manually, or you can specify the scaling factor. Our project works best on images that are 300 dpi, or more. If we're working with images that have a DPI of less than 300 dpi, we consider rescaling. Otherwise, rescaling is not essential. It is preferred that the images are at least 300 dpi, rather than rescaling them later in the pipeline.

Different interpolation methods are used. The images that are rescaled are either shrunk or enlarged. If the scaling is done to shrink the image, **INTER_AREA** is used. On the other hand, as in most cases, we may need to scale our image to a larger size to recognize small characters. In this case, **INTER_CUBIC** generally performs better than other alternatives, though it's also slower than others. If we prefer to trade off some of our image quality for faster performance, we use **INTER_LINEAR** for enlarging images.

3.4.2.2. Noise Removal:

Most of the text documents on paper are likely to experience noise to some extent. Though the reasons for this noise may vary, it is hard to recognize characters of the text. This noise on images can be removed by using a few techniques combined together. These includes converting image to grayscale, dilation, erosion and filtering.

Dilation, erosion and filtering require a kernel matrix to work with. Larger the kernel size is, the wider the region this method gets to work on. And again, there's not a fixed kernel size value that fits all. We use hit and trial method until we find the right values for our images. However, a good rule of thumb could be starting with small kernel values for small fonts. Similarly, for larger fonts, we may experiment with larger kernels.

A. Image to Grayscale:

Transformations within RGB space like adding/removing the alpha channel, reversing the channel order, conversion to/from 16-bit RGB color (R5:G6:B5 or R5:G5:B5), as well as conversion to/from grayscale using:

RGB [A] to Gray: $Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$

and

Gray to RGB [A]: $R \leftarrow Y, G \leftarrow Y, B \leftarrow Y, A \leftarrow \max(ChannelRange)$

B. Filtering:

Image filtering is usually achieved by convolving the image with a low-pass filter kernel. While filters are usually used to blur the image or to reduce noise, there are a few differences between them.

1. Averaging

After convolving an image with a normalized box filter, this simply takes the average of all the pixels under the kernel area and replaces the central element.

2. Gaussian blurring

This works in a similar fashion to Averaging, but it uses Gaussian kernel, instead of a normalized box filter, for convolution. Here, the dimensions of the kernel and standard deviations in both directions can be determined independently. Gaussian blurring is very useful for removing Gaussian noise from the image. On the contrary, Gaussian blurring does not preserve the edges in the input. We use this type of blurring in our project for better results.

3. Median blurring

The central element in the kernel area is replaced with the median of all the pixels under the kernel. Particularly, this outperforms other blurring methods in removing salt-and-pepper noise in the images.

Median blurring is a non-linear filter. Unlike linear filters, median blurring replaces the pixel values with the median value available in the neighborhood values. So, median blurring preserves edges as the median value must be the value of one of neighboring pixels.

4. Bilateral filtering

Speaking of keeping edges sharp, bilateral filtering is quite useful for removing the noise without smoothing the edges. Similar to Gaussian blurring, bilateral filtering also uses a Gaussian filter to find the Gaussian weighted average in the neighborhood. However, it also takes pixel difference into account while blurring the nearby pixels.

Thus, it ensures only those pixels with similar intensity to the central pixel are blurred, whereas the pixels with distinct pixel values are not blurred. In doing so, the edges that have larger intensity variation, so-called edges, are preserved.

3.4.2.3. Binarization:

This is the most essential step in image preprocessing. In this cyber-era where everything eventually comes down to 1’s and 0’s, converting image to black and white immensely helps our system to recognize characters more precisely. There are various image thresholding methods that fits all types of documents.

1. Simple Threshold

“Things are not always black and white”. For a simple threshold, things are pretty straight-forward.

First, you pick a threshold value, say 127. If the pixel value is greater than the threshold, it becomes black. If less, it becomes white.

2. Adaptive Threshold

In the previous method, we used a global value as threshold value. But it may not be good in all the conditions where image has different lighting conditions in different areas. In that case, we go for adaptive thresholding. Rather than setting a one global threshold value, we let the algorithm calculate the threshold for small regions of the image. Thus, we end up having various threshold values for different regions of the image for the images with varying illumination, which is the flexible way of binarization.

There are two adaptive methods for calculating the threshold value. While **Adaptive Thresh Mean** returns the mean of the neighborhood area, **Adaptive Gaussian Mean** calculates the weighted sum of the neighborhood values.

3. Otsu's Threshold

This method particularly works well with **bimodal images**, which is an image whose histogram has two peaks. If this is the case, we might be keen on picking a threshold value between these peaks. This is what Otsu's Binarization actually does. It's pretty useful for some cases while it may fail to binarize images that are not bimodal.

3.4.3 Clustering and Segmentation

Each language are composed of basic syntactic rules like words. One of the basic rules for words is words are the combination of characters without any spaces between them. Though spaces are present between the words. Clustering algorithms is used to identify the words from the individual characters that are extracted. Features on which the clustering is based are spacing between the characters and orientation of black and white images. Usually threshold spacing is used to cluster the characters to obtain the words. This is the most crucial step to obtain the word present in the picture.

Segmentation of objects from an image is a very important task in character recognition. The two most popular segmentation techniques are Mean shift segmentation and efficient graph based segmentation. Mean shift segmentation is done based on filtration of data followed by the clustering of data. This clustering would result in the segmentation of objects from an image. The latter method, efficient graph based segmentation works directly on data points without performing filtering step. Adaptive thresholding and single linkage clustering becomes the key to the success of this method. The characteristics that were considered to check the efficiency are correctness and stability with respect to parameter and image choice. In all the characteristics, mean shift algorithm performed significantly better. There is one more algorithm designed, called hybrid algorithm, which is the amalgamation of mean shift filtering followed by graph based segmentation.

3.4.3.1 Contour detection

The extraction of character from an image becomes easy if the boundaries of the character can be found through some algorithm. The boundaries of a character is determined in many ways depending on many properties like the edges of the character, intensity of the text present in the data. One such method is Contour detection. This method will be applied on an image after detecting the edges of the character. To start with, the first pixel of a character needs to be identified. This first pixel can be present at any point of the character. This striking of position of first pixel depends on the way we track the input image to extract characters. By default, as the origin of an image lies at the top left corner, the top most pixel of the character will be encountered. Once the first pixel is identified, the other pixels are identified by following the path along the edge from the first identified pixel. This helps to track along the edge of a character. Based on the fact that the edge detection gives closed path, there is always a chance that we reach the first identified pixel after visiting all other pixels along the edge. This helps as a breaking condition for identifying the boundaries of the tracked character. Once the boundaries of the character are known in the source image, it can be extracted and used for further references. The below image displays the origin of an image and the arrow in the image denotes the first pixel visited during the scan of the total image. This approach applies for all the characters in the image.

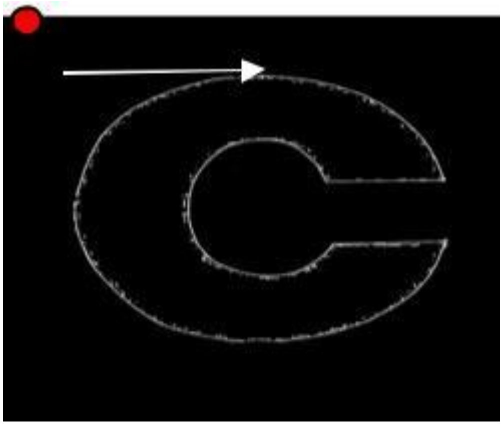


Figure 4 Letter 'C' with the origin of the image represented in red circle and arrow indicates the first pixel that is visited.

For any given image, first image is converted into binary image using the threshold. Once the binary image is obtained, then edges are obtained for the given image. The edges of each object are found along with the inner data. Now the inner data increases along with the thickness of the object and the rate of finding the data decreases along with the increase of thickness of the data. Hence, contours are more useful in these types of applications which are always closed and curved and help in finding the boundaries of each and every object present in the given character. The contours help in identifying only the boundary pixels of each and every object present in the image. Now the scanning of the whole image takes place and the boundaries of each and every pixel is found. These boundaries are used to extract the characters from the image. Now each boundaries are converted into the image of single characters and passed through following procedures before being compared with templates of trained data.

3.4.3.2 Padding Images

As mentioned above, the obtained contour consists of images of single words only. Moreover, the images are of different sizes because different words are of different lengths and heights. For instance the image of the word ‘error’ has a lower width than the image of the word ‘congratulations’ because of the length of the words. Similarly, the image heights differed among images due to the heights of their characters. For instance, the image of the word car has a lower height than the image of the buy since the characters of the word ‘buy’ extend above and below with ‘b’ and ‘y’. Our architectures, however, assumed the input images to be of the same size just like any other convolutional neural network architecture. This is essential as the weights of the layers are adjusted according to the first input image, and the model would not work as well if weights were not consistent, or changed shapes, for different inputs. Thus, we decided to make all the images of the same shape. It did not make sense to crop large images to an average size because cropping removes some part of characters from the image, which in turn can cause the image of a word to be interpreted differently. For instance cropping the image of the word ‘scholarly’ can make the image look like it is the image of the word scholar, and having two different labels for two similar images would definitely negatively affect the accuracy of our models. Moreover, we also thought that rescaling the image size would not work because of a few reasons. First of all, the aspect ratio of every image is different. This means that if we wanted to set the height or the width of an image to a specific value, the other dimension would have been different for all the other images with different aspect ratios. Secondly, if we resized the image on the height and width dimensions independently, then the image would get distorted, and we thought that this again would negatively affect our model since the inherent characteristics of the picture are being lost. Therefore, we decided to pad our images with whitespace to the maximum width and height as present in our dataset. While doing so, the white space was added evenly on both sides of the height and the width dimensions. This approach does not necessarily change the inherent characteristic of the word image, and since the images of the same words would pretty much be padded with similar sizes, the relative relationship between the images do not change.

3.4.3.3 Zero-centering Data

$$\mathrm{X}^{(\mathrm{K})}{}_{\mathrm{centered}} = \mathrm{X}^{(\mathrm{k})} - \left(\frac{1}{N}\right) \sum_{k=1}^{N} \frac{1}{H*W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_{i,j}^{(k)}$$

We center our data by subtracting the dataset mean pixel values from the pixel values of the images before training. This is essential because we use one single learning rate when we update our weights, we want to have a relative scale of the weights among each other so that when we multiply with the single learning rate, all the weights are affected by this multiplication in the same relative manner. If we had not done centering, some weights would be hugely amplified at every update, and others would stay very low

3.4.4 Training and classification

3.4.4.1 Datasets

Datasets for training the network is downloaded from Kaggle which is a repository for huge amount of data for analysis purpose. 200 images for each class of characters including small and capital alphabets were obtained. Images are handwritten samples of each letter and can be increased in number to increase the accuracy with variation. 90% of dataset images were used for training purpose and 10% was used for testing the trained model.

3.4.4.2 Training Data

TensorFlow library is used to train and build the machine learning model. It is an open source deep learning library which is provided by Google. Numerical computations are done using data flow graphs. The nodes present in the graph represent mathematical operations while the edges represent the tensors between them. Tensors are the data that move between the nodes which actually are multi-dimensional arrays consisting of real values. TensorFlow uses Convolutional Neural Network (CNN) for image recognition due to which the process of feature extraction is eliminated. Four separate models are trained for improving performance. 1st for ascender, 2nd for descender, 3rd for core and compound letters and 4th for symbols and numbers. Accuracy of MobileNet model and Inception model are compared in table below. Trained data is saved in graph file.

