TRIBHUWAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

Optical Character Recognition using Neural Network

IN FULFILLMENT OF SOFTWARE ENGINEERING PROJECT

Submitted by

Dinesh Bhattarai	072 BCT 512
Aashutosh Poudel	072 BCT 502
Jeevan Thapa	072 BCT 514
Rupesh Shrestha	072 BCT 530
Simon Dahal	072 BCT 538
Yogesh Rai	072 BCT 548

Submitted To
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

March 14, 2018

Acknowledgments

We would like to express our sincere gratitude to our teachers Mr. Biswas Pokhrel and Mrs. Rama Bastola for providing their invaluable guidance, comments and suggestions throughout the course of the project. We would also like to thank the Department of Electronics and Computer Engineering for providing us this golden opportunity to apply the theoretical concepts we have learned during the course that ultimately helped us in understanding the practical side of it.

We would also like to thank our friends and seniors who helped us in choosing the right tools, algorithms and techniques for the project. Also, we must acknowledge our deep sense of gratitude to the mentors of DN: AI Developers Nepal group and their AI Saturdays Meet Up which helped us to get started with our project and gain knowledge of various topics in AI.

Abstract

Optical Character Recognition systems solve the problem of digitizing handwritten or typed documents, forms, letters, etc. OCR systems are widely used to automate and speed up data entry process thereby reducing human errors. The OCR system developed by us has a web interface that is capable of taking input through a camera and giving the output of the recognized text to the user. A system capable of recognizing both handwritten and written English alphabets has been developed as part of this project.

A four layer CNN architecture has been trained and used in order to classify the individual alphabets in the input. Further pre-processing and feature extraction of the input images is done using algorithms like MSER, Gaussian filters, etc. The input image is taken through the camera or through file upload to the system. The feature extraction part is then done to collect inputs for the trained model. The output given by the model is the recognized character output of the entire system. An overall accuracy of 83% has been observed with the system. Further improvements in the system can be made using a more robust set of features and a much larger dataset to improve the accuracy.

Keywords: OCR, handwriting recognition, CNN, neural networks, digits recognition, MSER, Canny Edge Detection

Contents

List of Figures

Introduction

1.1 Background and Recent Research

Optical character recognition is the mechanical or electronics conversion of images of types, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image. It is widely used as a form of information entry from printed paper data records from passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed online, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

In the early days, the OCR systems needed to be trained with images of each characters, and worked on one font at a times. But now advance systems capable of producing high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. In 2006, Geoffry Hinton et al. published a paper [2] showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%). They branded this techniques "Deep learning". Since then many advanced algorithms have been implemented not only to recognize but also to identify and segment texts in images.

1.2 Problem Statement

The major problem with the OCR systems today is that they are trained with only a small number of fonts and thus cannot predict accurately the characters written in various other fonts. The problem is even more serious in case of handwritten digits which are even harder to recognize as each indi-

vidual has a completely different style of writing and thus a unique character representation in their writing. This makes the job of OCR systems much more difficult.

In the present world, many of the human tasks are being automated and equipped with machine intelligence. This means machines need to see and understand the outside world as we humans do. That is why, they need to identify objects, texts, from images and interpret them just like we do. For this a robust and accurate system to identify and classify texts is needed. So in this project an OCR system capable of recognizing handwritten and printed digits has been developed.

There are OCR systems already in the market but they are very specific in their inputs. They are specific to one or two fonts, and require perfect lighting conditions and orientation of the characters in the image.

This system is different from the present systems because it has been trained on various datasets consisting of letters in varieties of fonts, colors and sizes. It has been developed with the convolutional neural networks which are great for efficient learning of features from images and classifying them. Also a number of preprocessing is performed in the image before it is send to the prediction model. This considerably reduces the errors for the prediction model and thus improves its accuracy.

