A Major Project Report

on

“LONG SHORT-TERM MEMORY BASED OCR”

Submitted in partial fulfilment of the requirements for the award of the degree

Of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SIMHADRI SAI YASHWANTH

(20EG105147)

Under the guidance of

Dr. K. SHAILAJA

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Venkatapur(V), Ghatkeshar(M), Medchal(D) – 500088

2023-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the report / dissertation entitled **"LONG SHORT-TERM MEMORY BASED OCR"** that is being submitted by **RAMISHETTI JOSHUA RAHUL [20EG105713]** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering to the Anurag University, Hyderabad is a record of bonafide work carried out by him under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Internal Guide

Dr. K. Shailaja
Assistant Professor, Dept. of CSE

Dr. G. Vishnu Murthy
Professor &Dean, Dept. of CSE

External Examiner

ACKNOWLEDGMENT

I would like to express my sincere thanks and deep sense of gratitude to the project supervisor **Dr. K. Shailaja** for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved my grasp of the subject and steered me to the fruitful completion of the work. Her patience, guidance, and encouragement made this project possible.

I would like to acknowledge my sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. I also express my deep sense of gratitude to **Dr. V V S S S Balaram**, Academic coordinator, **Dr. Pallam Ravi**, Project in-charge, and **Dr. G. Prabhakar Raju** . Project Coordinator and Project Review Committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage of my project work.

I would like to express my special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in my B.Tech program.

SIMHADRI SAI YASHWANTH
(20EG105147)

DECLARATION

I, hereby declare that the Report entitled **"LONG SHORT-TERM MEMORY BASED OCR"** submitted for the award of Bachelor of technology Degree is my original work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

SIMHADRI SAI YASHWANTH
(20EG105147)

PLACE: HYDERABAD
DATE:

ABSTRACT

Optical Character Recognition (OCR) is a pivotal technology in the digitization of printed and handwritten texts. It enables the conversion of various types of documents, such as scanned paper documents, PDF files, or images captured by digital cameras, into machine-encoded text. This process involves capturing the image of the text and employing algorithms to detect and recognize characters, which are then transformed into a digital format that can be edited, searched, and processed. OCR has widespread applications across numerous fields, including data entry, document management, and accessibility, enhancing efficiency and productivity. With ongoing advancements in machine learning and artificial intelligence, OCR continues to evolve, offering improved accuracy and the ability to recognize a broader array of fonts and languages.

CONTENTS

LIST OF FIGURES

CHAPTER-1 : INTRODUCTION

1.1 BACKGROUND

OCR stands for Optical Character Recognition. It's the process that magically transforms a text image (like a scanned document or a photo) into a machine-readable text format that computers can understand. Imagine you scan an invoice or a receipt. Your computer saves it as an image file, but the phrases within that image file cannot be edited, searched, or counted using a regular text editor. OCR takes that image and performs character recognition, converting it into a text file where the contents are saved as editable text data.

Ray Kurzweil invented OCR in 1974 to read text printed in almost any font. Over time, OCR evolved, benefiting from advances in computer vision, artificial intelligence, and machine learning. Now, with smartphones and mobile apps, a wider audience enjoys OCR for tasks like text recognition from photos.

1.2 OBJECTIVES

1. Recognize Characters from given Image: Successfully identify different types of characters of Telugu language based on the camera sensor data or a given input image.

   Characters in Telugu can be categorized as

   - Achullu
   - Guninthalu
   - Hallulu
   - Othulu
2. Convert recognized characters into words and sentences : Accurately recognize the characters present in input image and provide the text format in desired output file.

1.3 SIGNIFICANCE

Ultimately, the goal of this project is not only to build an accurate and reliable character recognition system and also to contribute to the ongoing efforts in enhancing digitization of the old school literature and preserve the culture of the language.

1.4 PROJECT OVERVIEW

Our project proceeds through a structured approach involving data collection, preprocessing, model training, testing, and user interface development. This framework enables us to leverage the abundance of data available on the internet and apply advanced deep learning techniques to perform Optical Character Recognition.

CHAPTER-2 : LITERATURE SURVEY

2.1 OPTICAL CHARACTER RECOGNITION (OCR) FOR TELUGU:

- **Authors**: Konkimalla Chandra Prakash, Y. M. Srikar, Gayam Trishal, Souraj Mandal, Sumohana S. Channappayya

Telugu is a Dravidian language spoken by more than 80 million people worldwide. The optical character recognition (OCR) of the Telugu script has wide ranging applications including education, health-care, administration etc. The beautiful Telugu script however is very different from Germanic scripts like English and German. This makes the use of transfer learning of Germanic OCR solutions to Telugu a non-trivial task. To address the challenge of OCR for Telugu, we make three contributions in this work: (i) a database of Telugu characters, (ii) a deep learning based OCR algorithm, and (iii) a client server solution for the online deployment of the algorithm.The performance of an OCR system depends hugely on the performance of its classifier. Previous works on Telugu OCR have done the character level segmentation based on histograms along the x and y directions. Assuming that the histogram method for segmentation would work perfectly, they have used SVM based classifiers for character classification. However, we have observed that in real scenarios, the histogram method fails to properly segment out the vattu and the main character together. It also fails when the characters are rotated or if they share a common region when projected on x-axis or y-axis.Inspired by the success of deep neural networks for feature learning, we have explored CNNs to classify the characters and proposed a new architecture for the same. It eliminates the dependency on hand-crafted features and directly learns useful features from the training data itself. It is a combination of both a feature extractor and a classifier and mainly consists of convolutional (weight-sharing), pooling and fully connected layers.

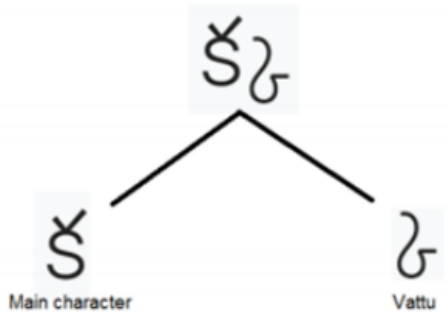


Fig 2.1.1: Main Character and Vattu

Pre-processing consists of skew correction in which tilt in the image is adjusted and binarization after which the image is binarized and segmented into individual words. This is followed by character level segmentation and classification.Our segmentation algorithm assumes that there is no skew in the image. This makes skew correction a matter of utmost importance. We have used a straight line Hough transform based technique for correcting skew that can detect and correct skew upto 90 degrees. We used a modified version of Otsu's thresholding for our binarization. We used a morphological closing algorithm for noise removal. We then computed the logical OR between the denoised image and that of the Otsu's thresholding result and applied mode based threshold on it.

Classifier

The performance of an OCR system depends hugely on the performance of its classifier. Previous works on Telugu OCR have done the character level segmentation based on his- tograms along the x and y directions. Assuming that the histogram method for segmentation would work perfectly, they have used SVM based classifiers for character classifica- tion. However, we have observed that in real scenarios, the histogram method fails to properly segment out the vattu and the main character together. It also fails when the characters are rotated or if they share a common region when projected on x-axis or y-axis.

Pre-Processing and Segmentation

Pre-processing consists of skew correction in which tilt in the image is adjusted and binarization after which the image is binarized and segmented into individual words. This is followed by character level segmentation and classification.

Mobile Application

To facilitate usability, we have developed an Android app that deploys the proposed OCR solution. The app does on- line image to text conversion with Industry standard (MVP Architecture) which works on any Android (4.4+) device.