Number of iterations	**MobileNet model (in %)**	**Inception model (in %)**
500	96.0	85.5
1000	96.4	87.6
1500	97.5	89.3
2000	96.8	91.4
2500	97.5	92.4
3000	97.5	92.1
3500	97.5	92.8
4000	97.5	93.4

Table 3 Inception VS Mobilenet over Different Iterations

TensorFlow library also provides a feature called TensorBoard to generate graph on trained models. Below figure shows graphs generated for Inception and MobileNet model respectively over four thousand iterations.

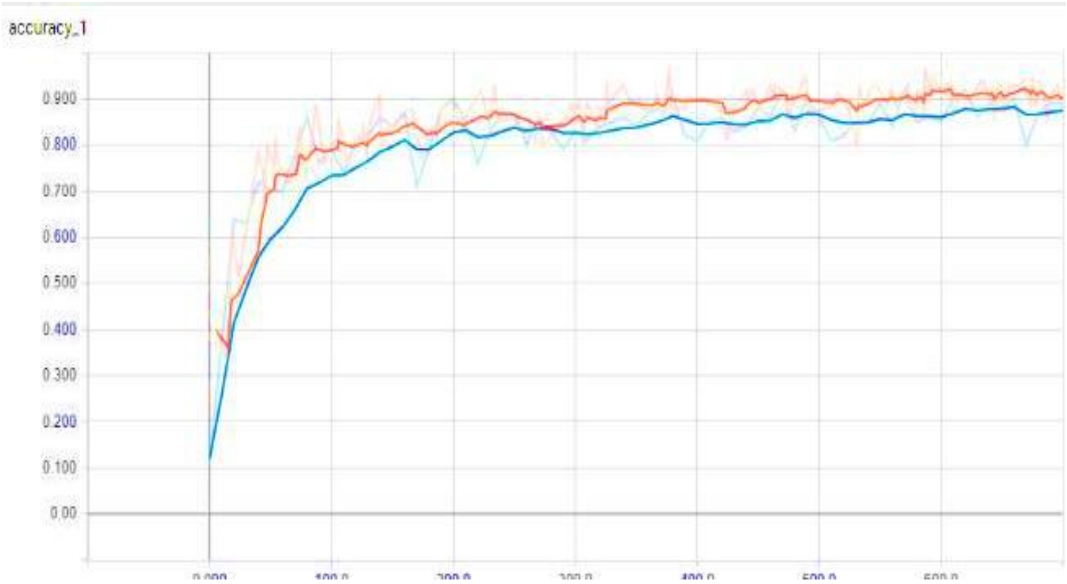


Figure 5: Accuracy of Inception Model over 4000 iterations

Figure 5 Accuracy of Inception Model over 4000 iterations

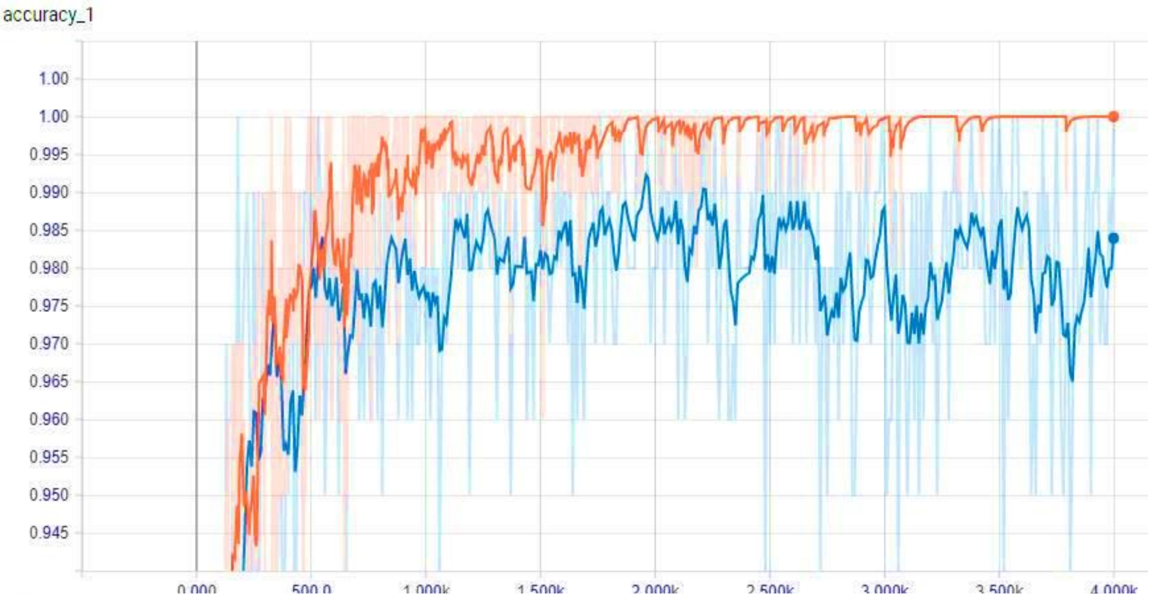


Figure 6 Accuracy of MobileNet over 4000 iteration

3.4.4.3 Classification

Classification is based on MobileNet training. MobileNet is a computer vision model. It is designed for devices having restricted resources that are small, low-power, low-latency models. MobileNet effectively maximizes accuracy. This model predicts the correct answer as one of its top 5 guesses. The one with highest confidence is taken as the output. If presence of ascender-or kana is detected then it can be core letter or compound letter. If not then it can be core or compound letter with or without descender, or it can be symbol.

3.5. Flow of Process:

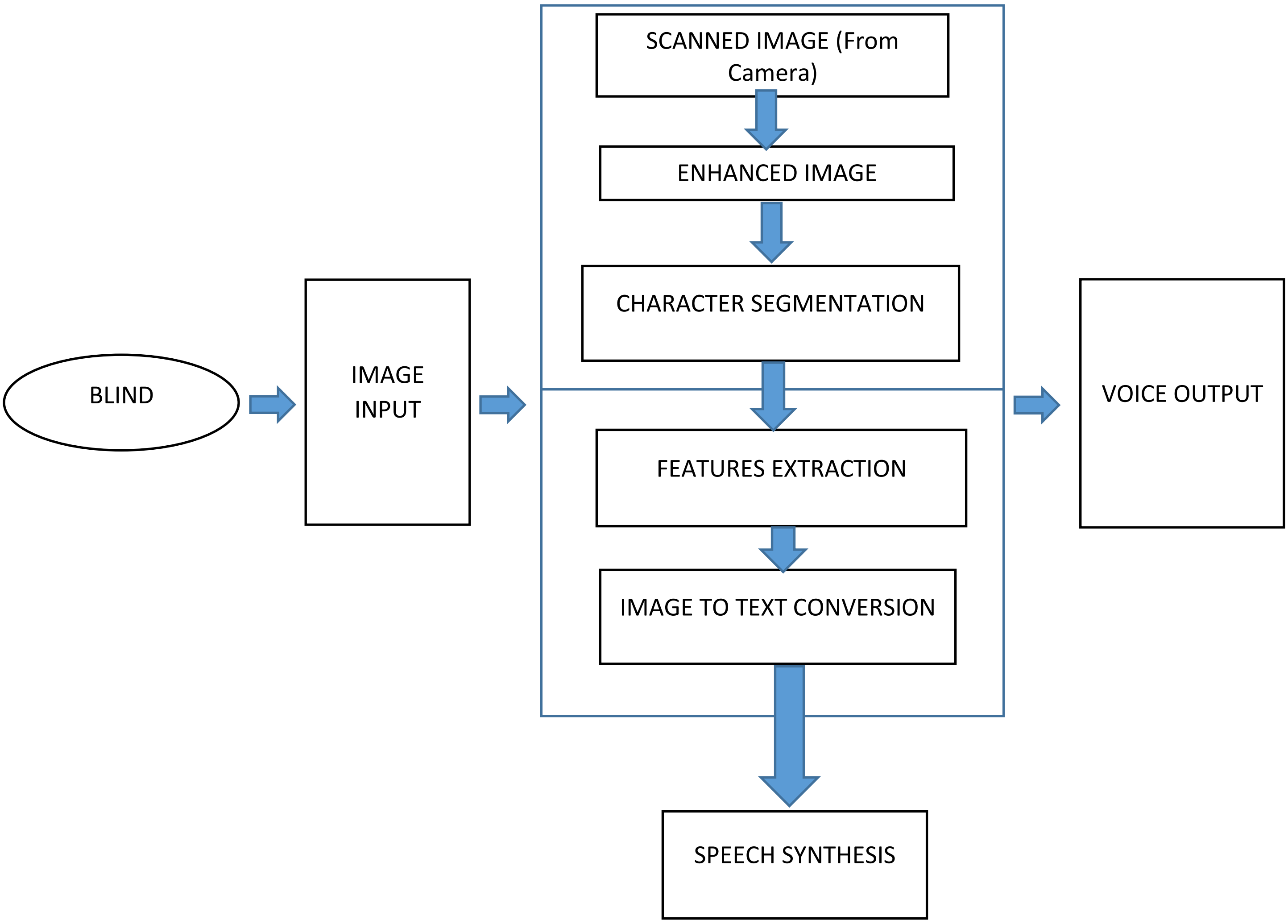


Figure 7 Flow of Process

-The first step in which the device is moved over the printed page and the inbuilt camera captures the images of the text. The quality of the image captured will be high so as to have fast and clear recognition due to the high resolution camera.

OCR (Optical Character Recognition) Process

- Step-1

Loading the image file: In order for OCR to be effective, it must support a wide array of file formats, including PDF, BMP, TIFF, JPEG, and PNG files. Once the file is loaded, the software can begin to work. These files can be scanned documents, photographs, or even read-only files. Regardless of the original format, OCR software will transform these files into easily accessible & editable data.

- Step-2

Improving image quality and orientation: Depending on the method in which the image file was created, there are a number of issues that may arise. More often than not, an image file will be skewed or contain “noise” (varying brightness or color). In this stage of OCR, the software will work to de-skew, remove any “noise”, and improve the overall quality of the images. This is a critical step – as blurry or skewed images are not interpreted properly.

- Step-3

Analyzing the page: During this stage of Optical Character Recognition, the layout of the original file is noted and processed. This includes the detection of text positions, white space, and the prioritization of important text areas or sections. Each characters are identified in the form of contours. Comparing the position of contours its positions are aligned in order.

- Step-4

Detecting words and lines of text: This is the beginning stage of actual character recognition. The created image files of each contour containing single characters are compared with trained graphs and then recognized.

-Step-5

Analyzing and fixing of “broken” or “merged” characters: Depending on the quality of the original file, there are often errors in which characters are broken or blurred together. The OCR software must now break down and resolve these errors in order to properly interpret the appropriate characters.

-Step-6

Saving the file: After the file has been fully interpreted, it can be saved to our desired file format. While there is much more to OCR software, these 8 steps make up the primary processes involved in Optical Character Recognition.