1.3 Objectives

- Develop an OCR system to automate problems like digitizing forms, documents, data entry, etc.
- Develop an OCR system to recognize handwritten as well as printed digits

1.4 Motivation

The major problem with the OCR systems today is that they are trained with only a small number of fonts and thus cannot predict accurately the characters written in various other fonts. The problem is even more serious in case of handwritten digits which are even harder to recognize as each individual has a completely different style of writing and thus a unique character representation in their writing. This makes the job of OCR systems much more difficult. In the present world, many of the human tasks are being automated and equipped with machine intelligence. This means machines need to see and understand the outside world as we humans do. That is why,

they need to identify objects, texts, from images and interpret them just like we do. For this a robust and accurate system to identify and classify texts is needed. So in this project an OCR system capable of recognizing handwritten and printed digits has been developed. There are OCR systems already in the market but they are very specific in their inputs. They are specific to one or two fonts, and require perfect lighting conditions and orientation of the characters in the image. This system is different from the present systems because it has been trained on various datasets consisting of letters in varieties of fonts, colors and sizes. It has been developed with the convolutional neural networks which are great for efficient learning of features from images and classifying them. Also a number of preprocessing is performed in the image before it is send to the prediction model. This considerably reduces the errors for the prediction model and thus improves its accuracy.

Literature Review

There are many methods that are applied in text extraction and recognition. Maximally Stable Extremal Regions along with canny edge detection has been used as a method of text detection in images [3]. The approach used by us is partly based on Deep learning based large scale handwritten Devanagari character recognition [1] which concerns only with the recognition of Devanagari characters using a CNN architecture. In addition to that an image pre-processor is used to enhance further the accuracy of the CNN architecture.

We have tried to combine the concepts of the two in our system.

2.1 Maximally Stable Extremal Regions and Canny Edge Detector

MSER has been identified as one of the best region detectors due to its robustness against view point, scale, and lighting changes. But it is sensitive to image blur. To cope with blurred images, the complimentary property of Canny edges is combined with MSER to enhance its performance. The outline of extremal regions can be enhanced by applying the precisely located but not necessarily connected Canny edges. i.e. the MSER pixels outside the boundary formed by the Canny edges are removed.

2.2 Convolutional Neural Networks

Deep Neural Networks do not require any feature to be explicitly defined, instead they work on the raw pixel data generating the best features and using it to classify the inputs into different classes. Deep Neural Networks consist of multiple nonlinear hidden layers so the number of connections and trainable parameters are very large. Besides being very hard to train, such networks require a very large set of examples to prevent overfitting. One class

of DNN with comparatively smaller set of parameters and easier to train is Convolutional Neural Networks. The ability of CNN to correctly model the input dataset can be varied by changing the number of hidden layers and the trainable parameters in each layer and they also make correct assumption on the nature of images.

Like a standard feed forward network, they can model complex non-linear relationship between input and output. But CNN have very few trainable parameters than a fully connected feed-forward network of same depth. CNNs introduce the concept of local receptive field, weight replication and temporal subsampling which provide some degree of shift and distortion invariance. CNNs for image processing generally are formed of many convolution and sub-sampling layers between input and output layer.

Theoretical Background

3.1 Convolutional Neural Networks:

Convolutional Neural Network (CNN or ConvNet) is a biologically inspired trainable machine learning architecture that can learn from experiences like standard multilayer neural networks. ConvNets consist of multiple layers of overlapped tiling collections of smaller neurons to achieve better representation of the original image. ConvNets are widely used for image and video recognition. There are three main types of layers used to build a ConvNet architecture.

1. Convolution Layer

   The convolution layer is the core building block of a convolutional neural network. It convolves the input image with a set of learnable filters or weights, each producing one feature map in the output image.

2. Pooling Layer

   The pooling layer is used to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The pooling layer takes small rectangular blocks from the convolution layer and subsamples it to produce a single output from the block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block.

3. Fully-Connected Layer

   The fully connected layer is used for the high-level reasoning in the neural network. It takes all neurons in the previous layer and connects it to every single neuron it has. Their activations can be computed with a matrix multiplication followed by a bias offset as a standard neural networks.