2.2 TELUGU OCR FRAMEWORK USING DEEP LEARNING

- **Authors:** Rakesh Achanta , Trevor Hastie, and Stanford University

Abstract: In this work, the authors address the task of Optical Character Recognition(OCR) for the Telugu script. We present an end-to-end framework that segments the text image, classifies the characters and extracts lines using a language model. The segmentation is based on mathematical morphology. The classification module, which is the most challenging task of the three, is a deep convolutional neural network. The language is modelled as a third degree markov chain at the glyph level. Telugu script is a complex alphasyllabary and the language is agglutinative, making the problem hard. In this paper they apply the latest advances in neural networks to achieve state-of- the-art error rates. We also review convolutional neural networks in great detail and expound the statistical justification behind the many tricks needed to make Deep Learning work.

1. Introduction.

There has been limited study in the development of an end-to-end OCR system for the Telugu script. While the availability of a huge online corpus of scanned documents warrants the necessity for an OCR system, the complex script and agglutinative grammar make the problem hard. Building a system that works well on real-world documents containing noise and erasure is even more challenging.The task of OCR is mainly split into segmentation and recognition. The design of each is guided by that of the other. The more robust (to noise, erasure, skew etc.) the segmentation is, the easier the task of the recognizer becomes, and vice-versa. The techniques used in segmentation are somewhat similar across scripts. This is because, usually, one connected component (a contiguous region of ink) can be extracted to give one unit of written text. While this principle applies to the Roman scripts with few exceptions; it does not hold for complex scripts like Devanagari and Arabic, where words, not letters, are written in one contiguous piece of ink. The Telugu script is of intermediate complexity, where consonant-vowel pairs are written as one unit.

The recognition task is traditionally split into feature extraction and clasification. The former has been hand-engineered for a very long time.

Keywords and phrases: Machine Learning, Deep Learning, Convolutional Neural Networks, Optical Character Recognition, Gradient Based Learning, Document Recognition, Telugu, Natural Language Processing.

Early as 1977, Telugu OCR systems used features that encode the curves that trace a letter, and compare this encoding with a set of predefined templates (Rajasekaran and Deekshatulu, 1977; Rao and Ajitha, 1995). The first attempt to use neural networks for Telugu OCR to our knowledge was in Sukhaswami, Seetharamulu and Pujari (1995). They train multiple neural networks, and pre-classify an input image based on its aspect ratio and feed it to the corresponding network. This reduces the number of classes that each sub-network needs to learn. But this is likely to increase error rate, as failure in pre-classification is not recoverable. The neural network employed is a Hopfield net on a down-sampled vectorized image. Later work on Telugu OCR primarily followed the featurization-classification paradigm. Combinations like ink-based features with nearest class centroid (Negi, Bhagvati and Krishna, 2001); ink-gradients with nearest neighbours (Lakshmi and Patvardhan, 2002); principal components with support vector machines (Jawahar, Kumar and Kiran, 2003); wavelet features with Hopfield nets (Pujari et al., 2004) were used. More recent work in this field is centred around improving the supporting modules like segmentation, skew-correction and language modelling. While our work was under review, Google Drive added an OCR functionality which works for Telugu and a lot of other world languages. Although the details of it are not public, it seems to be based on their Tesseract multilingual OCR system (Smith, 2007) augmented with neural networks.

While previous work was restricted to using only a handful of fonts, we develop a robust font-independent OCR system by using training data from fifty fonts in four styles. This data (along with the rest of the OCR program) is publicly released to act as a benchmark for future research. The training and test data are big and diverse enough to let one get reliable estimates of accuracy. Our classifier achieves near human classification rate. We also integrate a much more advanced language model, which also helps us recover broken letters. Our system performs better than the one offered by Google, to the best of our knowledge, the only other publicly available OCR for Telugu.In our work, we break from the above mentioned ‘featurize and classify’ paradigm. We employ a convolutional neural network(CNN), which learns

the two tasks in tandem. In addition, a CNN also exploits the correlation between adjacent pixels in the two dimensional space (LeCun et al., 1998). Originally introduced for digit classification (a sub-task of OCR), CNNs have been adapted to classify arbitrary colour images from even a thousand classes (Krizhevsky, Sutskever and Hinton, 2012). This was aided in part by better regularization techniques like training data augmentation and by increased computing power. We review the new technological “tricks” associated with Deep Learning, and apply them and some of our own in developing a CNN for our task.The rest of the paper is organized as follows. We introduce the problem by describing the Telugu language and its calligraphy. The segmentation process and generation of training data are also explained. CNNs in complete detail elaborated on our architecture. Results and regularization are discussed. The language model, including the recovery of broken letters, is described. We conclude with an evaluation of the end-to-end system. The Problem. Telugu is a Dravidian language with over 80 million speakers, mainly in the southern India state of Andhra Pradesh. It has a strong consonant-vowel structure, i.e. most consonants are immediately followed by a vowel. A consonant and a vowel combine to give a syllable.

For example the syllable ka in the word sakarma in Figure 2.2.1 is one such entity, and ra in the same figure is another. Each such syllable is written as one contiguous ligature. This alpha syllabic form of writing is called an abugida as opposed to an alphabet. There are 16 vowels and 37 consonants which combine to give over 500 simple syllables. Of them about 400 are commonly used. There are nearly sixty other symbols including vowel-less consonants (m in Figure 1), punctuation and numbers.

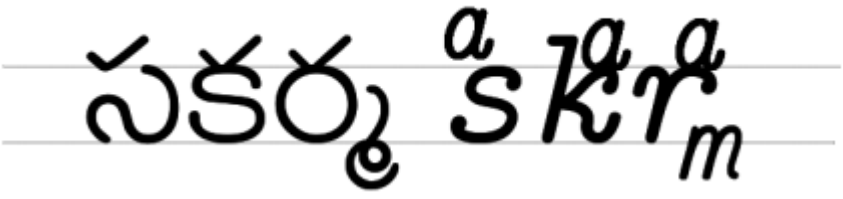


Fig 2.2.1: Unicode Text for Telugu Characters

Figure 2.2.1. The word సకర (sakarma) rendered in Telugu and in English with Telugu styling. The ✓ corresponds to the vowel a, usually attached to the consonant. The syllable స(sa) is written in two parts, క(ka) in one, and ర(rma) is split into ర(ra) and ◌ం(m).Our goal is to develop an end-to-end system that takes an image of Telugu text and converts it to Unicode text.. Segmentation. Given a binary image of text, where 'ink' is unity and background is zero, we first find its row-ink-marginal, i.e. the running count of the number of pixels that are ON. The image is skew corrected to maximize the variance of the first differential of this marginal. That is, we find the rotation that results in the most sudden rises and falls in the row pixel-counts .