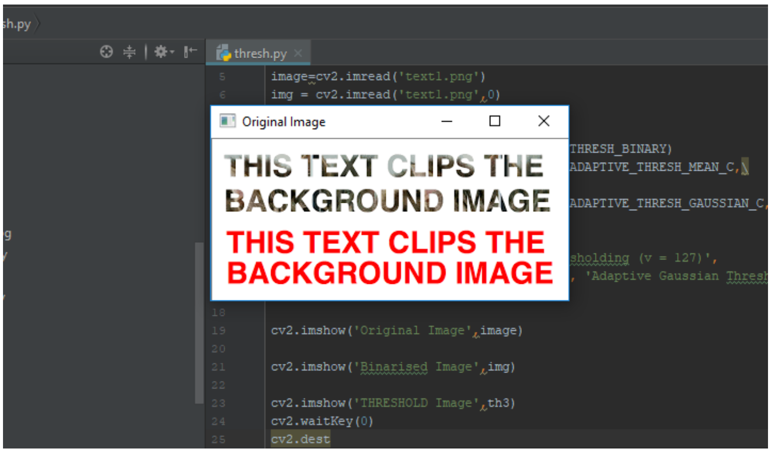
- **Text to speech:** The scope of this module is initiated with the conclusion of the receding module of Character Recognition. The module performs the task of conversion of the transformed English/Nepali text to audible form. The Raspberry Pi has an on-board audio jack, the on-board audio is generated by a PWM output and is minimally filtered. A USB audio card can greatly improve the sound quality and volume.

CHAPTER FOUR: RESULTS AND ANALYSIS

We ran the program using a full database of trained images trained on 400 images of each characters of English Alphabets. After running a few tests, the recognition success rate is about 90%.

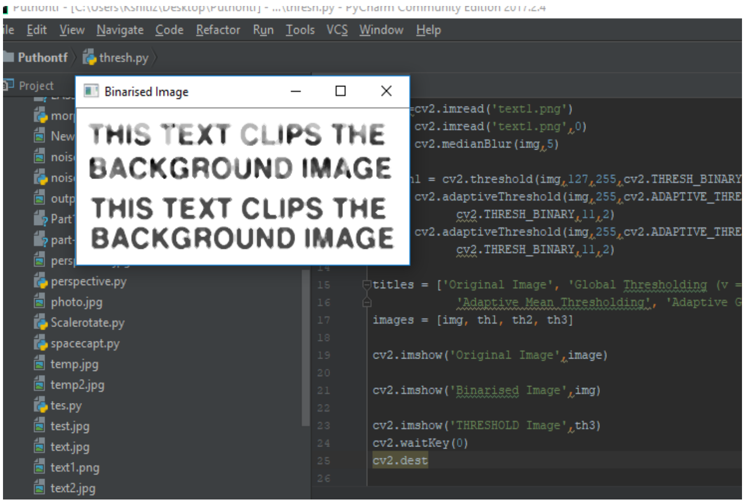
When the recognition phase completes, the recognized text is successfully converted into a text file. The text file undergoes through the TTS Engine and the audio file is generated which is the final output of our project.

The obtained output images after pre-processing are displayed below. Figure 8 shows the original image. Figure 2 to Figure 4 display the pre-processing done in each stage. And finally Figure 4 represents the image which is given as input to the OCR testing process. Figure 5 displays the text obtained at the output of the OCR engine. It is evident that the result is not completely accurate. This is because of the less resolution of the camera used. Better results can be obtained if the camera used is a High definition camera.



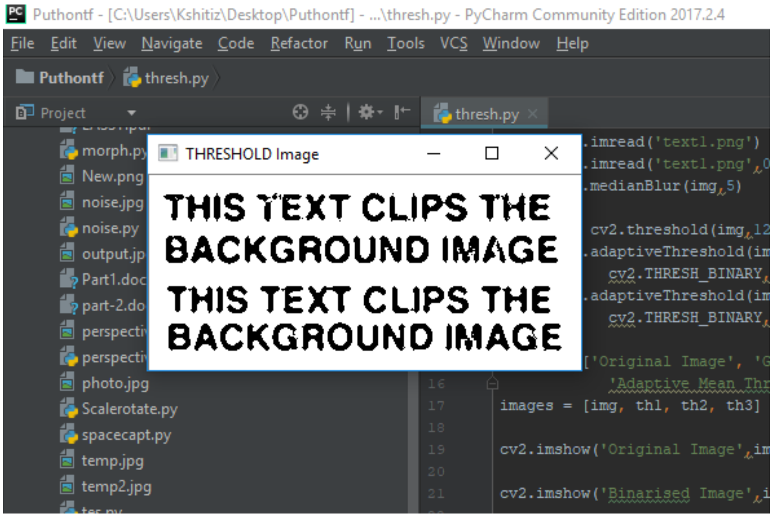
```
image=cv2.imread('text1.png')
img = cv2.imread('text1.png',0)

THRESH_BINARY)
ADAPTIVE_THRESH_MEAN_C,\

ADAPTIVE_THRESH_GAUSSIAN_C,

sholding (v = 127)',
, 'Adaptive Gaussian Thresh

cv2.imshow('Original Image',image)

cv2.imshow('Binarised Image',img)

cv2.imshow('THRESHOLD Image',th3)
cv2.waitKey(0)
cv2.dest
```

Figure 8 Original Image



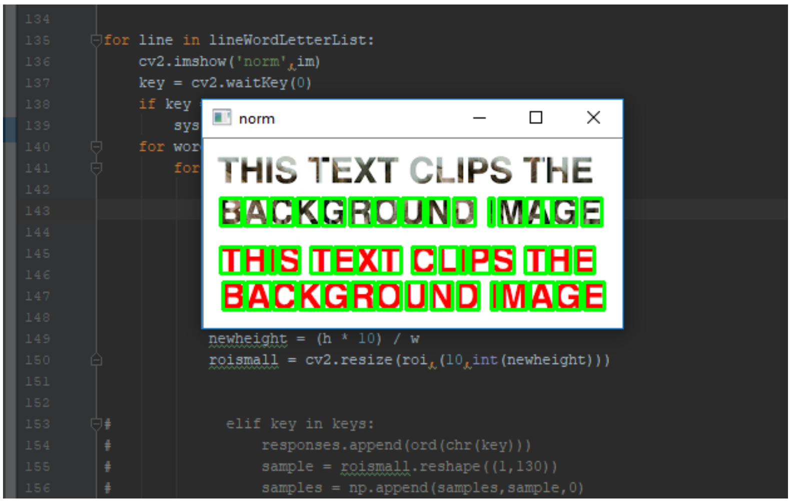
```
cv2.imread('text1.png')
cv2.imread('text1.png',0)
cv2.medianBlur(img,5)

l = cv2.threshold(img,127,255,cv2.THRESH_BINARY
cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRE
      cv2.THRESH_BINARY,11,2)
cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRE
      cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v =
          'Adaptive Mean Thresholding', 'Adaptive G
images = [img, th1, th2, th3]

cv2.imshow('Original Image',image)

cv2.imshow('Binarised Image',img)

cv2.imshow('THRESHOLD Image',th3)
cv2.waitKey(0)
cv2.dest
```

Figure 9 Grayscale Image



```
.imread('text1.png')
.imread('text1.png',0
.medianBlur(img,5)

 cv2.threshold(img,12
.adaptiveThreshold(im
   cv2.THRESH_BINARY,
.adaptiveThreshold(im
   cv2.THRESH_BINARY,

['Original Image', 'G
   'Adaptive Mean Thr
images = [img, th1, th2, th3]

cv2.imshow('Original Image',im

cv2.imshow('Binarised Image',i
```

Figure 10 Image after Adaptive Thresholding



```
for line in lineWordLetterList:
    cv2.imshow('norm',im)
    key = cv2.waitKey(0)
    if key
        sys
    for wor
        for



            newheight = (h * 10) / w
            roismall = cv2.resize(roi,(10,int(newheight)))


#             elif key in keys:
#                 responses.append(ord(chr(key)))
#                 sample = roismall.reshape((1,130))
#                 samples = np.append(samples,sample,0)
```

Figure 11 Contour Detection

If only the three lines of the fig. 4 is considered, then the contour of each segmented character bounded by the green box is saved separately and compared with the training datasets. The recognized text output is shown below:

```
        BACKGROUND  IMAGE

      THIS  TEXT  CIIPS  THE
      BACKGROUND   IMAGE
```

Figure 12 Text Output

CHAPTER FIVE: EPILOGUE

5.1 APPLICATIONS

In recent years, OCR (Optical Character Recognition) technology has been applied throughout the entire spectrum of industries, revolutionizing the document management process. OCR has enabled scanned documents to become more than just image files, turning into fully searchable documents with text content that is recognized by computers. With the help of OCR, people no longer need to manually retype important documents when entering them into electronic databases. Instead, OCR extracts relevant information and enters it automatically.

Our iListen project not only includes OCR but also text to speech conversion. OCR with text-to-speech converts texts from book/document to audio form which facilitates visually impaired to read the texts. It aids every visually impaired enabling them to study the books they want and widening their range of learning confined in Braille language.

This helps the visually challenged to be independent. Without the assistance of a third person, a visually blind person can overcome the challenge of reading a book.

5.2. PROJECT LIMITATIONS

- Requires high quality camera for better result

- Processing takes some time which causes lag in total output

- Proper lighting is required for image capturing

5.3. FUTURE ENHANCEMENTS

The use of commercial frameworks is something that needs to be evaluated, since it displayed better results than the free framework when used without any filter, which suggests that the use of these frameworks in images that have been submitted to filters to improve recognition will lead to even more satisfactory results. The dataset used for this project could be increased, by seeking a broader and more representative set of real use situations.

Another test that could be carried out to assess the quality of the system involves comparing the perceived audio line with the actual text, asking users to write what they thought they had heard and finally making the comparison between the understood text and the original text.

Another issue that should be targeted for improvement is recognition in real time. The alignment of the image is a critical aspect that needs future research work. If the blind user register an image with incomplete text (cropped words on the sides of the image), the final result cannot be useful. Therefore, in the future the application should assist the user in aligning the device with the text to be recognized. Recognition can be eventually done through a process of reading the image pixels, seeking to find the limits between lighter patterns, where possibly there is no text, such as the borders of a page, and darker patterns, where potentially the text is located.

Support for other languages will be another issue to be reviewed. At this stage, the project is designed and developed for English and Nepali language, but in the future it should allow use in other languages, thus extending the number of people who can benefit from the advantages of the project. It is also reserved for future work the increment of the menu options, allowing the user to change other application options such as the language or the setting of sounds and orientations.

5.4. CONCLUSION

The complete effort of this project is the proposal to develop a technology that can convert the picture into textual characters and then into speech signal, this application can be used by Visually Impaired (VI) peoples. This project is expected to be portable, low cost and efficient device which could make the visually impaired people understand the printed text with any scanning difficulty and recognition problems since they are much familiar with the Braille reading. The system can adopt to the maximum font size depending on the resolution of camera used and can scan any text below this size. This technology can be used to identify text from image source captured by USB camera and speech is prepared through a speaker or headphone. There is possibility to convert information both text and picture. This device is of great use to people with visual disability to read textual characters.

REFERENCES

[1]. S. Venkateswarlu, D. B. K. Kamesh, J. K. R. Sastry, Radhika Rani [2016] Text to Speech Conversion. Indian Journal of Science and Technology, Vol 9(38), DOI: 10.17485/ijst/2016/v9i38/102967, October 2016

[2]. D.Velmurugan, M.S.Sonam, S.Umamaheswari, S.Parthasarathy, K.R.Arun[2016]. A Smart Reader for Visually Impaired People Using Raspberry PI. International Journal of Engineering Science and Computing IJESC Volume 6 Issue No. 3.

[3]. K Nirmala Kumari, Meghana Reddy J [2016]. Image Text to Speech Conversion Using OCR Technique in Raspberry Pi. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 5, Issue 5, May 2016.