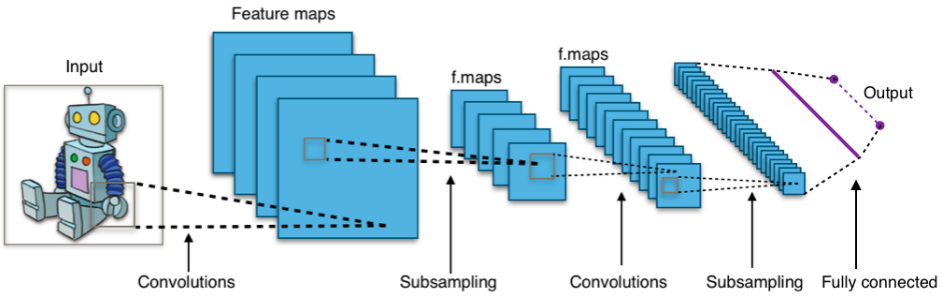

Figure 3.1: Typical CNN architecture

3.2 Maximally Stable Extremal Regions

Maximally Stable Extremal Regions (MSER) is a feature detector. The MSER algorithm extracts from an image a number of co-variant regions, called MSERs. An MSER is a stable connected component of some level sets of the image. Optionally, elliptical frames are attached to the MSERs by fitting ellipses to the regions.

3.3 Canny Edge Detection

The Canny Edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. The process of Canny edge detection algorithm are:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edges by hysteresis: finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Methodology

4.1 Use case diagram

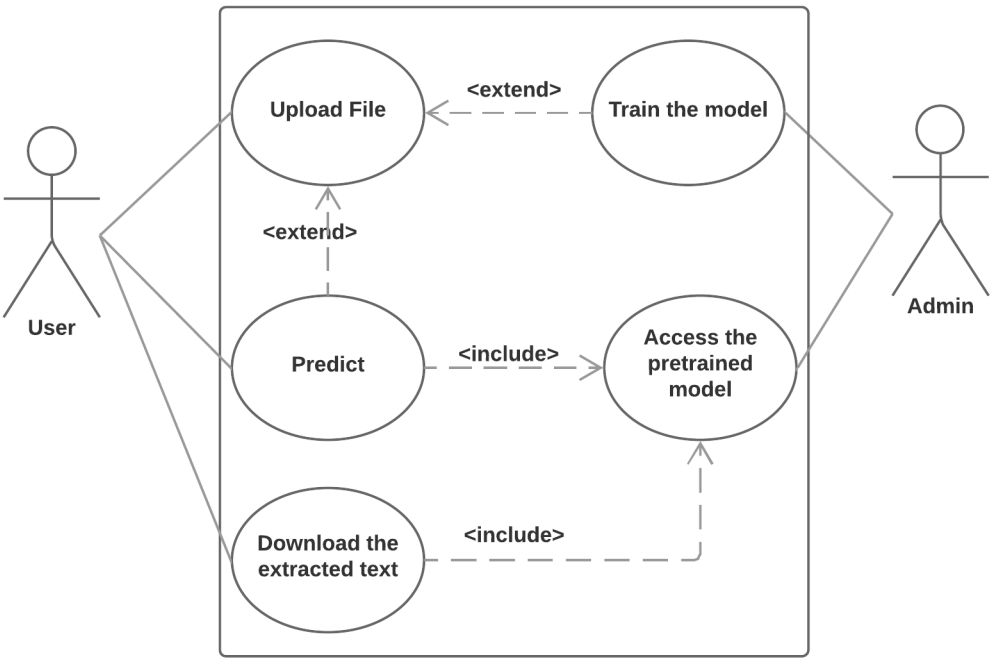

Figure 4.1: Use Case diagram

The purpose of a use case diagram here is to demonstrate the different ways that a user might interact with a system. The main actors in our systems are the actual users who use it and the administrator responsible for updating or training the model used in the system. The user uses the system

in order to predict the text present in the image by uploading the image and gets the output from the prediction model as indicated in the relationship above.

Also, after the user chooses to predict the text in the image, the pre-trained model is automatically accessed to produce the required output as shown by the relationship between the two use cases.