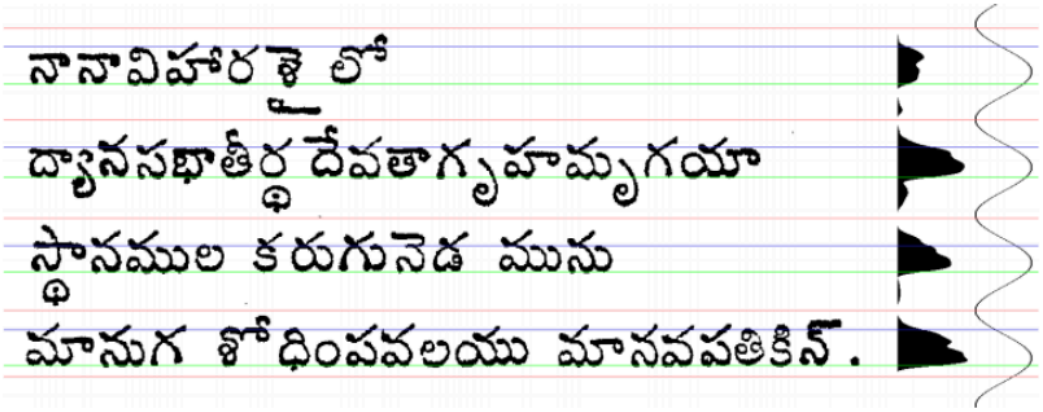


Fig 2.2.2: Text detection and Line detection

2.3 A Convolutional Recurrent Neural-Network-Based Machine Learning for Scene Text Recognition Application

- **Authors** : Yiyi Liu, Yuxin Wang, and Hongjian Shi

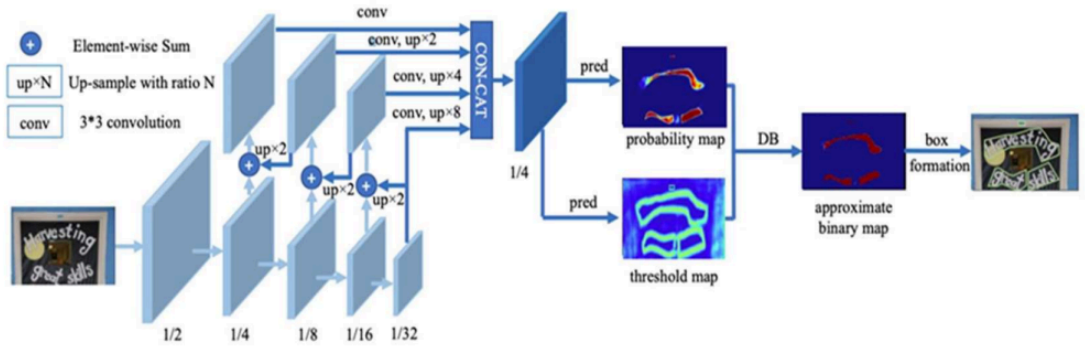


Fig 2.3.1: Convolutional Recurrent Neural Network

Figure 2.3.1 illustrates three distinct threshold maps to highlight how employing a border- type threshold map benefits the outcomes. It clearly shows that the projected threshold graph can differentiate several areas without text, text boundary, and text core, even without the supervision of the threshold graph. The boundary obtained by the threshold graph is more precise than that in Figure 3d, which depicts the loss of the image with the addition of the word boundary.

2.3.1. Label Generation

The label for the approximate binary map and the probability map are the same. This label generation uses the Vatti pruning technique to condense the text with the PSENet kernel concept [10]. We reduce the size of the probability graph label, which is the original standard text box Gt to Gs . The shrink D's deviation is determined as follows:

$$D = \frac{A(1-r^2)}{L}$$

where A is the polygon's surface area, L is its perimeter, and r is its contraction ratio. K is set to 0.4.

The threshold map with/without supervision. (a) Original image. (b) Probability map. (c) Unsupervised threshold map. (d) Supervised threshold map [3].Labels can be constructed for the threshold graph using a similar procedure. First, with the same offset D, the text polygon Gt is extended to Gd. The text area's border is defined as the space between Gt and Gd , from which the label of the threshold graph can be constructed by figuring out how far Gt is from the nearest line segment. The outcome of the label generation is shown.

Example result of label generation for circular symmetric shape containing text. The annotation of the text polygon is visualized in red lines. The shrunk and dilated polygons are displayed in blue and green lines, respectively.

2.3.2 Text Direction Classification

Paddle text direction classifier is a module that is added between the text detection and recognition modules to deal with text in different directions. It uses a convolutional neural network (CNN) with four fully connected layers to extract features from the input image and classify them into four categories. It chooses the direction with the highest probability score as the final output. The text direction classifier network structure is as follows:

- A CNN backbone with 16 convolutional layers and 4 max-pooling layers;
- A global average pooling layer;
- Four fully connected layers with 256, 64, 16, and 4 neutrons respectively;
- A SoftMax layer for outputting probability scores.

When the image is not 0 degrees, degree classification is utilized. In this case, the text lines found in the image need to be fixed. After text detection, a text line image is obtained. This image is then affine transformed and passed to the recognition model. This study requires training a two-class (0° and 180°).

CHAPTER-3 : PROPOSED METHOD

Develop a neural network model capable of accurately recognizing and classifying various characters present in the language model of Telugu.

The OCR engine works by using the following steps:

Image acquisition

A scanner reads documents and converts them to binary data. The OCR software analyzes the scanned image and classifies the light areas as background and the dark areas as text.

Preprocessing

The OCR software first cleans the image and removes errors to prepare it for reading. These are some of its cleaning techniques:

- Deskewing or tilting the scanned document slightly to fix alignment issues during the scan.
- Despeckling or removing any digital image spots or smoothing the edges of text images.
- Cleaning up boxes and lines in the image.
- Script recognition for multi-language OCR technology

Text recognition

The two main types of OCR algorithms or software processes that an OCR software uses for text recognition are called pattern matching and feature extraction.

Pattern matching

Pattern matching works by isolating a character image, called a glyph, and comparing it with a similarly stored glyph. Pattern recognition works only if the stored glyph has a similar font and scale to the input glyph. This method works well with scanned images of documents that have been typed in a known font.

Feature extraction

Feature extraction breaks down or decomposes the glyphs into features such as lines, closed loops, line direction, and line intersections. It then uses these features to find the best match or the nearest neighbor among its various stored glyphs.

Post Processing

After analysis, the system converts the extracted text data into a computerized file. Some OCR systems can create annotated PDF files that include both the before and after versions of the scanned document.

Steps to perform OCR:

- Preprocess the input image by applying binarization, noise removal, skew correction, and resizing techniques to enhance the quality and uniformity of the image.

- Segment the image into lines, words, and characters using a combination of projection profiles, connected components, and contour analysis. Alternatively, use a segmentation-free approach that directly extracts features from the whole image or patches of the image.

- Extract features from the segmented or whole image using convolutional neural networks (CNNs), which can learn robust and invariant representations of the characters. Use max-pooling and dropout layers to reduce the dimensionality and overfitting of the features.

- Feed the sequence of features extracted by the CNNs to a recurrent neural network (RNN) with long short-term memory (LSTM) cells, which can learn the temporal dependencies and context information of the characters. Use a bidirectional RNN to capture both forward and backward information from the sequence.

- Use a connectionist temporal classification (CTC) loss function to train the RNN-LSTM network, which can handle variable-length inputs and outputs without requiring explicit alignment. CTC can also output a blank symbol to indicate no character or a repeated character in the sequence.

3.1 LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network that can be used in Optical Character Recognition (OCR) to recognize patterns. LSTM is an optical character recognition algorithm that can identify characters from a captured number plate image.

LSTM predicts based on the activation of the memory cell in the previous timestep. LSTM based OCR has low error rates even without language modelling. A Convolutional Recurrent Neural Network (CRNN) model uses a convolutional neural network (CNN) to extract visual features, which are then fed to an LSTM network. The LSTM's output is then mapped to character label space with a Dense layer. The LSTM layers learn the language model of the text.