[4]. Silvio Ferreira, C´eline Thillou, Bernard Gosselin. From Picture to Speech: An Innovative Application for Embedded Environment. Faculté Polytechnique de Mons, Laboratoire de Théorie des Circuits et Traitement du Signal Bˆatiment Multitel - Initialis, 1, avenue Copernic, 7000, Mons, Belgium

[5] Ray Kurzweil KReader Mobile User Guide, knfb Reading Technology Inc. (2008). [Online]. Available: http://www.knfbReading.com

[6] Ms. AthiraPanicker Smart Shopping assistant label reading system with voice output for blind using raspberry pi, Ms. Anupama Pandey, Ms. Vrunal Patil YTIET, University of Mumbai ISSN: 2278 – 1323 International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. 5, Issue 10, Oct 2016 2553 www.ijarcet.org

[7] Marut Tripathi, Manish Kumar, Vivek Kumar, Warsha Kandlikar A Navigation System for blind people International journal for research in applied science and engineering technology (ijraset) – Vol. 2 issue 4, Jul-Aug 2014

[8] Dimitrios Dakopoulos and Nikolaos G.Bourbakis Wearable Obstacle Avoidance Electronic Travel Aids for Blind IEEE Transactions on systems, man and cybernetics, Part C (Applications and Reviews). Vol. 40, issue 1, Jan 2010.

[9] Xilin Chen, , Jie Yang, Jing Zhang, and Alex Waibel Automatic Detection and Recognition of Signs From Natural Scenes ieee transactions on image processing, Vol. 13, NO. 1, Jan 2004.

[10] William A. Ainsworth A system for converting English text into speech IEEE Transactions on Audio and Electroacoustics, Vol. 21, Issue 3, Jun 1973

[11] Zoran Zivkovic. Improved Adaptive Gaussian Mixture Model for Background Subtraction Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on 20 Sep 2004.

[12] S. B. Shrote, Mandar Deshpande, Prashant Deshmukh, Sanjaykumar Mathapati Assistive Translator for Deaf & Dumb People IJECCE Vol. 5, Issue 4 July, Technovision-2014.

[13] Michael McEnancy Finger Reader Is audio reading gadget for Index Finger IJECCE Vol. 5, Issue 4 July-2014.

[14]Vasanthi. G and Ramesh Babu Y India. Vision Based Assistive System for Label Detection with Voice Output, Department of ECE, DMI College of Engineering, Chennai, Jan 2014.

[15]. http://www.zdnet.com/article/raspberry-pi-11-reasons-why-its-the-perfect-small-server/

[16]. Chen X, AL Yuille, Detecting and reading text in natural scenes, in Proc. Computer. Vision Pattern Recognition, 2004; II-366–II-373.

[17]. Kumar S, R Gupta, et al. Text extraction and document image segmentation using matched wavelets and MRF model, IEEE Trans Image Process, August 2007; 16:2117–2128.

[18]. K Kim, K Jung, et al. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm, IEEE Trans. Pattern Anal. Mach. Intell, December 2003; 25: 1631–1639.

[19]. N Giudice, G Legge, Blind navigation and the role of technology, in The Engineering Handbook of Smart Technology for Aging, Disability and Independence, AA Helal, M Mokhtari, B Abdulrazak, Eds. Hoboken, NJ, USA: Wiley, 2008.

[20]. Chen J Y, J Zhang, et al. Automatic detection and recognition of signs from natural scenes, IEEE Trans. Image Process., January 2004 ;13: 87–99.

[21]. D Dakopoulos, NG Bourbakis, Wearable obstacle avoidance electronic travel aids for blind: A survey, IEEE Trans. Syst., Man, Cybern, January 2010; 40: 25–35.