4.2 Level 0 Data Flow Diagram

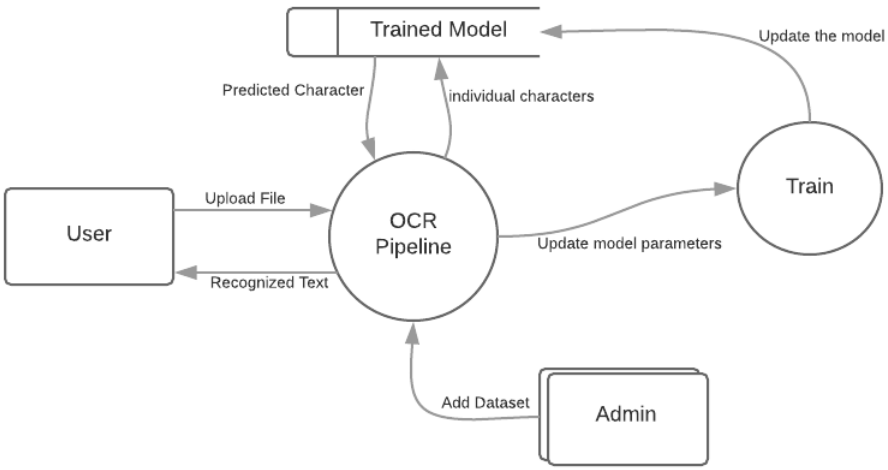

Figure 4.2: Level 0 Data Flow Diagram

DFD Level 0, also called Context Diagram, shows a basic overview of the whole system. The above diagram provides an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. The circles in the diagram represent a process, the rectangular boxes represent entities and the arrows indicate the data flow direction.

In our system, the external entities are the user and the admin. The OCR pipeline and the trained model are internal to the system. The user directly interacts with the pipeline giving image file as an input and gets the desired output in the form of text from the pipeline. Similarly the admin also interacts with the pipeline with the help of dataset that is used for training as well as testing purposes. Further, the pipeline interacts with the trained prediction model to generate the approximate match of the input character. It can also train the model using the dataset provided by the administrator.

4.3 Level 1 Data Flow Diagram

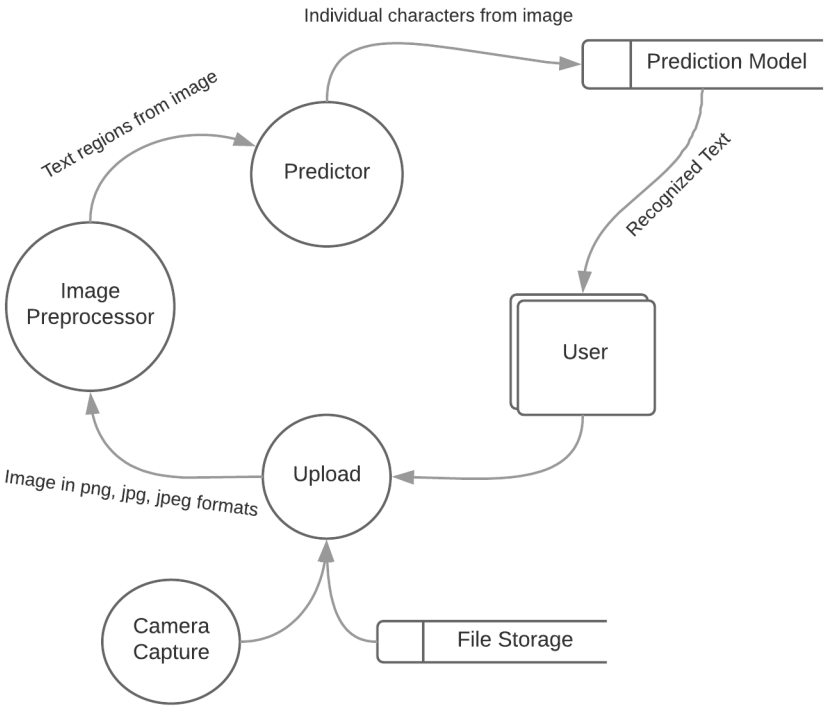

Figure 4.3: Level 1 Data Flow Diagram

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. The above level 1 DFD diagram of our system shows the OCR pipeline in more detail.

It includes an input mechanism through an external device like camera or through a file storage unit. The OCR pipeline consists of an image processor unit which applies various image processing algorithms to the image to identify text regions in the image. The next processes consists of predicting the text regions with the individual characters separated from the image and updating the user of the identified texts present in the image.