3.2 CNN CLASSIFIER

A convolutional neural network (CNN) classifier is a type of machine learning model that is used to classify images and videos. CNNs are inspired by the structure of the human visual cortex, which is the part of the brain that is responsible for processing visual information.CNNs are made up of a series of layers, each of which performs a specific task. The first layer of a CNN is typically a convolutional layer, which extracts low-level features from the input image. These features can include things like edges, corners, and textures.

The next layer of a CNN is typically a pooling layer, which reduces the size of the feature maps from the previous layer. This is done by merging neighboring pixels together. Pooling layers help to reduce the number of parameters in the network and make it more robust to noise.The convolutional and pooling layers are typically stacked on top of each other multiple times, with each layer extracting higher-level features from the previous layer. The final layer of a CNN is typically a fully connected layer, which classifies the input image into one of a number of categories.

CNN classifiers are trained on large datasets of labeled images. During training, the network learns to associate the features that it extracts from the images with the corresponding labels. Once the network is trained, it can be used to classify new

images by extracting the features from the images and then using the fully connected layer to predict the label.

CNN classifiers are used in a wide range of applications, including:

- Image classification: Classifying images into different categories, such as cats, dogs, and cars.
- Object detection: Detecting objects in images and videos, such as people, faces, and traffic signs.
- Scene segmentation: Segmenting images into different regions, such as the sky, the ground, and the objects in the foreground.
- Medical image analysis: Detecting and diagnosing diseases in medical images, such as X-rays and MRI scans.

We employ the image classifier Convolutional Neural Network (CNN) to make emotion predictions based on the extracted features. CNN is a powerful and efficient machine learning algorithm known for its speed and accuracy. It's particularly well-suited for handling large datasets and complex feature spaces.

3.3. DATA SOURCES

3.3.1 Base Model

The base model primarily relies on image content from an older collection available on the Internet. The images are larger in size and hence the training time is increased.

3.3.2 Our Approach

In contrast, our approach capitalizes on the images from IEEE Dataport. The dataset is organized into four main categories (achulu, guninthalu, hallulu, otthulu) along with their subcategories.

3.4. DATA VOLUME AND AVAILABILITY

3.4.1 Base Model

The base model's reliance on image data may face challenges in obtaining sufficient and relevant images. The images are larger in size and hence the training time is increased. Also, the images were not pre-processed enough to maintain resemblance throughout the dataset.

3.4.2 Our Approach

Our dataset is well classified and pre-processed to maintain the same resolution and quality throughout the dataset. We have also drastically reduced the image size which enabled us to train our model on a large test dataset containing 11000+ images.

3.5. ACCURACY AND GENERALIZATION

3.5.1 Base Model

The effectiveness of the base model may vary depending on the quality and diversity of image content. It could be limited by the challenges of accurately interpreting images and generalizing findings to a broader user base.

3.5.2 Our Approach

Our approach benefits from the structured nature of text data, which can be analysed with a higher degree of accuracy and consistency. It is better suited for recognition of various characters of the language.

3.6. SPEED AND EFFICIENCY

3.6.1 Base Model

Processing image data without optimization for character prediction can be computationally intensive, potentially leading to longer processing times and increased resource requirements.

3.6.2 OUR APPROACH

Our approach is computationally efficient and faster as we have used Our approach is computationally efficient and faster as we have used a CRNN network (Convolutional-Recurrent Neural Network).

CHAPTER 4 : IMPLEMENTATION

4.1 DATA COLLECTION

We downloaded our dataset from IEEE Dataport. The data consists of 50x50 pixel images of every unique character in Telugu. The task is to categorize each image into a grammar type based on the input image given to the model (achulu, guninthalu, hallulu, otthulu).

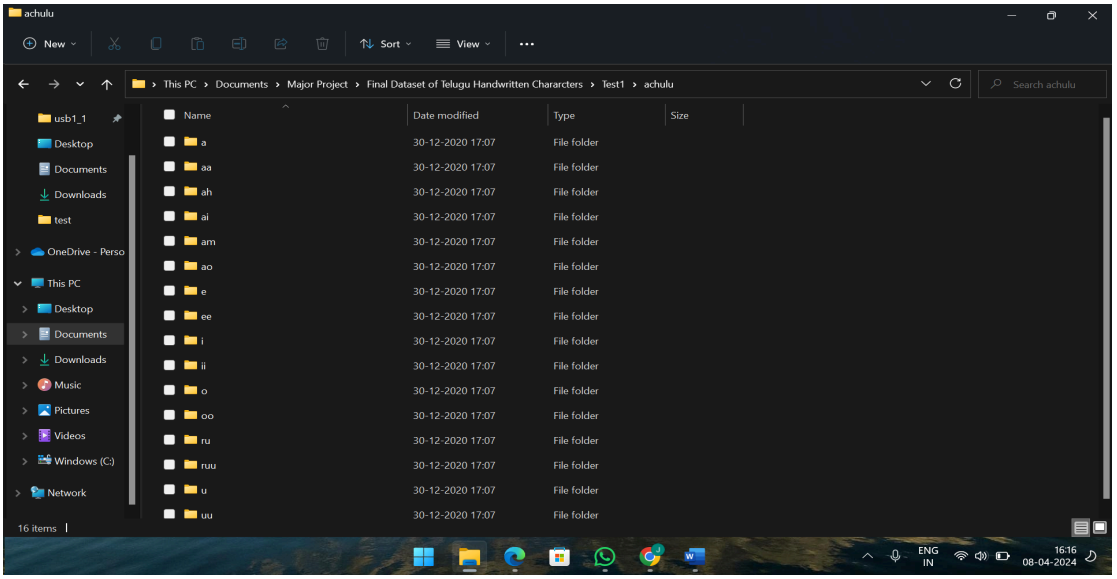


Fig 4.1.1: Labelled Dataset

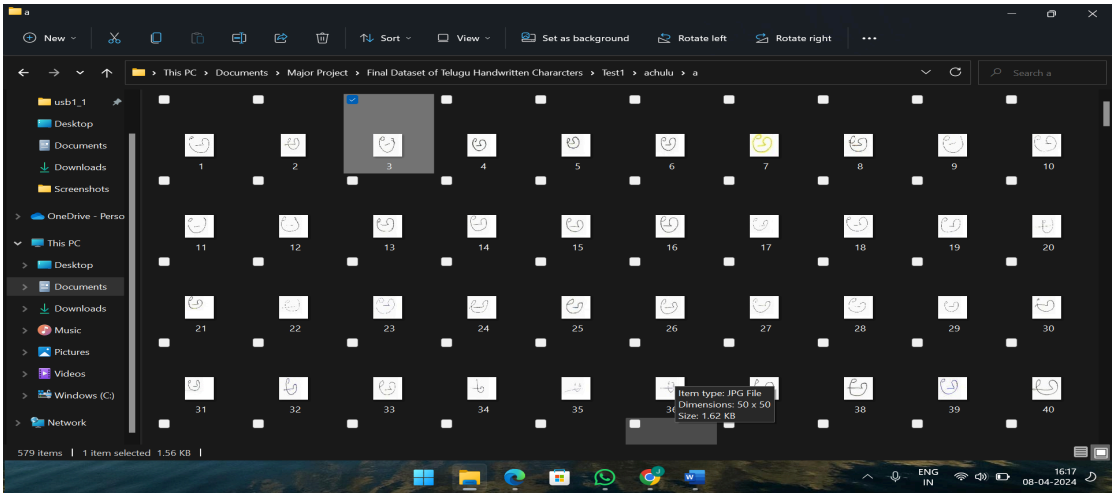


Fig 4.1.2: Images in Dataset

4.2 DATASET

The training set consists of 11,000 examples and the public test set consists of 200 examples. Both the test dataset and the train dataset are categorized into four classes (achulu, guninthalu, hallulu, otthulu).