Appendix

Source Code

Code to train data

```
from datetime import datetime
import hashlib
import os
import os.path
import random
import re
import sys
import tarfile

import numpy as np
from six.moves import urllib
import tensorflow as tf

from tensorflow.contrib.quantize.python import quant_ops
from tensorflow.python.framework import graph_util
from tensorflow.python.framework import tensor_shape
from tensorflow.python.platform import gfile
from tensorflow.python.util import compat


MIN_NUM_IMAGES_REQUIRED_FOR_TRAINING = 10
MIN_NUM_IMAGES_SUGGESTED_FOR_TRAINING = 100

MIN_NUM_IMAGES_REQUIRED_FOR_TESTING = 3

MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1  # ~134M

TRAINING_IMAGES_DIR = os.getcwd() + '/training_images'

TEST_IMAGES_DIR = os.getcwd() + "/test_images/"

OUTPUT_GRAPH = os.getcwd() + '/' + 'retrained_graph.pb'

INTERMEDIATE_OUTPUT_GRAPHS_DIR = os.getcwd() + '/intermediate_graph'

INTERMEDIATE_STORE_FREQUENCY = 0

OUTPUT_LABELS = os.getcwd() + '/' + 'retrained_labels.txt'

TENSORBOARD_DIR = os.getcwd() + '/' + 'tensorboard_logs'

HOW_MANY_TRAINING_STEPS=500

LEARNING_RATE = 0.01

TESTING_PERCENTAGE = 10
```

```
VALIDATION_PERCENTAGE = 10

EVAL_STEP_INTERVAL = 10

TRAIN_BATCH_SIZE = 100

TEST_BATCH_SIZE = -1

VALIDATION_BATCH_SIZE = 100

PRINT_MISCLASSIFIED_TEST_IMAGES = False

MODEL_DIR = os.getcwd() + "/" + "model"

BOTTLENECK_DIR = os.getcwd() + '/' + 'bottleneck_data'

FINAL_TENSOR_NAME = 'final_result'

FLIP_LEFT_RIGHT = False

RANDOM_CROP = 0

RANDOM_SCALE = 0

RANDOM_BRIGHTNESS = 0


def main():
    print("starting program . . .")
    tf.logging.set_verbosity(tf.logging.INFO)

    if not checkIfNecessaryPathsAndFilesExist():
        return

    prepare_file_system()

    model_info = create_model_info(ARCHITECTURE)
    if not model_info:
        tf.logging.error('Did not recognize architecture flag')
        return -1
    # end if
    print("downloading model (if necessary) . . .")
    downloadModelIfNotAlreadyPresent(model_info['data_url'])
    print("creating model graph . . .")
    graph, bottleneck_tensor, resized_image_tensor = (create_model_graph(model_info))

    print("creating image lists . . .")
 image_lists = create_image_lists(TRAINING_IMAGES_DIR, TESTING_PERCENTAGE,
VALIDATION_PERCENTAGE)
    class_count = len(image_lists.keys())
    if class_count == 0:
        tf.logging.error('No valid folders of images found at ' + TRAINING_IMAGES_DIR)
        return -1
    # end if
    if class_count == 1:
        tf.logging.error('Only one valid folder of images found at ' + TRAINING_IMAGES_DIR + '
- multiple classes are needed for classification.')
        return -1
    # end if

    # determinf if any of the distortion command line flags have been set
    doDistortImages = False
    if (FLIP_LEFT_RIGHT == True or RANDOM_CROP != 0 or RANDOM_SCALE != 0 or
RANDOM_BRIGHTNESS != 0):
        doDistortImages = True
    # end if

    print("starting session . . .")
    with tf.Session(graph=graph) as sess:
        # Set up the image decoding sub-graph.
        print("performing jpeg decoding . . .")
        jpeg_data_tensor, decoded_image_tensor = add_jpeg_decoding( model_info['input_width'],
        model_info['input_height'], model_info['input_depth'], model_info['input_mean'],
            model_info['input_std'])
        print("caching bottlenecks . . .")
        distorted_jpeg_data_tensor = None
        distorted_image_tensor = None
        if doDistortImages:
 (distorted_jpeg_data_tensor, distorted_image_tensor) =
add_input_distortions(FLIP_LEFT_RIGHT, RANDOM_CROP, RANDOM_SCALE,
RANDOM_BRIGHTNESS, model_info['input_width'], model_info['input_height'],
model_info['input_depth'], model_info['input_mean'], model_info['input_std'])

        Else:
cache_bottlenecks(sess, image_lists, TRAINING_IMAGES_DIR, BOTTLENECK_DIR,
jpeg_data_tensor, decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
ARCHITECTURE)

        print("adding final training layer . . .")
(train_step, cross_entropy, bottleneck_input, ground_truth_input, final_tensor) =
add_final_training_ops(len(image_lists.keys()), FINAL_TENSOR_NAME,  bottleneck_tensor,
 model_info['bottleneck_tensor_size'],  model_info['quantize_layer'])
        # Create the operations we need to evaluate the accuracy of our new layer.
        print("adding eval ops for final training layer . . .")
        evaluation_step, prediction = add_evaluation_step(final_tensor, ground_truth_input)


        # Merge all the summaries and write them out to the tensorboard_dir
        print("writing TensorBoard info . . .")
        merged = tf.summary.merge_all()
        train_writer = tf.summary.FileWriter(TENSORBOARD_DIR + '/train', sess.graph)
        validation_writer = tf.summary.FileWriter(TENSORBOARD_DIR + '/validation')

        # Set up all our weights to their initial default values.
        init = tf.global_variables_initializer()
        sess.run(init)

        # Run the training for as many cycles as requested on the command line.
        print("performing training . . .")
        for i in range(HOW_MANY_TRAINING_STEPS):
            # Get a batch of input bottleneck values, either calculated fresh every
            # time with distortions applied, or from the cache stored on disk.
            if doDistortImages:
 (train_bottlenecks, train_ground_truth) = get_random_distorted_bottlenecks(sess, image_lists,
TRAIN_BATCH_SIZE, 'training', TRAINING_IMAGES_DIR, distorted_jpeg_data_tensor,
 distorted_image_tensor, resized_image_tensor, bottleneck_tensor)

            else:
(train_bottlenecks, train_ground_truth, _) = get_random_cached_bottlenecks(sess, image_lists,
TRAIN_BATCH_SIZE, 'training', BOTTLENECK_DIR, TRAINING_IMAGES_DIR,
jpeg_data_tensor, decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
ARCHITECTURE)
            # end if

            # Feed the bottlenecks and ground truth into the graph, and run a training
            # step. Capture training summaries for TensorBoard with the `merged` op.
train_summary, _ = sess.run([merged, train_step], feed_dict={bottleneck_input:
train_bottlenecks, ground_truth_input: train_ground_truth})
            train_writer.add_summary(train_summary, i)

            # Every so often, print out how well the graph is training.
            is_last_step = (i + 1 == HOW_MANY_TRAINING_STEPS)
            if (i % EVAL_STEP_INTERVAL) == 0 or is_last_step:
train_accuracy, cross_entropy_value = sess.run([evaluation_step, cross_entropy],
feed_dict={bottleneck_input: train_bottlenecks, ground_truth_input: train_ground_truth})
 tf.logging.info('%s: Step %d: Train accuracy = %.1f%%' % (datetime.now(), i, train_accuracy *
100))

            tf.logging.info('%s: Step %d: Cross entropy = %f' % (datetime.now(), i,
cross_entropy_value))
validation_bottlenecks, validation_ground_truth, _ = (get_random_cached_bottlenecks(sess,
image_lists, VALIDATION_BATCH_SIZE, 'validation', BOTTLENECK_DIR,
TRAINING_IMAGES_DIR, jpeg_data_tensor, decoded_image_tensor, resized_image_tensor,
bottleneck_tensor, ARCHITECTURE))

validation_summary, validation_accuracy = sess.run( [merged, evaluation_step],
feed_dict={bottleneck_input: validation_bottlenecks, ground_truth_input:
validation_ground_truth})
           validation_writer.add_summary(validation_summary, i)

tf.logging.info('%s: Step %d: Validation accuracy = %.1f%% (N=%d)' % (datetime.now(), i,
validation_accuracy * 100, len(validation_bottlenecks)))
            # end if

            # Store intermediate results
            intermediate_frequency = INTERMEDIATE_STORE_FREQUENCY

            if (intermediate_frequency > 0 and (i % intermediate_frequency == 0) and i > 0):
 intermediate_file_name = (INTERMEDIATE_OUTPUT_GRAPHS_DIR + 'intermediate_' +
str(i) + '.pb')
                tf.logging.info('Save intermediate result to : ' + intermediate_file_name)
                save_graph_to_file(sess, graph, intermediate_file_name)
            # end if
        # end for
        print("running testing . . .")

 test_bottlenecks, test_ground_truth, test_filenames = (get_random_cached_bottlenecks(sess,
image_lists, TEST_BATCH_SIZE, 'testing', BOTTLENECK_DIR, TRAINING_IMAGES_DIR,
jpeg_data_tensor, decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
ARCHITECTURE))

test_accuracy, predictions = sess.run([evaluation_step, prediction], feed_dict={bottleneck_input:
test_bottlenecks, ground_truth_input: test_ground_truth})
        tf.logging.info('Final test accuracy = %.1f%% (N=%d)' % (test_accuracy * 100,
len(test_bottlenecks)))

        if PRINT_MISCLASSIFIED_TEST_IMAGES:
            tf.logging.info('=== MISCLASSIFIED TEST IMAGES ===')
            for i, test_filename in enumerate(test_filenames):
                if predictions[i] != test_ground_truth[i]:
                    tf.logging.info('%70s  %s' % (test_filename, list(image_lists.keys())[predictions[i]]))


        print("writing trained graph and labbels with weights")
        save_graph_to_file(sess, graph, OUTPUT_GRAPH)
        with gfile.FastGFile(OUTPUT_LABELS, 'w') as f:
            f.write('\n'.join(image_lists.keys()) + '\n')
        print("done !!")




def checkIfNecessaryPathsAndFilesExist():
    if not os.path.exists(TRAINING_IMAGES_DIR):
        print('')
        print('ERROR: TRAINING_IMAGES_DIR "' + TRAINING_IMAGES_DIR + '" does not
seem to exist')
        print('Did you set up the training images?')
        print('')
        return False

    class TrainingSubDir:
        def __init__(self):
            self.loc = ""
            self.numImages = 0


    trainingSubDirs = []

    for dirName in os.listdir(TRAINING_IMAGES_DIR):
        currentTrainingImagesSubDir = os.path.join(TRAINING_IMAGES_DIR, dirName)
        if os.path.isdir(currentTrainingImagesSubDir):
            trainingSubDir = TrainingSubDir()
            trainingSubDir.loc = currentTrainingImagesSubDir
            trainingSubDirs.append(trainingSubDir)
    if len(trainingSubDirs) == 0:
 print("ERROR: there don't seem to be any training image sub-directories in " +
TRAINING_IMAGES_DIR)
             print("Did you make a separare image sub-directory for each classification type?")
             return False

    for trainingSubDir in trainingSubDirs:
        for fileName in os.listdir(trainingSubDir.loc):
            if fileName.endswith(".jpg"):
                trainingSubDir.numImages += 1


    for trainingSubDir in trainingSubDirs:
        if trainingSubDir.numImages < MIN_NUM_IMAGES_REQUIRED_FOR_TRAINING:
print("ERROR: there are less than the required " +
str(MIN_NUM_IMAGES_REQUIRED_FOR_TRAINING) + " images in " + trainingSubDir.loc)
            print("Did you populate each training sub-directory with images?")
            return False






    for trainingSubDir in trainingSubDirs:
        if trainingSubDir.numImages < MIN_NUM_IMAGES_SUGGESTED_FOR_TRAINING:
print("WARNING: there are less than the suggested " +
str(MIN_NUM_IMAGES_SUGGESTED_FOR_TRAINING) + " images in " +
trainingSubDir.loc)
            print("More images should be added to this directory for acceptable training results")


    if not os.path.exists(TEST_IMAGES_DIR):
        print('')
        print('ERROR: TEST_IMAGES_DIR "' + TEST_IMAGES_DIR + '" does not seem to exist')
        print('Did you break out some test images?')
        print('')
        return False

    numImagesInTestDir = 0
    for fileName in os.listdir(TEST_IMAGES_DIR):
        if fileName.endswith(".jpg"):
            numImagesInTestDir += 1

    if numImagesInTestDir < MIN_NUM_IMAGES_REQUIRED_FOR_TESTING:
print("ERROR: there are not at least " +
str(MIN_NUM_IMAGES_REQUIRED_FOR_TESTING) + " images in " +
TEST_IMAGES_DIR)
        print("Did you break out some test images?")
        return False

    return True
# end function

def prepare_file_system():
    if tf.gfile.Exists(TENSORBOARD_DIR):
        tf.gfile.DeleteRecursively(TENSORBOARD_DIR)
    # end if
    tf.gfile.MakeDirs(TENSORBOARD_DIR)
    if INTERMEDIATE_STORE_FREQUENCY > 0:
        makeDirIfDoesNotExist(INTERMEDIATE_OUTPUT_GRAPHS_DIR)
    # end if
    return
# end function

def makeDirIfDoesNotExist(dir_name):
    if not os.path.exists(dir_name):
        os.makedirs(dir_name)





def create_model_info(architecture):
    architecture = architecture.lower()
    is_quantized = False
    if architecture == 'inception_v3':
        # pylint: disable=line-too-long
        data_url = 'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-
05.tgz'
        # pylint: enable=line-too-long
        bottleneck_tensor_name = 'pool_3/_reshape:0'
        bottleneck_tensor_size = 2048
        input_width = 299
        input_height = 299
        input_depth = 3
        resized_input_tensor_name = 'Mul:0'
        model_file_name = 'classify_image_graph_def.pb'
        input_mean = 128
        input_std = 128
    elif architecture.startswith('mobilenet_'):
        parts = architecture.split('_')
        if len(parts) != 3 and len(parts) != 4:
            tf.logging.error("Couldn't understand architecture name '%s'", architecture)
            return None
        # end if
        version_string = parts[1]
if (version_string != '1.0' and version_string != '0.75' and version_string != '0.50' and
version_string != '0.25'):
tf.logging.error("""The Mobilenet version should be '1.0', '0.75', '0.50', or '0.25', but found '%s'
for architecture '%s'""", version_string, architecture)
            return None
        # end if
        size_string = parts[2]
        if (size_string != '224' and size_string != '192' and size_string != '160' and size_string !=
'128'):
 tf.logging.error(""The Mobilenet input size should be '224', '192', '160', or '128', but found '%s'
for architecture '%s'""", size_string, architecture)
            return None
        # end if
        if len(parts) == 3:
            is_quantized = False
        else:
            if parts[3] != 'quantized':
                tf.logging.error(
                    "Couldn't understand architecture suffix '%s' for '%s'", parts[3], architecture)
                return None
            is_quantized = True
        # end if

        if is_quantized:
            data_url = 'http://download.tensorflow.org/models/mobilenet_v1_'
            data_url += version_string + '_' + size_string + '_quantized_frozen.tgz'
            bottleneck_tensor_name = 'MobilenetV1/Predictions/Reshape:0'
            resized_input_tensor_name = 'Placeholder:0'
            model_dir_name = ('mobilenet_v1_' + version_string + '_' + size_string +
'_quantized_frozen')
            model_base_name = 'quantized_frozen_graph.pb'
        else:
            data_url = 'http://download.tensorflow.org/models/mobilenet_v1_'
            data_url += version_string + '_' + size_string + '_frozen.tgz'
            bottleneck_tensor_name = 'MobilenetV1/Predictions/Reshape:0'
            resized_input_tensor_name = 'input:0'
            model_dir_name = 'mobilenet_v1_' + version_string + '_' + size_string
            model_base_name = 'frozen_graph.pb'
        # end if

        bottleneck_tensor_size = 1001
        input_width = int(size_string)
        input_height = int(size_string)
        input_depth = 3
        model_file_name = os.path.join(model_dir_name, model_base_name)
        input_mean = 127.5
        input_std = 127.5
    else:
        tf.logging.error("Couldn't understand architecture name '%s'", architecture)
        raise ValueError('Unknown architecture', architecture)
    # end if

 return {'data_url': data_url, 'bottleneck_tensor_name': bottleneck_tensor_name,
'bottleneck_tensor_size': bottleneck_tensor_size,  'input_width': input_width, 'input_height':
input_height, 'input_depth': input_depth, 'resized_input_tensor_name':
resized_input_tensor_name, 'model_file_name': model_file_name, 'input_mean': input_mean,
'input_std': input_std, 'quantize_layer': is_quantized, }

def downloadModelIfNotAlreadyPresent(data_url):
    dest_directory = MODEL_DIR
    if not os.path.exists(dest_directory):
        os.makedirs(dest_directory)
    # end if
    filename = data_url.split('/')[-1]
    filepath = os.path.join(dest_directory, filename)
    if not os.path.exists(filepath):
        def _progress(count, block_size, total_size):
```

```
sys.stdout.write('\r>> Downloading %s %.1f%%' % (filename, float(count * block_size) /
float(total_size) * 100.0))
            sys.stdout.flush()




        filepath, _ = urllib.request.urlretrieve(data_url, filepath, _progress)
        print()
        statinfo = os.stat(filepath)
tf.logging.info('Successfully downloaded ' + str(filename) + ', statinfo.st_size = ' +
str(statinfo.st_size) + ' bytes')
        print('Extracting file from ', filepath)
        tarfile.open(filepath, 'r:gz').extractall(dest_directory)
    else:
        print('Not extracting or downloading files, model already present in disk')


def create_model_graph(model_info):
    with tf.Graph().as_default() as graph:
        model_path = os.path.join(MODEL_DIR, model_info['model_file_name'])
        print('Model path: ', model_path)
        with gfile.FastGFile(model_path, 'rb') as f:
            graph_def = tf.GraphDef()
            graph_def.ParseFromString(f.read())
            bottleneck_tensor, resized_input_tensor = (tf.import_graph_def(graph_def, name='',
return_elements=[model_info['bottleneck_tensor_name'],
model_info['resized_input_tensor_name'],]))
    return graph, bottleneck_tensor, resized_input_tensor

def create_image_lists(image_dir, testing_percentage, validation_percentage):

    if not gfile.Exists(image_dir):
        tf.logging.error("Image directory '" + image_dir + "' not found.")
        return None

    result = {}

    sub_dirs = [x[0] for x in gfile.Walk(image_dir)]

    is_root_dir = True
    for sub_dir in sub_dirs:
        if is_root_dir:
            is_root_dir = False
            continue

        dir_name = os.path.basename(sub_dir)
        if dir_name == image_dir:
            continue

        extensions = ['jpg', 'jpeg']
        file_list = []
        tf.logging.info("Looking for images in '" + dir_name + "'")
        for extension in extensions:
```

```
      file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
      file_list.extend(gfile.Glob(file_glob))

    if not file_list:
      tf.logging.warning('No files found')
      continue

    if len(file_list) < 20:
      tf.logging.warning('WARNING: Folder has less than 20 images, which may cause issues.')
      elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
 tf.logging.warning('WARNING: Folder {} has more than {} images. Some images will never be
selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))

    label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
    training_images = []
    testing_images = []
    validation_images = []
    for file_name in file_list:
      base_name = os.path.basename(file_name)

      hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
 percentage_hash = ((int(hash_name_hashed, 16) % (MAX_NUM_IMAGES_PER_CLASS + 1))
* (100.0 / MAX_NUM_IMAGES_PER_CLASS))
      if percentage_hash < validation_percentage:
        validation_images.append(base_name)
      elif percentage_hash < (testing_percentage + validation_percentage):
        testing_images.append(base_name)
      else:
        training_images.append(base_name)
      # end if
 result[label_name] = {'dir': dir_name, 'training': training_images, 'testing': testing_images,
'validation': validation_images,}
  return result

###############################################################################
##########################################

def add_jpeg_decoding(input_width, input_height, input_depth, input_mean, input_std):
  jpeg_data = tf.placeholder(tf.string, name='DecodeJPGInput')
  decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
  decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
  decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
  resize_shape = tf.stack([input_height, input_width])
  resize_shape_as_int = tf.cast(resize_shape, dtype=tf.int32)
  resized_image = tf.image.resize_bilinear(decoded_image_4d, resize_shape_as_int)
  offset_image = tf.subtract(resized_image, input_mean)
  mul_image = tf.multiply(offset_image, 1.0 / input_std)
  return jpeg_data, mul_image

###############################################################################
##########################################
def add_input_distortions(flip_left_right, random_crop, random_scale, random_brightness,
input_width, input_height, input_depth, input_mean, input_std):

  jpeg_data = tf.placeholder(tf.string, name='DistortJPGInput')
```

```
    decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
    decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
    decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
    margin_scale = 1.0 + (random_crop / 100.0)
    resize_scale = 1.0 + (random_scale / 100.0)
    margin_scale_value = tf.constant(margin_scale)
    resize_scale_value = tf.random_uniform(tensor_shape.scalar(), minval=1.0,
maxval=resize_scale)
    scale_value = tf.multiply(margin_scale_value, resize_scale_value)
    precrop_width = tf.multiply(scale_value, input_width)
    precrop_height = tf.multiply(scale_value, input_height)
    precrop_shape = tf.stack([precrop_height, precrop_width])
    precrop_shape_as_int = tf.cast(precrop_shape, dtype=tf.int32)
    precropped_image = tf.image.resize_bilinear(decoded_image_4d, precrop_shape_as_int)
    precropped_image_3d = tf.squeeze(precropped_image, squeeze_dims=[0])
    cropped_image = tf.random_crop(precropped_image_3d, [input_height, input_width,
input_depth])
    if flip_left_right:
        flipped_image = tf.image.random_flip_left_right(cropped_image)
    else:
        flipped_image = cropped_image
    # end if
    brightness_min = 1.0 - (random_brightness / 100.0)
    brightness_max = 1.0 + (random_brightness / 100.0)
    brightness_value = tf.random_uniform(tensor_shape.scalar(), minval=brightness_min,
maxval=brightness_max)
    brightened_image = tf.multiply(flipped_image, brightness_value)
    offset_image = tf.subtract(brightened_image, input_mean)
    mul_image = tf.multiply(offset_image, 1.0 / input_std)
    distort_result = tf.expand_dims(mul_image, 0, name='DistortResult')
    return jpeg_data, distort_result
# end function


def cache_bottlenecks(sess, image_lists, image_dir, bottleneck_dir, jpeg_data_tensor,
decoded_image_tensor,  resized_input_tensor, bottleneck_tensor, architecture):



    how_many_bottlenecks = 0
    makeDirIfDoesNotExist(bottleneck_dir)
    for label_name, label_lists in image_lists.items():
        for category in ['training', 'testing', 'validation']:
            category_list = label_lists[category]

            for index, unused_base_name in enumerate(category_list):
 get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir, category,
bottleneck_dir, jpeg_data_tensor, decoded_image_tensor, resized_input_tensor,
bottleneck_tensor, architecture)
            how_many_bottlenecks += 1
            if how_many_bottlenecks % 100 == 0:
                tf.logging.info(str(how_many_bottlenecks) + ' bottleneck files created.')

################################################################################
##########################################
```

```
def get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir, category,
bottleneck_dir, jpeg_data_tensor,  decoded_image_tensor, resized_input_tensor,
bottleneck_tensor, architecture):

    label_lists = image_lists[label_name]
    sub_dir = label_lists['dir']
    sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
    makeDirIfDoesNotExist(sub_dir_path)
    bottleneck_path = get_bottleneck_path(image_lists, label_name, index, bottleneck_dir,
category, architecture)

    if not os.path.exists(bottleneck_path):
create_bottleneck_file(bottleneck_path, image_lists, label_name, index, image_dir, category, sess,
jpeg_data_tensor,  decoded_image_tensor, resized_input_tensor, bottleneck_tensor)

    with open(bottleneck_path, 'r') as bottleneck_file:
        bottleneckBigString = bottleneck_file.read()
    # end with

    bottleneckValues = []
    errorOccurred = False
    try:
        bottleneckValues = [float(individualString) for individualString in
bottleneckBigString.split(',')]
    except ValueError:
        tf.logging.warning('Invalid float found, recreating bottleneck')
        errorOccurred = True

    if errorOccurred:
        create_bottleneck_file(bottleneck_path, image_lists, label_name, index, image_dir, category,
sess,
                               jpeg_data_tensor, decoded_image_tensor, resized_input_tensor,
bottleneck_tensor)

        with open(bottleneck_path, 'r') as bottleneck_file:
            bottleneckBigString = bottleneck_file.read()
        bottleneckValues = [float(individualString) for individualString in
bottleneckBigString.split(',')
    return bottleneckValues

def get_bottleneck_path(image_lists, label_name, index, bottleneck_dir, category, architecture):
return get_image_path(image_lists, label_name, index, bottleneck_dir, category) + '_' +
architecture + '.txt'

################################################################################
##########################################
def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,  image_dir, category,
sess, jpeg_data_tensor, decoded_image_tensor, resized_input_tensor, bottleneck_tensor):
    """Create a single bottleneck file."""
    tf.logging.info('Creating bottleneck at ' + bottleneck_path)
    image_path = get_image_path(image_lists, label_name, index, image_dir, category)
    if not gfile.Exists(image_path):
        tf.logging.fatal('File does not exist %s', image_path)
    # end if
    image_data = gfile.FastGFile(image_path, 'rb').read()
    try:
 bottleneck_values = run_bottleneck_on_image(sess, image_data, jpeg_data_tensor,
decoded_image_tensor, resized_input_tensor, bottleneck_tensor)
    except Exception as e:
        raise RuntimeError('Error during processing file %s (%s)' % (image_path, str(e))

    bottleneck_string = ','.join(str(x) for x in bottleneck_values)
    with open(bottleneck_path, 'w') as bottleneck_file:
        bottleneck_file.write(bottleneck_string)

def run_bottleneck_on_image(sess, image_data, image_data_tensor, decoded_image_tensor,
resized_input_tensor, bottleneck_tensor):

    resized_input_values = sess.run(decoded_image_tensor, {image_data_tensor: image_data})
    # Then run it through the recognition network.
    bottleneck_values = sess.run(bottleneck_tensor, {resized_input_tensor: resized_input_values})
    bottleneck_values = np.squeeze(bottleneck_values)
    return bottleneck_values

################################################################################
##########################################
def get_image_path(image_lists, label_name, index, image_dir, category):
    if label_name not in image_lists:
        tf.logging.fatal('Label does not exist %s.', label_name)
    label_lists = image_lists[label_name]
    if category not in label_lists:
        tf.logging.fatal('Category does not exist %s.', category)
    category_list = label_lists[category]
    if not category_list:
        tf.logging.fatal('Label %s has no images in the category %s.', label_name, category)
    mod_index = index % len(category_list)
    base_name = category_list[mod_index]
    sub_dir = label_lists['dir']
    full_path = os.path.join(image_dir, sub_dir, base_name)
    return full_path

################################################################################
##########################################
def add_final_training_ops(class_count, final_tensor_name, bottleneck_tensor,
bottleneck_tensor_size, quantize_layer):

    with tf.name_scope('input'):
        bottleneck_input = tf.placeholder_with_default(bottleneck_tensor, shape=[None,
bottleneck_tensor_size], name='BottleneckInputPlaceholder')
        ground_truth_input = tf.placeholder(tf.int64, [None], name='GroundTruthInput')

layer_name = 'final_training_ops'
 with tf.name_scope(layer_name):
        quantized_layer_weights = None
        quantized_layer_biases = None
        with tf.name_scope('weights'):
            initial_value = tf.truncated_normal([bottleneck_tensor_size, class_count], stddev=0.001)
            layer_weights = tf.Variable(initial_value, name='final_weights')
            if quantize_layer:
                quantized_layer_weights = quant_ops.MovingAvgQuantize(layer_weights,
is_training=True)
                attachTensorBoardSummaries(quantized_layer_weights)

            attachTensorBoardSummaries(layer_weights)

        with tf.name_scope('biases'):
            layer_biases = tf.Variable(tf.zeros([class_count]), name='final_biases')
            if quantize_layer:
                quantized_layer_biases = quant_ops.MovingAvgQuantize(layer_biases,
is_training=True)
                attachTensorBoardSummaries(quantized_layer_biases)


        attachTensorBoardSummaries(layer_biases)
        with tf.name_scope('Wx_plus_b'):
            if quantize_layer:
                logits = tf.matmul(bottleneck_input, quantized_layer_weights) +
quantized_layer_biases
 logits = quant_ops.MovingAvgQuantize(logits, init_min=-32.0, init_max=32.0, is_training=True,
num_bits=8,  narrow_range=False, ema_decay=0.5)
                tf.summary.histogram('pre_activations', logits)
            else:
                logits = tf.matmul(bottleneck_input, layer_weights) + layer_biases
                tf.summary.histogram('pre_activations', logits)

    final_tensor = tf.nn.softmax(logits, name=final_tensor_name)

    tf.summary.histogram('activations', final_tensor)




    with tf.name_scope('cross_entropy'):
cross_entropy_mean = tf.losses.sparse_softmax_cross_entropy(labels=ground_truth_input,
logits=logits)

    tf.summary.scalar('cross_entropy', cross_entropy_mean)

    with tf.name_scope('train'):
        optimizer = tf.train.GradientDescentOptimizer(LEARNING_RATE)
        train_step = optimizer.minimize(cross_entropy_mean)

    return (train_step, cross_entropy_mean, bottleneck_input, ground_truth_input, final_tensor)

def attachTensorBoardSummaries(var):
    with tf.name_scope('summaries'):
        mean = tf.reduce_mean(var)
        tf.summary.scalar('mean', mean)
        with tf.name_scope('stddev'):
            stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
        # end with
        tf.summary.scalar('stddev', stddev)
        tf.summary.scalar('max', tf.reduce_max(var))
        tf.summary.scalar('min', tf.reduce_min(var))
        tf.summary.histogram('histogram', var)


################################################################################
##########################################
def add_evaluation_step(result_tensor, ground_truth_tensor):


    with tf.name_scope('accuracy'):
        with tf.name_scope('correct_prediction'):
            prediction = tf.argmax(result_tensor, 1)
            correct_prediction = tf.equal(prediction, ground_truth_tensor)
        # end with
        with tf.name_scope('accuracy'):
            evaluation_step = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

    tf.summary.scalar('accuracy', evaluation_step)
    return evaluation_step, prediction

################################################################################
##########################################
def get_random_distorted_bottlenecks(sess, image_lists, how_many, category, image_dir,
input_jpeg_tensor, distorted_image, resized_input_tensor, bottleneck_tensor):

    class_count = len(image_lists.keys())
    bottlenecks = []
    ground_truths = []
    for unused_i in range(how_many):
        label_index = random.randrange(class_count)
        label_name = list(image_lists.keys())[label_index]
        image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
        image_path = get_image_path(image_lists, label_name, image_index, image_dir, category)
        if not gfile.Exists(image_path):
            tf.logging.fatal('File does not exist %s', image_path)
        # end if
        jpeg_data = gfile.FastGFile(image_path, 'rb').read()
        distorted_image_data = sess.run(distorted_image, {input_jpeg_tensor: jpeg_data})
        bottleneck_values = sess.run(bottleneck_tensor, {resized_input_tensor:
distorted_image_data})
        bottleneck_values = np.squeeze(bottleneck_values)
        bottlenecks.append(bottleneck_values)
        ground_truths.append(label_index)
    # end for
    return bottlenecks, ground_truths
# end function

################################################################################
##########################################
def get_random_cached_bottlenecks(sess, image_lists, how_many, category, bottleneck_dir,
image_dir, jpeg_data_tensor,
                                  decoded_image_tensor, resized_input_tensor, bottleneck_tensor,
architecture):

    class_count = len(image_lists.keys())
    bottlenecks = []
    ground_truths = []
    filenames = []
    if how_many >= 0:
        # Retrieve a random sample of bottlenecks.
        for unused_i in range(how_many):
            label_index = random.randrange(class_count)
            label_name = list(image_lists.keys())[label_index]
            image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
            image_name = get_image_path(image_lists, label_name, image_index, image_dir,
category)
            bottleneck = get_or_create_bottleneck(sess, image_lists, label_name, image_index,
image_dir, category, bottleneck_dir,
                                                  jpeg_data_tensor, decoded_image_tensor, resized_input_tensor,
bottleneck_tensor, architecture)
            bottlenecks.append(bottleneck)
            ground_truths.append(label_index)
            filenames.append(image_name)
        # end for
    else:
        # Retrieve all bottlenecks.
        for label_index, label_name in enumerate(image_lists.keys()):
            for image_index, image_name in enumerate(image_lists[label_name][category]):
                image_name = get_image_path(image_lists, label_name, image_index, image_dir,
category)
                bottleneck = get_or_create_bottleneck(sess, image_lists, label_name, image_index,
image_dir, category, bottleneck_dir,
                                                      jpeg_data_tensor, decoded_image_tensor, resized_input_tensor,
bottleneck_tensor, architecture)
                bottlenecks.append(bottleneck)
                ground_truths.append(label_index)
                filenames.append(image_name)
    return bottlenecks, ground_truths, filenames
# end function

################################################################################
##########################################




def save_graph_to_file(sess, graph, graph_file_name):
output_graph_def = graph_util.convert_variables_to_constants(sess, graph.as_graph_def(),
[FINAL_TENSOR_NAME] , with gfile.FastGFile(graph_file_name, 'wb') as f:
        f.write(output_graph_def.SerializeToString())
    # end with
    return
# end function

################################################################################
##########################################
if __name__ == '__main__':
    main()
```