4.4 Level 2 Data Flow Diagram

DFD Level 2 goes one step deeper into parts of Level 1.

Thus the OCR pipeline from Level 1 is further divided into its constituent's parts. The user interacts with the system using a web interface to upload the image file as well as get the output of the system. The image is then pre-processed using binarization, extraction using a combination of MSER and Canny Edge detection. After the text regions are obtained the regions gets segmented into characters and fed into the recognition engine. The recognition engine uses a pre-trained model trained by a four layer CNN architecture to predict the approximate output of the character or text. The predicted characters are combined to form words or sentences that is displayed to the user.

4.5 Entity Relationship Diagram

An Entity Relationship (ER) diagram shows how entities like people, objects, or concepts relate to each other within a system. In an ER diagram, the rectangular boxes represent the objects, or entities, diamonds represent the relationship between the entities, and an oval represents an attribute of an entity.

In the proposed system we have two major entities the user and the OCR pipeline. A user can upload one or many files at a time to the system. The pipeline has various components that analyses the given image and prints the gives the output of the images to on or many users.

4.6 Activity Diagram

An activity diagram is essentially a flowchart that shows activities performed by a system. The major activities in our system is outlined in the green rounded rectangles. This includes all the major portions of the data flow diagrams as well as the decision making steps carried out to give the output to the user.

4.7 Training and recognition

To do character recognition we used Python Programming Language along with neural network tools like Tensorflow, Keras, etc and some image processing tools like Scipy, OpenCV. The whole process is divided into the following categories:

- Training Architecture: We used Convolutional Neural Network (CNN) as neural network. The CNN used is sequential model which consists of nine layers with four convolutional layers excluding the input and output layers.

- Pre-processing of the image for creation of dataset(in CSV file): First of all, we converted each image of the datasets into grayscale and converted it into numerical two dimensional arrays corresponding to the pixel intensity of each image and saved the arrays in CSV file appending the label of character of each image to each array. The data arrays were arranged randomly for the better training of the datasets.

- Training and Testing of the network: For training, 80% of the datasets were taken and the remaining were used for cross validation and testing. The datasets were used 10 times i.e for 10 times we trained the CNN with same datasets. So, the whole process of training & testing took about 15 minutes and the accuracy obtained was about 86%.

- Recognition:

  Recognition is same as that of testing. The image which is to be recognized is passed into image processor for conversion of it into gray-scaled array of the pixel intensity and passed into predictor function which predicts the label of the image.