4.3 PREPROCESSING

All the images are resized into 50 by 50 pixel size.

The following preprocessing steps are performed:

- The images are converted into grayscale(monochrome)

- Dilaton: Dilation is a morphological operation in image processing that expands the boundaries of an object. It's a basic operator in mathematical morphology

- Erosion: Erosion is a fundamental operation in mathematical morphology and image processing. It's one of two basic operators in mathematical morphology, the other being dilation.

- Gaussian Blur: Gaussian blur is a pre-processing stage in computer vision algorithms that reduces image noise and detail. It's a low-pass filter that preserves low spatial frequency and removes speckles. The blur creates a smooth effect that resembles viewing an image through a translucent screen.

- Thresholding: Image thresholding is a technique that simplifies a grayscale image into a binary image by classifying each pixel value as either black or white based on its intensity level or gray-level compared to the threshold value.

4.4 MODEL TRAINING AND TESTING

We divide our dataset into a training set and a testing set. The training set is used to teach the CNN classifier to recognize patterns in the train data that are associated with class wise folders. Once trained, we evaluate the model's performance using the testing set, employing metrics like accuracy, Character Error Rate(CER).We use keras libraries from Tensorflow to train the model. The following libraries are used in our project to perform OCR :

4.4.1 Tensorflow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, including Python, JavaScript, C++, and Java.

TensorFlow serves as a core platform and library for machine learning. TensorFlow's APIs use Keras to allow users to make their own machine learning models. In addition to building and training their model, TensorFlow can also help load the data to train the model, and deploy it using TensorFlow Serving.

4.4.2 Keras Layers

Keras layers are the basic building blocks of Keras models. A layer consists of a tensor-in tensor-out computation function (the layer's call method) and some state, held in TensorFlow variables (the layer's weights).

Keras layers are divided into two main categories: core layers and specialized layers. Core layers are the most basic types of layers, such as dense layers, convolutional layers, and pooling layers. Specialized layers are designed to perform specific tasks, such as image classification, natural language processing, and computer vision.

Here are some examples of core Keras layers:

- *Dense layers*: Dense layers are fully connected layers, which means that each neuron in the layer is connected to every neuron in the previous layer. Dense layers are used for a variety of tasks, such as classification, regression, and dimensionality reduction.
- *Convolutional layers*: Convolutional layers are used to extract features from images and videos. Convolutional layers are typically used in computer vision tasks, such as image classification, object detection, and scene segmentation.
- *Pooling layers*: Pooling layers are used to reduce the size of feature maps from convolutional layers. Pooling layers also help to make convolutional networks more robust to noise.

Illustration: Think of model training as a student. The student (CNN) learns from a textbook (the training data) to understand the material (emotion patterns). Once the student is well-prepared, they take an exam (testing data) to prove how well they've learned. The exam results (metrics) determine how effective the teaching process has been.

In our code, we are performing data collection, preprocessing, and using machine learning models to Telugu characters. The best-performing model appears to be the CRNN model. The user interface is defined in the code to allow users to interact with the system.

4.4.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source software library designed for computer vision and machine learning applications.

Originally developed by Intel, OpenCV is now maintained by a community of developers under the OpenCV Foundation. It provides a common infrastructure for various computer vision tasks and accelerates the use of machine perception in commercial products.OpenCV boasts over 2500 optimized algorithms that cover both classic and state-of-the-art computer vision and machine learning techniques.

4.4.4 Numpy

NumPy (pronounced /ˈnʌmpaɪ/ NUM-py) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.[3] The predecessor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

4.4.5 Tkinter

Tkinter is a Python library that provides a Python binding to the Tk GUI toolkit, making it the standard Python interface to the Tk GUI toolkit. It's open-source and comes included with standard Linux, Microsoft Windows, and macOS installs of Python. Tkinter is known for its simplicity and graphical user interface (GUI).

Tkinter calls are translated into Tcl commands, which are fed to the embedded Tcl interpreter, thus making it possible to mix Python and Tcl in a single application.

4.4.6 Pytesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images. Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg,

png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

4.5 UML DIAGRAMS

A UML diagram is a way to visualize systems and software using Unified Modeling Language (UML). Software engineers create UML diagrams to understand the designs, code architecture, and proposed implementation of complex software systems. UML diagrams are also used to model workflows and business processes. Coding can be a complicated process with many interrelated elements. There are often thousands of lines of programming language that can be difficult to understand at first glance. A UML diagram simplifies this information into a visual reference that's easier to digest.It uses a standardized method for writing a system model and capturing conceptual ideas.

4.5.1 Class Diagram:

Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams

- Shows static structure of classifiers in a system
- Diagram provides a basic notation for other structure diagrams prescribed by UML
- Helpful for developers and other team members too
- Business Analysts can use class diagrams to model systems from a business perspective

A UML class diagram is made up of:

- A set of classes and
- A set of relationships between classes.

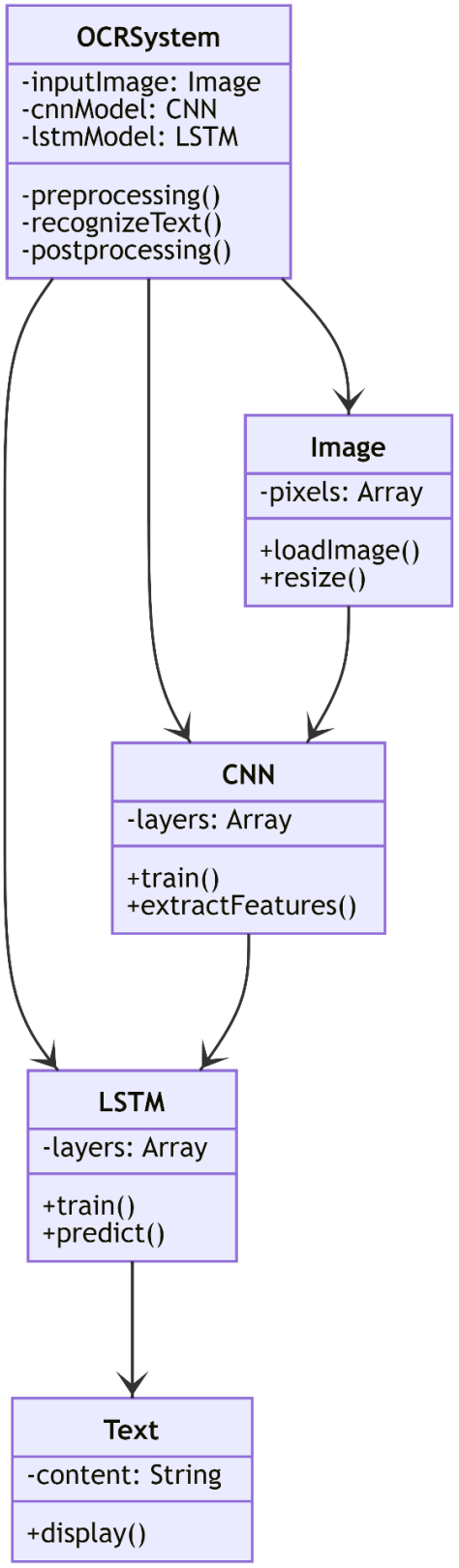


Fig 4.5.1: Class Diagram for Proposed Method

4.5.2 Sequence Diagram:

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time.In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

Purpose of a Sequence Diagram12

1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.
3. It either models generic interactions or some certain instances of interaction.