Code for Classification

```
import os
import tensorflow as tf
import numpy as np
import cv2

RETRAINED_LABELS_TXT_FILE_LOC = os.getcwd() + "/" + "retrained_labels.txt"
RETRAINED_GRAPH_PB_FILE_LOC = os.getcwd() + "/" + "retrained_graph.pb"

TEST_IMAGES_DIR = os.getcwd() + "/test_images"

SCALAR_RED = (0.0, 0.0, 255.0)
SCALAR_BLUE = (255.0, 0.0, 0.0)

def main():
    print("starting program . . .")

    if not checkIfNecessaryPathsAndFilesExist():
        return
    classifications = []
    for currentLine in tf.gfile.GFile(RETRAINED_LABELS_TXT_FILE_LOC):
        classification = currentLine.rstrip()
        classifications.append(classification)

    with tf.gfile.FastGFile(RETRAINED_GRAPH_PB_FILE_LOC, 'rb') as retrainedGraphFile:

        graphDef = tf.GraphDef()

        graphDef.ParseFromString(retrainedGraphFile.read())

  if not os.path.isdir(TEST_IMAGES_DIR):
        print("the test image directory does not seem to be a valid directory, check file / directory
paths")
        return
    mystr = " "

    with tf.Session() as sess:
        print("Session has started")
        i = 0
        for Names in os.listdir(TEST_IMAGES_DIR):
            i = i + 1
            print(Names)




        k_prev = 0
        for k in range(i):
            file = str(k) + '.jpg'
            print(file)
```

```
if not (file.lower().endswith(".jpg") or file.lower().endswith(".jpeg")):
    continue


imageFileWithPath = os.path.join(TEST_IMAGES_DIR, file)
openCVImage = cv2.imread(imageFileWithPath)

if openCVImage is None:
    print("unable to open " + file + " as an OpenCV image")
    continue

finalTensor = sess.graph.get_tensor_by_name('final_result:0')
tfImage = np.array(openCVImage)[:, :, 0:3]
predictions = sess.run(finalTensor, {'DecodeJpeg:0': tfImage})
sortedPredictions = predictions[0].argsort()[-len(predictions[0]):][::-1]


for prediction in sortedPredictions:
    strClassification = classifications[prediction]

    if strClassification.endswith("s"):
        strClassification = strClassification[:-1]

    confidence = predictions[0][prediction]

    if onMostLikelyPrediction:
        if (k - k_prev) > 0:
            mystr = mystr + str(' ')
            k_prev = k
        k_prev += 1

        res = str(strClassification)
        # mystr=mystr+str(strClassification)
        if res == str('11'):
            res = 'a'
        if res == str('12'):
            res = 'b'
        if res == str('13'):
            res = 'c'

        if res == str('14'):
            res = 'd'
        if res == str('15'):
            res = 'e'
        if res == str('16'):
            res = 'f'
        if res == str('17'):
            res = 'g'
        if res == str('18'):
            res = 'h'
        if res == str('19'):
```

```
                        res = 'i'
                    if res == str('20'):
                        res = 'j'
                    if res == str('21'):
                        res = 'k'
                    if res == str('22'):
                        res = 'l'
                    if res == str('23'):
                        res = 'm'
                    if res == str('24'):
                        res = 'n'
                    if res == str('25'):
                        res = 'o'
                    if res == str('26'):
                        res = 'p'
                    if res == str('27'):
                        res = 'q'
                    if res == str('28'):
                        res = 'r'
                    if res == str('29'):
                        res = 's'
                    if res == str('30'):
                        res = 't'
                    if res == str('31'):
                        res = 'u'
                    if res == str('32'):
                        res = 'v'
                    if res == str('33'):
                        res = 'w'
                    if res == str('34'):
                        res = 'x'
                    if res == str('35'):
                        res = 'y'
                    if res == str('36'):
                        res = 'z'
                    print(res)
                    mystr = mystr + res
                    onMostLikelyPrediction = False
    print(mystr)
    tfFileWriter = tf.summary.FileWriter(os.getcwd())
    tfFileWriter.add_graph(sess.graph)
    tfFileWriter.close()

def checkIfNecessaryPathsAndFilesExist():
    if not os.path.exists(TEST_IMAGES_DIR):
        print('')
        print('ERROR: TEST_IMAGES_DIR "' + TEST_IMAGES_DIR + '" does not seem to exist')
        print('Did you set up the test images?')
        print('')
        return False
    if not os.path.exists(RETRAINED_LABELS_TXT_FILE_LOC):
 print('ERROR: RETRAINED_LABELS_TXT_FILE_LOC "' +
RETRAINED_LABELS_TXT_FILE_LOC + '" does     not seem to exist')
        return False

    if not os.path.exists(RETRAINED_GRAPH_PB_FILE_LOC):
        print('ERROR: RETRAINED_GRAPH_PB_FILE_LOC "' +
RETRAINED_GRAPH_PB_FILE_LOC + '" does not seem to exist')
        return False

    return True

################################################################################
##########################################
if __name__ == "__main__":
    main()
```