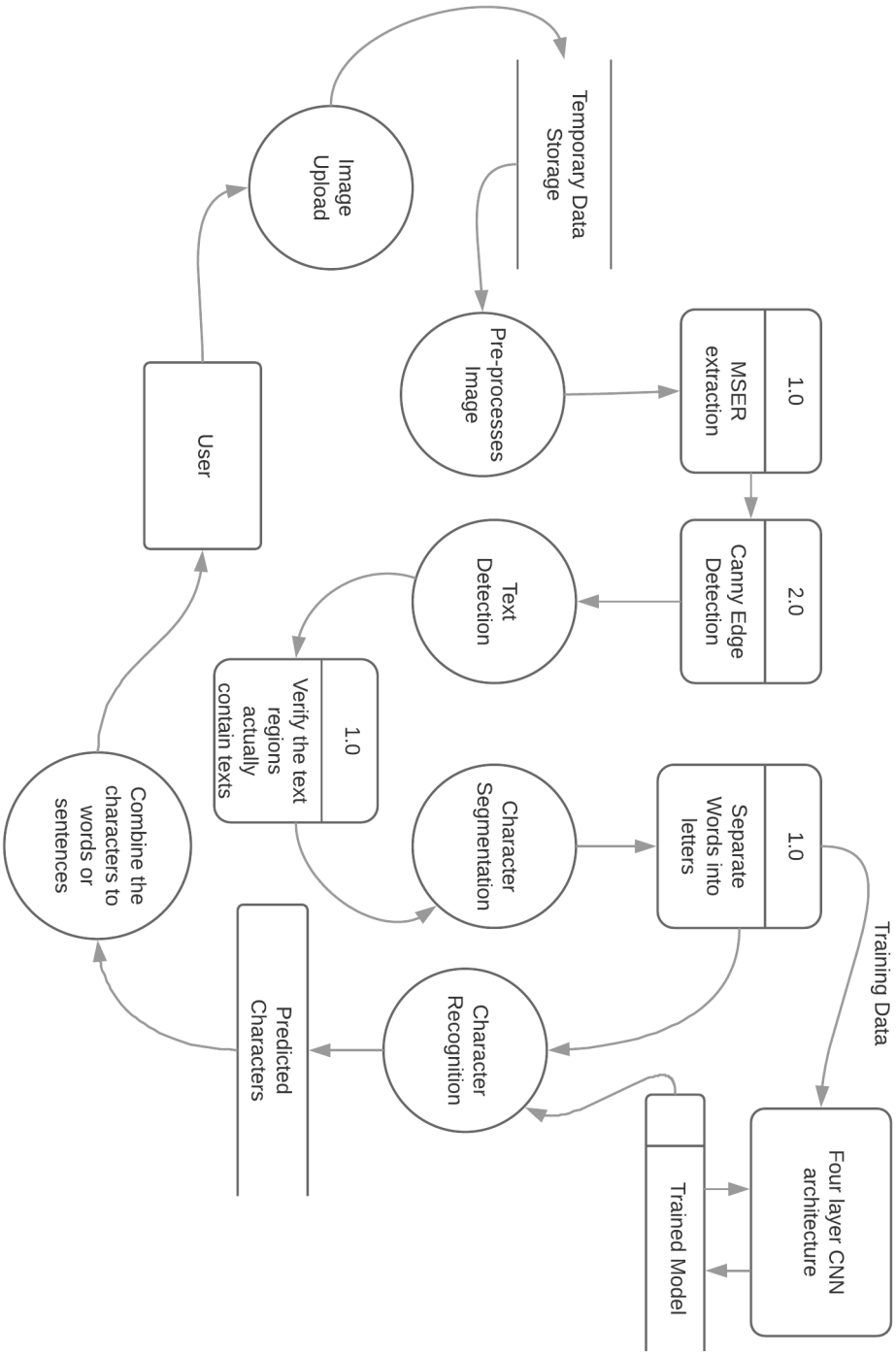

Figure 4.4: Level 2 Data Flow Diagram

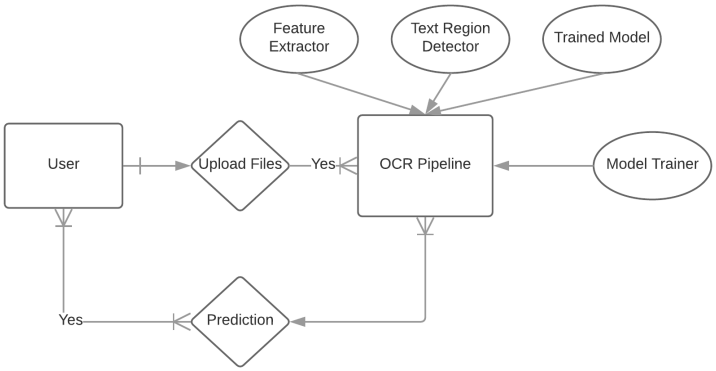

Figure 4.5: Entity Relationship Diagram

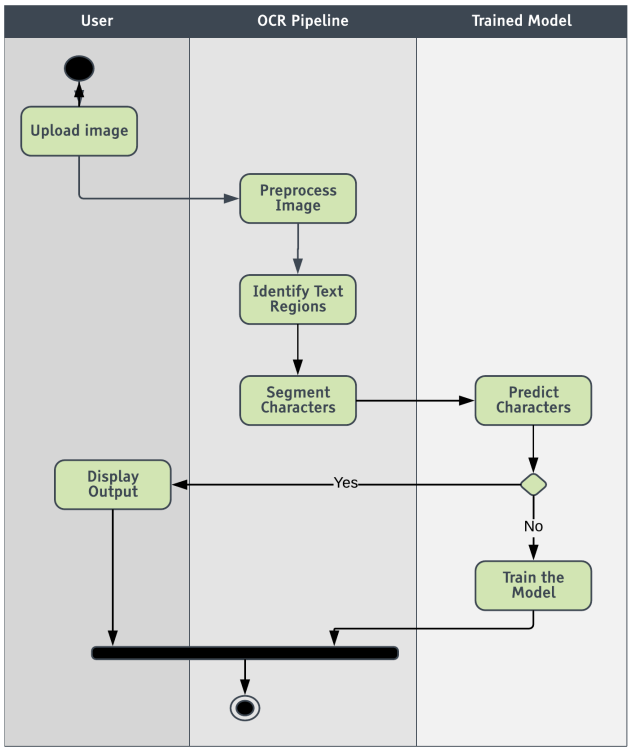

Figure 4.6: Activity Diagram

Figure 4.7: Training images for our model

Result

First we got our hands dirty with MNIST datasets and Neural Network with the help of the book Make Your Own Neural Network [4]. The output from the model trained from this dataset for digits recognition was 97%.

Then we approached Optical Character Recognition problem with divide and conquer. We first enlisted all the possible algorithms/models. Then we shortlisted the models and bruteforced each model individually and compared the results. Again we re-evaluated our models and tried to optimize our model further by parameter tuning. Then we finally compared the results and decided to use neural networks since it gave best result at reasonable latency.

5.1 Output

When we trained our model with MNIST dataset for digit recognition, it gave accuracy of 97%.

We used 74 k datasets for our training model. This dataset contains 74000 sets of data with 62 categories including capital and small letters and 10 digits. Thus trained model was firstly 83% accurate. Later we improved it to 86% accuracy with parameter tuning and optimizations.

5.2 Comparision

Comparing the system with other available systems, our system performs decently. Most of the Optical Character Recognition Systems have accuracy in the range of 80-95%.

However, we also developed a web front-end for our system which makes the system accessible to the end-users. Most of the other systems are just APIs or command line tools that are not so usable for the end users. In this regard, our systems stands out.

Conclusion