The User initiates the process by interacting with the Webpage object. Upon entering a URL for prediction, a message is sent from the Web Page to the Flask object, which serves as the backend server handling the requests.Once the Flask object receives the URL request, it triggers the Feature Extraction object to process the data. After the features are extracted, the processed data is then utilized by the Prediction object. This prediction is made using the XGBoost Model object, which represents the trained machine learning model.Additionally, the sequence diagram suggests that the User can perform more advanced interactions with the System. The User can send a message to the Flask object to Load ML Model or Train ML Model, indicating the ability for the user to manage and modify the machine learning model as required.

In summary, your sequence diagram showcases the flow of messages and interactions between the User, Webpage, Flask, Feature Extraction, XGBoost Model, and Prediction objects, highlighting the steps involved in data processing, prediction, and potential model management within the system.

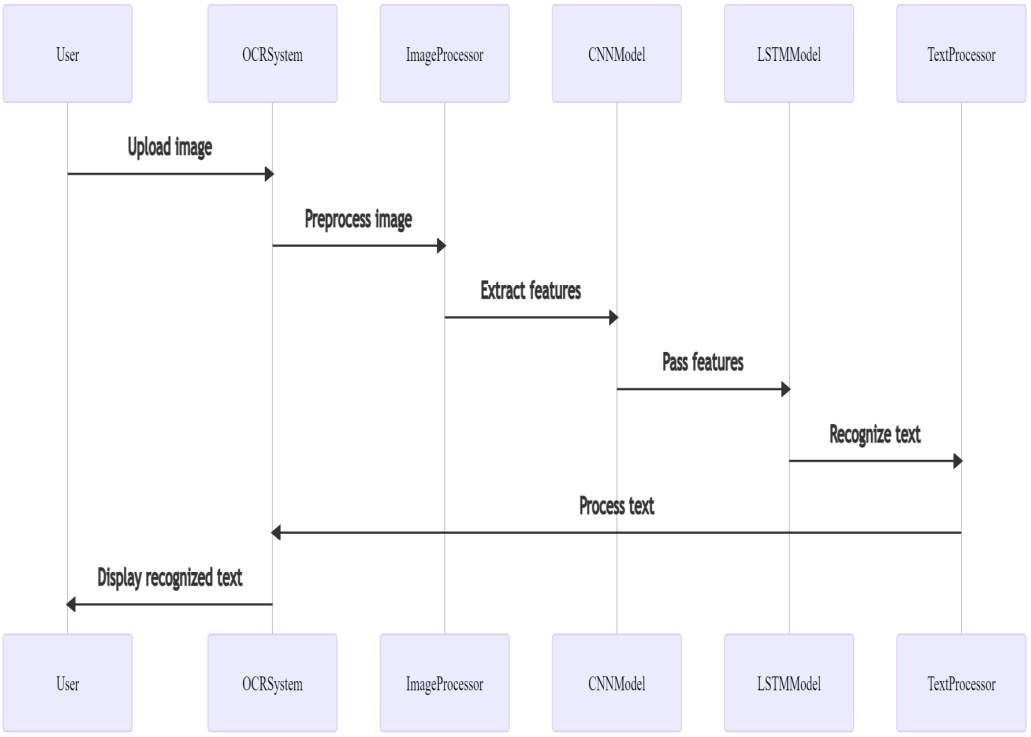


Fig 4.5.2: Sequence Diagram for Proposed Method

CHAPTER-5 : EXPERIMENTAL OBSERVATIONS

5.1 EVALUATION METRICS:

We used accuracy and Character Error Rate(CER) as the primary evaluation metrics to assess the performance of the models. These metrics provide insights into the accuracy of character predictions and the trade-off between character error rate and word error rate.

Train-Test Split: The dataset is split into training and testing sets 80-20 ratio, to evaluate the models' generalization performance.

5.2 PARAMETERS

In the context of deep learning models, there are various parameters and formulas that impact the model's behaviour and performance. Here are some parameters and formulae that you may have encountered in our project:

True/False, Positive/Negative.

Before we start looking at the metrics we first need to establish some basics. A benefit of the six images(dataset) we have is that we know the associated category, animal or not animal for each image. If we put these into a table with predicted categories as columns and actual categories as rows we have made ourselves a confusion matrix.

Confusion matrix:

The confusion matrix together with the terms in the cells (true positives/negatives and false positives/negatives) are super important for understanding the metrics later. Luckily they are quite simple to understand when we populate them with examples.

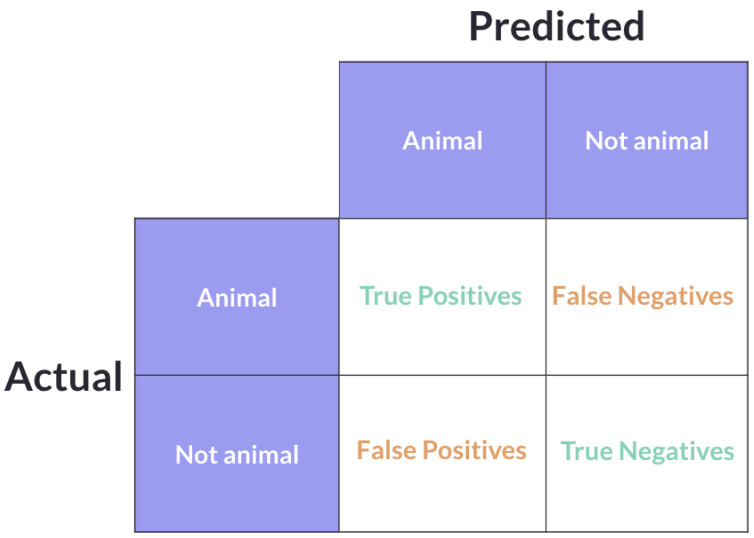


Fig 5.2.1: Confusion matrix

Accuracy

Accuracy is the most common metric to be used in everyday talk. Accuracy answers the question **"Out of all the predictions we made, how many were true?"**

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ positives}$$

As we will see later, accuracy is a blunt measure and can sometimes be misleading.

Character Error Rate(CER)

Character Error Rate (CER) is a metric that measures the percentage of characters that have been incorrectly transcribed by a model.