Code for preprocessing and contour detection

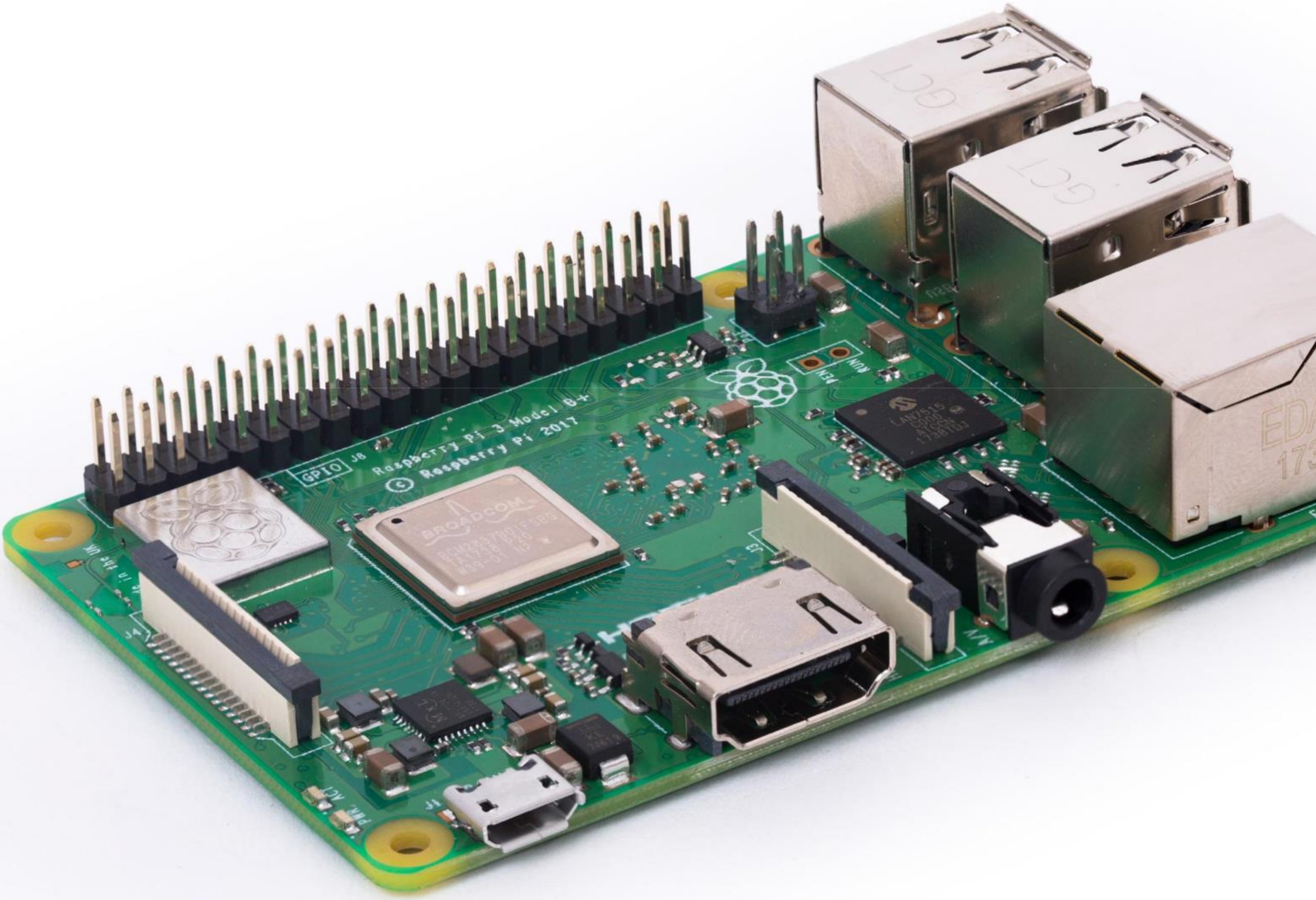
```
import sys
import numpy as np
import cv2
import math

threshold_Area = 20
im = cv2.imread('pak.png')
im3 = im.copy()

gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
thresh = cv2.adaptiveThreshold(gray, 255, 1, 1, 11, 2)


def midpoint_calc(x, y, w, h):
    cx = ((2 * x + w) / 2)
    cy = ((2 * y + h) / 2)
    return cx, cy


def distance_calc(cx1, cy1, cx2, cy2):
    dist = math.sqrt((cx2 - cx1) ** 2 + (cy2 - cy1) ** 2)
    return dist


#################      Now finding Contours        ##################

_, contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