In this project, an OCR system has been developed which employs Maximally Stable Extremal Regions as basic letter candidates. To overcome the sensitivity of MSER with respect to image blur and to detect even very small letters, a modified version of MSER complimented with the Canny edges is used. The detected text are binarized letter patches, which are directly used for text recognition purposes. CNN does not require specific feature to be added as input. It works on the raw pixel of image and extracts features from there based on the number of layers and the number of neurons in each layers. The system worked with an accuracy of around 83% which makes us confident for using it in recognizing handwritten as well as printed texts.

6.1 Limitations

1. Accuracy of the system is not satisfactory.

6.2 Future Enhancements

1. Accuracy of the system can be improved by using a much larger dataset.
2. Second, the CNN architecture can be improved further to include more features.
3. Pre-processing of image can be improved using techniques like stroke width transforms which provides a much greater confidence in identifying text regions in the image.
4. Extension of the system to identify whole words at a time instead of characters
5. Can be extended to other languages, logos, symbols, and even used in vehicle plate recognition.

References

[1] S. Acharya, A. K. Pant, and P. K. Gyawali. "Deep learning based large scale handwritten Devanagari character recognition". In: *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*. 2015, pp. 1–6. DOI: 10.1109/SKIMA.2015.7400041 (cit. on p. 4).

[2] Y. Bengio. *Learning Deep Architectures for AI*. Essence of knowledge, The. Now Publishers, 2009 (cit. on p. 1).

[3] Wenyi Huang, Dafang He, Xiao Yang, Zihan Zhou, Daniel Kifer, and C. Lee Giles. "Detecting Arbitrary Oriented Text in the Wild with a Visual Attention Model". In: *Proceedings of the 2016 ACM on Multimedia Conference*. MM '16. Amsterdam, The Netherlands: ACM, 2016, pp. 551–555. DOI: 10.1145/2964284.2967282 (cit. on p. 4).

[4] Tariq Rashid. *Make Your Own Neural Network*. 1st ed. CreateSpace Independent Publishing Platform, 2016 (cit. on p. 16).