Character Error Rate (CER):

$$CER = \frac{\text{Number of incorrect characters}}{\text{Total number of characters in the reference text}} \times 100\%$$

5.3 User Interface (Experiment Screenshots):

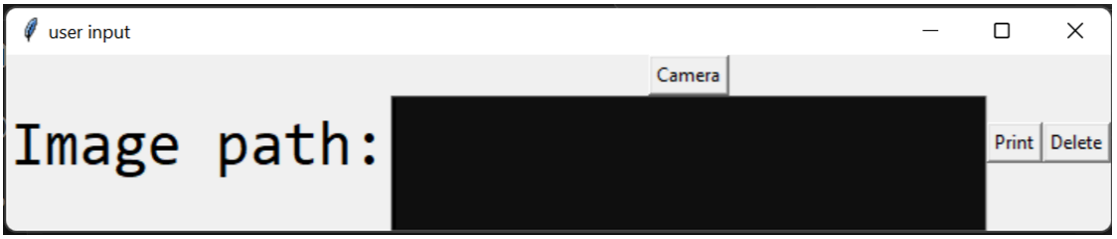


Fig 5.3.1 Tkinter GUI

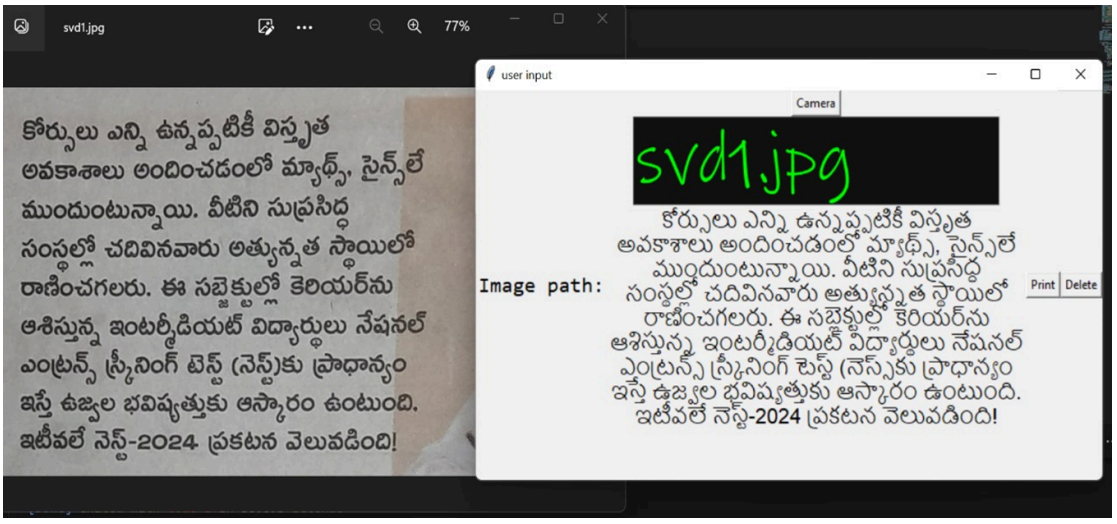


Fig 5.3.2 OCR performed on input image(newspaper)

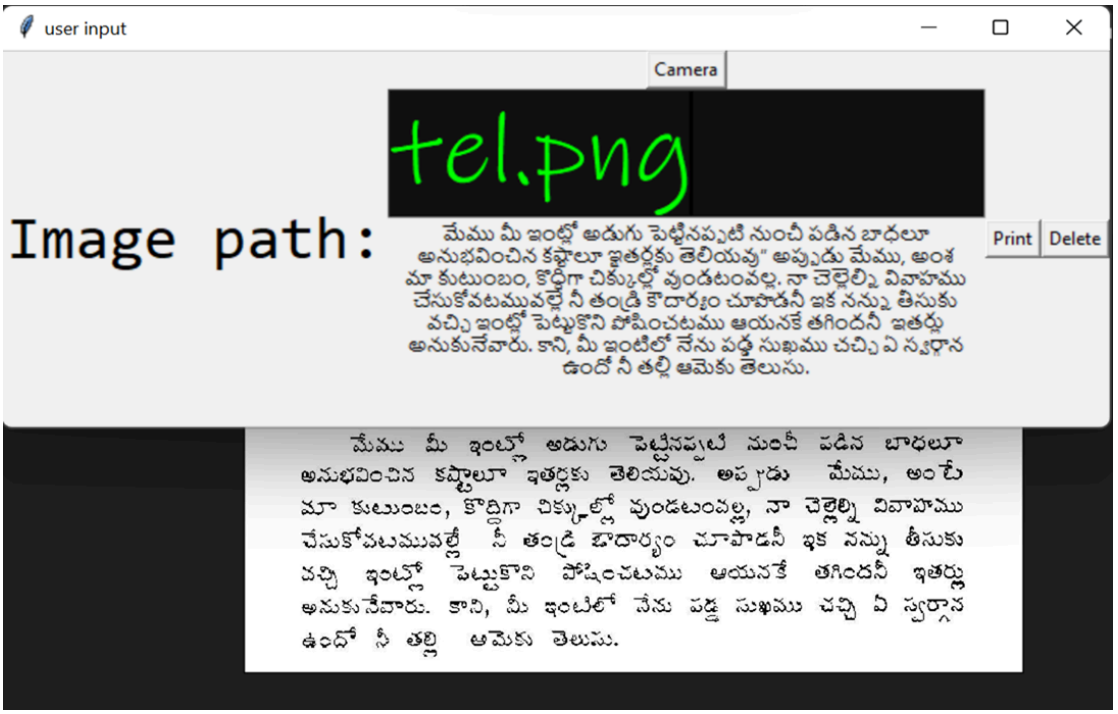


Fig 5.3.3 OCR performed on image with computerized text

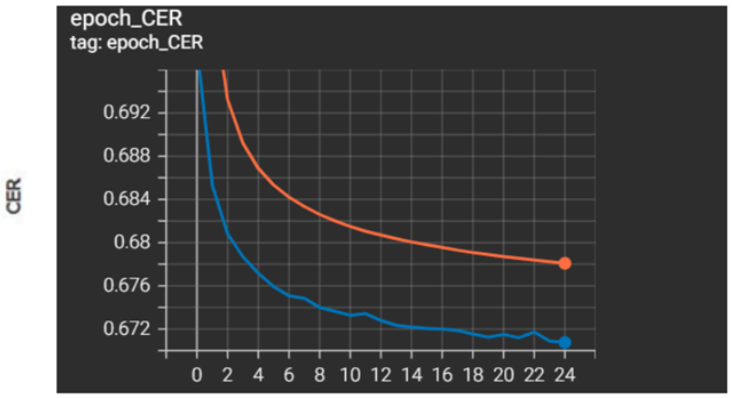
CHAPTER-6: DISCUSSION OF RESULTS

6.1 EVALUATION

In this project, we created a system that can recognize optical characters from a digital image using a special type of computer program called a Convolutional-Recurrent Neural Network (CRNN). We tried different methods, like using TensorFlow and LSTM, to see which one worked best for making predictions and classifying these activities.

Character Error Rate (CER) is a metric that measures the percentage of characters that are incorrectly transcribed by a model. It is based on the distance between a perfect transcription and the automatically recognized text. The CER is calculated by dividing the total number of incorrect characters by the total number of characters in the reference text. A lower value indicates better ASR system performance. A CER of 0 indicates a perfect score.

6.2 ACCURACIES:



We applied test on 1000 examples using print(f"Average CER: {np.average(accum_cer)}") which gave the following:

Average CER	7%

CHAPTER-7 : SUMMARY, CONCLUSION, RECOMMENDATIONS

7.1 SUMMARY

In this project, we developed an optical character recognition system based on the Convolutional-Recurrent Neural Network(CRNN) framework. We implemented and evaluated multiple deep learning techniques like tensorflow keras, CTC network to make predictions and classifications. Here, we summarize our findings and provide conclusions and recommendations based on the results.