```
samples = np.empty((0, 100))
keys = [i for i in range(ord('!'), ord('~'))]

rectList = []
for cnt in contours:
    area = cv2.contourArea(cnt)
    if area > threshold_Area:
        rectList.append(cv2.boundingRect(cnt))

lineList = []
line = []
letterHeight = rectList[0][3]
rectPrev = rectList[0]
for index, rect in enumerate(rectList):
    [x, y, w, h] = rect
    [_x, _y, _w, _h] = rectPrev

    print('Previous ' + str(rectPrev) + ' New ' + str(rect))
    if _y - y >= 10 and abs(x - _x) > 10:
        print('Found New Line')
        if len(line) > 0:
            lineList.append(line)
            line = []
        letterHeight = h
        line.append(rect)
    else:
        # print 'found same line'
        line.append(rect)

    rectPrev = rect
    # print rect

if len(line) > 0:
    lineList.append(line)

print("Total number of lines " + str(len(lineList)))

lineWordLetterList = []

lineHeightList = []
for singleLine in lineList:
    lineHeight = []
    yTop, yBottom = 999, 0
    sortedLine = sorted(singleLine, key=lambda rect: rect[0])
    for i, rect in enumerate(sortedLine):
        [x, y, w, h] = rect
        if yTop > y:
            yTop = y
        if yBottom < y + h:
            yBottom = y + h
    lineHeight.append(yTop)
    lineHeight.append(yBottom)
    lineHeightList.append(lineHeight)
print(lineHeightList)

for lineNo, singleLine in enumerate(lineList):
    sortedLine = sorted(singleLine, key=lambda rect: rect[0])

    wordGap = int(sortedLine[0][2] / 2)
    wordList = []
    word = []
    previousX = sortedLine[0][0]
    previousRect = sortedLine[0]
    count = 0
    # print wordGap
    isContinue = False

    for i, rect in enumerate(sortedLine):

        [x, y, w, h] = rect
        [_x, _y, _w, _h] = previousRect
        newRect = []

        # print '('+str(_x)+','+str(x)+','+str(_x+_w)+')'+'('+str(_y)+','+str(y)+','+str(_y+_h)+')'
        if _x <= x <= _x + _w and _y > y:  # for i j ! ; % =
            if i != 0:
                # print 'This  is  Enclosed by previous rectangle'
                continue
        newRect.append(x)
        newRect.append(lineHeightList[lineNo][0]),
        newRect.append(w)
        newRect.append(lineHeightList[lineNo][1] - lineHeightList[lineNo][0])

        if abs(previousX - x) > wordGap:
            if len(word) > 0:
                wordList.append(word)
                word = []

            word.append(newRect)
            count += 1
        else:
            word.append(newRect)
            count += 1

        previousX = x + h
        previousRect = rect
    if len(word) > 0:
        wordList.append(word)

    lineWordLetterList.append(wordList)
    lineWordLetterList.reverse()
k = 0
```

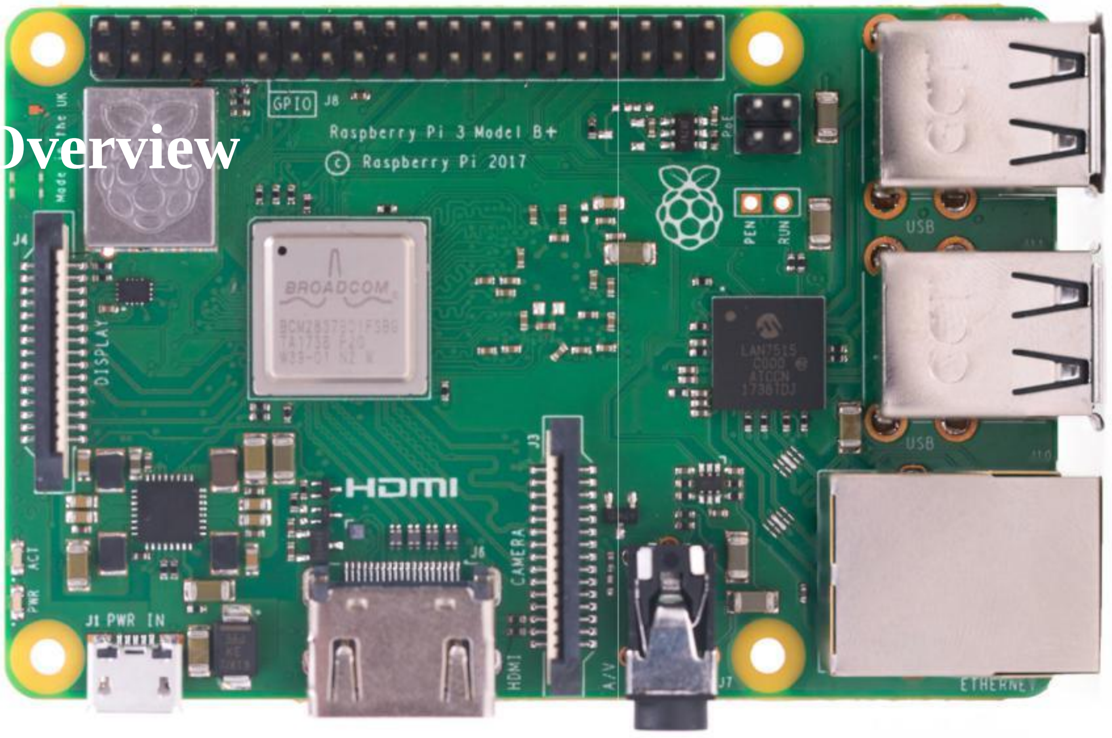
```
for line in lineWordLetterList:

    cv2.imshow('norm', im)
    key = cv2.waitKey(0)
    if key == 27:  # (escape to quit)
        sys.exit()
    i = 0
    for word in line:
        # [_x,_y,_w,_h]=word[0]
        for letter in word:
            [x, y, w, h] = letter
            if (i == 0):
                _cx, _cy = midpoint_calc(x, y, w, h)
                i += 1
            cx, cy = midpoint_calc(x, y, w, h)
            dist = distance_calc(cx, cy, _cx, _cy)
            print(dist)
            if dist > 20:
                k += 1
            _cx, _cy = cx, cy


            cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # To save each contours as image
            roi = thresh[y:y + h, x:x + w]
            resized = cv2.resize(roi, (20, 20))

            cv2.imwrite(str(k) + '.jpg', resized)
            k = k + 1

            newheight = (h * 10) / w
            roismall = cv2.resize(roi, (10, int(newheight)))

        k = k + 1
```

Documentation

Raspberry Pi 3 Model B



Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	1 × full size HDMI MIPI DSI display port MIPI CSI camera port 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	5V/2.5A DC via micro USB connector 5V DC via GPIO header Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50°C

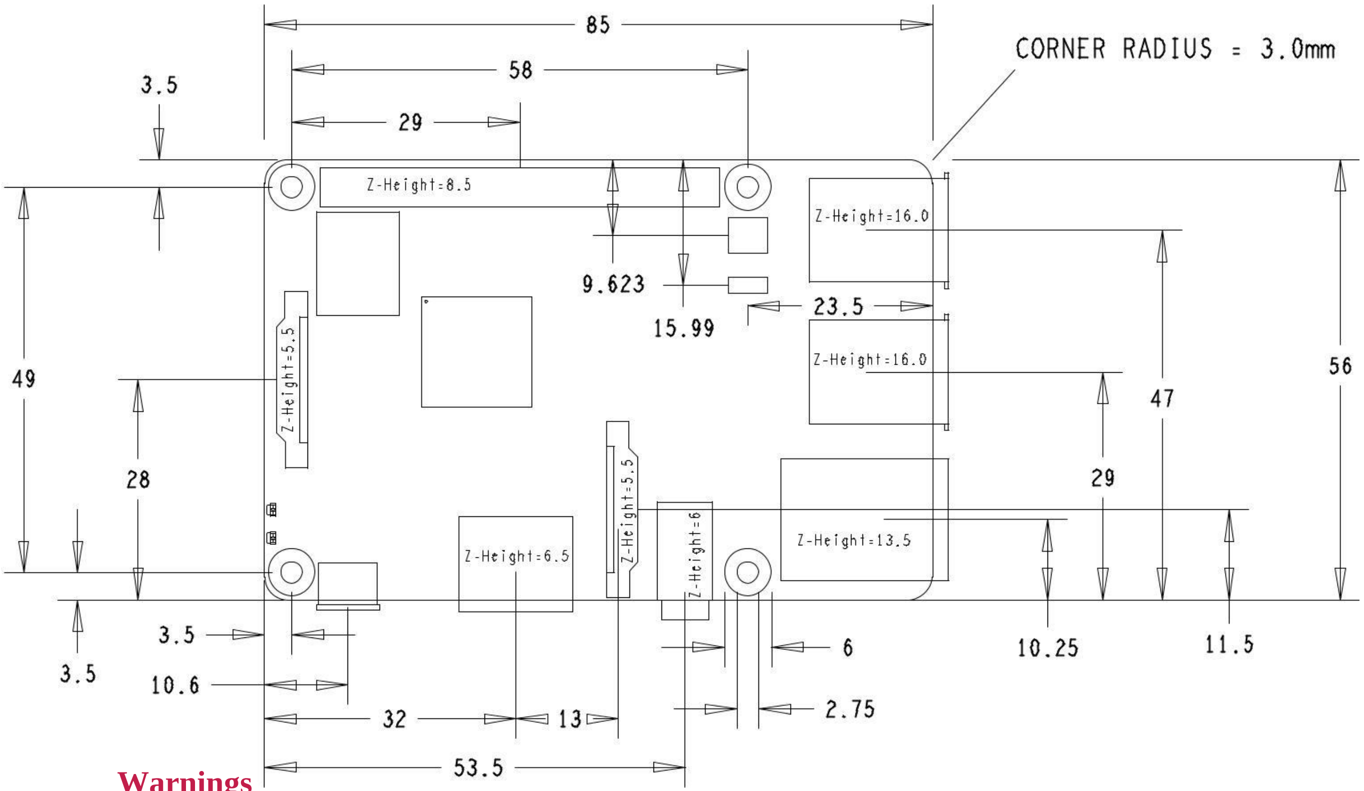
Compliance:

For a full list of local and regional product approvals, please visit **www.raspberrypi.org/products/raspberry - pi-3-mode l-**

Production lifetime:

The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

raspberrypi.org



Warnings

- This product should only be connected to an external power supply rated at 5V/2.5A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.

Raspberry Pi