7.2 FINDINGS

1. **Model Comparison**: We tested several models, and our findings reveal variations in their performance metrics. The CRNN classifier consistently outperformed other models, achieving lower CER(Character Error Rate).
2. **Accuracy**: Character Error Rate (CER) is a metric that measures the percentage of characters that are incorrectly transcribed by a model. It is based on the distance between a perfect transcription and the automatically recognized text. The CER is calculated by dividing the total number of incorrect characters by the total number of characters in the reference text. A lower value indicates better ASR system performance. A CER of 0 indicates a perfect score.

7.3 CONCLUSION

In conclusion, our project has successfully demonstrated the feasibility of predicting characters of Telugu language from images captured from a camera source. The CRNN model emerged as the most effective in terms of accuracy and F1-score. This suggests that our approach offers a robust solution for OCR.

7.4 RECOMMENDATIONS

Future Research

We have implemented an OCR system with a CRNN network. We have observed the major problems causing the poor accuracy are font dependency, joint characters and distorted characters. In stage two we have experimented with different types of feature vectors for solving font dependency. We have extracted features using wavelet features, gabor features and circular zonal features. We have observed Telugu characters and got better accuracy using wavelet feature vectors. And also observed circular zonal features with density are performing worst. We have repeated the same feature extraction methods on skeleton images generated. We have observed decrease in accuracies in case of gabor and wavelet features, but in case of circular zonal features there was a very good improvement in accuracies. From stage 2 results we can conclude wavelet features are giving better accuracy and are solving font dependency problems compared to other methods.

We have tried solving other problems in OCR, joint character problems. We have noticed joint character problem can be solved by finding co-efficient matrices for with which we can convolute the basis image and their sum will give the reconstructed image. Our objective is to minimize reconstruction error and solve for co-efficient matrices. Paper by Morten is solving a similar problem in the case of NMF. We have used their code and implemented a solution for our problem. The observation made by us is, we can solve the joint character problem using this when we have a huge training dataset. We have observed the number of joint characters are less in our dataset which is causing less improvement in our accuracy i.e less than 1%. We have observed the time taken for solving joint characters is 20more. We have used similar formulations with change in the regularizer for improving the solution.

We have noted slight improvement and this can also be used for font independence. We even observed another major problem for poor accuracy is distorted characters. Distorted characters caused due to bad printers or noise, which can not be solved completely using general font independence methods. But we observed one more reason for distorted characters in case of low resolution input images, binarization. We have used Otsu's method of binarization which is also

causing some broken characters. We have observed that using adaptive thresholding methods can decrease the number of broken/distorted characters.

Our future work involves improving accuracy by trying more types of feature vectors. We also need to try using combinations of the different types of feature vectors. We need to improve the solutions for joint character problems in terms of execution time. We need to find some optimizations, by observing the dataset we can say there are repetitions in joint characters which we can use for giving weights for some of the basis images and optimize the execution time. We can use additional heuristics for optimizing the solution. We have observed broken characters can be caused by binarization methods also. We need to experiment more on different binarization methods to decrease the number of broken characters. We have to apply some more filters which can remove remaining noise from the image after our noise removal step.

1. **Ethical Considerations**: As this system involves processing personal data, it's crucial to prioritize ethical considerations, including privacy, consent, and data security. Ensure that the system complies with all relevant data protection regulations.

2. **Scalability**: Consider the scalability of the system to accommodate a growing user base and increasing data volumes. Optimize the system's efficiency to handle a large number of predictions.

7.5 JUSTIFICATION OF FINDINGS

Our findings are justified by a thorough evaluation of multiple machine-learning models, focusing on accuracy as performance metrics. The consistently superior performance of the CRNN model across these metrics indicates its reliability in personality prediction. The findings are based on a well-structured dataset and a robust preprocessing approach, making them suitable for practical implementation.

CHAPTER-8: REFERENCES

1. Optical Character Recognition for Handwritten Telugu Text by B Revathi · 2023
2. Telugu Optical Character Recognition Using Deep Learning by G Suresh · 2022
3. Offline Handwritten Basic Telugu Optical Character Recognition by B Kalpana · 2023
4. Optical Character Recognition (OCR) for Telugu by KC Prakash · 2018
5. Implicit Language Model in LSTM for OCR by Ekraam Sabir . 2018
6. A Convolutional Recurrent Neural-Network-Based Machine
7. Learning for Scene Text Recognition Application by Yiyi Liu,Yuxin Wang, and Hongjian Shi . 2023
8. Rakesh Chandra Balabantaray, Surajit Mohanty, Marimuthu Karuppiah, Debabrata Samanta (2022). Approach for Preprocessing in Offline Optical Character Recognition (OCR).
9. Liu, Y., & Yao, J. (2018). Character-level attention for text recognition in scene images. 2018 13th International Conference on Document Analysis and Recognition (ICDAR), 1-8.
10. Wang, Z., Li, W., & Wang, C. (2019). EAST: An efficient and accurate scene text detector. 2019 14th IAPR International Conference on Machine Learning and Applications (ICMLA), 1155-1160.
11. Jiao, Y., He, S., He, Y., & Yang, L. (2020). An end-to-end approach for scene text recognition with parallel attention mechanisms. Pattern Recognition, 101, 107190.
12. Luo, C., Wang, Z., Li, W., & Qiao, Y. (2020). CTPN: Cascaded proposal network for robust scene text detection. IEEE Transactions on Image Processing, 29(12), 6376-6388.
13. Handwritten Text Recognition: Nguyen, T. Q., Liwicki, M., Bunke, H., & Tran, Q. H. (2018). Bounding box regression with iterative refinement for handwritten text recognition. 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), 1-6.

14. Su, H., Li, H., Yao, C., & Zhou, X. (2019). A comprehensive survey of end-to-end handwritten text recognition. Neurocomputing, 397, 138-152.
15. Bluche, L., Epshtein, B., Vincent, N., & Mamalet, F. (2020). Handwritten text recognition with recurrent neural networks. Pattern Recognition Letters, 133, 246-253.
16. Puigcerver, J., Toselli, A. G., & Montesinos, P. (2021). Handwritten text recognition using attention-based recurrent neural networks. Pattern Recognition Letters, 140, 178-184.
17. Su, T., Li, Y., Zhang, H., & Wang, X. (2018). A novel method for correcting text recognition errors based on character n-gram language model. 2018 13th International Conference on Document Analysis and Recognition (ICDAR), 133-138.
18. Liu, K., Liang, D., Wang, L., & Pan, S. J. (2019). An analysis of ocr errors: Towards improving ocr robustness with deep learning. 2019 14th IAPR International Conference on Machine Learning and Applications (ICMLA), 1089-1094.
19. Gupta, A., Sharma, A., Gupta, A., & Dogra, A. (2020). A survey of text denoising techniques in ocr: A deep learning perspective. Pattern Recognition, 100, 107061.
20. Liu, X., Yin, F., & Zhang, Y. (2021). An ocr error correction method based on attention mechanism and sequence-to-sequence learning. 2021 International Joint Conference on Neural Networks (IJCNN), 1-8.
21. Gupta, H., Gupta, A., Prabhakar, S., & Sharma, A. (2018). Text extraction from historical document images: A survey. Pattern Recognition, 80, 1-24.
22. Shafait, B., & Santos, P. (2018). Applying optical character recognition to historical manuscripts: A survey. International Journal on Document Analysis and Recognition, 21(1), 99-123.
23. Chen, X., Yao, J., & Zhou, Z. (2019). An ocr-based approach for music score analysis. 2019 14th IAPR International Conference on Machine Learning and Applications (ICMLA), 1079-1084