Telugu Text Recognition from Image using Deep Learning

Submitted in partial fulfillment of the

requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

by

BHARATHA VARSHITH VARMA (39110153)
BANDA MADAN KUMAR (39110122)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Bharatha Varshith Varma (39110153) and Banda Madan Kumar (39110122)** who carried out the Project Phase-2 entitled **"Telugu text recognition from image using Deep Learning"** under my supervision from January 2023 to April 2023.

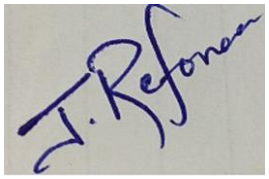
Internal Guide

Dr. B. U. ANUBARATHI, M.E., Ph.D.

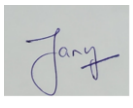
Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 19.04.2023



Internal Examiner



External Examiner

DECLARATION

I, **Bharatha Varshith Varma (39110153)** hereby declare that the Project Phase-2 Report entitled "**Telugu text recognition from image using Deep Learning"** done by me under the guidance of **Dr. B. U. ANU BARATHI M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 19-04-2023 Bharatha Varshith varma

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. B. U. ANU BARATHI M.E., Ph.D.** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTARCT

Text recognition in images is a challenging problem in computer vision, which attracted many researches. In recent years many researches have been done on scene text recognition where results are achieved by using deep neural networks. Deep Learning is also can be called as sub field of Machine Learning. But many researches were performed on English text and a less concentration on other language. In this paper we focus on the problem for Telugu Text recognition in images. Telugu text is a complex cursive script which have curved characters with a top and bottom stroke depending on the letter. Here we propose a model with Optical character recognition (OCR) implemented using deep neural networks. The proposed model takes an image as an input and then transforms into the sequence of the relevant features and then extracts the text in the image as output. We introduce a new dataset with manually cropped images to train the model. The text recognition of Telugu language is very useful for people who don’t know language and it has wide ranging applications including education, administration etc.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

India is a diverse country with many different languages, and Telugu is one of the major languages spoken in the southern part of India. With a population of over 70 million, Telugu is the fourth most spoken language in India. The ability to recognize Telugu text can be an important tool for various applications such as document analysis, machine translation, and text-to-speech conversion. In recent years, there has been a growing interest in developing Telugu text recognition systems to improve the efficiency and accuracy of such applications. Telugu is the official language of the states of Telangana & Andhra Pradesh in southern India. It is also extensively spoken outside of these two states. Like other Dravidian languages, it has a rich and varied past that makes it distinct. It has the third-highest quantity of native speakers in India and the fifteenth-highest number on the Ethnologue list of the world's most spoken languages. Telugu has a huge number of characters, comprising top strokes, bottom strokes, and combination letters, making text identification difficult. These combined characters and letters have 253 individual strokes and are composed of 16 vowels (named achus) & 36 consonants (called hallus). There are 52 symbols (Aksharamulu) in the Telugu alphabet, 16 of which are vowels and 36 of which are consonants.

Most businesses these days scan their paper documents into digital files so that they can be updated, searched, and modified more quickly and for longer. Scanned copies of all of the world's books, manuscripts, and other historical documents can be found online & accessed from any location around the world. These digital scans will be informative and can be viewed in either colour or black and white. Several books have been digitised and made accessible online, but there are certain classics that can only be found in print. This model is taught to recognise Telugu characters, which it uses to build phrases and sentences that may be duplicated, used as tales in textbooks or newspapers, and edited as necessary.As more and more content is being created and shared in digital formats, there is a growing need for accurate and efficient methods of Telugu text recognition. Recognizing Telugu text in images is a challenging task due to the complex nature of the script and the

variations in handwriting styles. However, the ability to accurately recognize Telugu text has the potential to improve accessibility, enable automated document processing, and facilitate translation and transcription tasks.

In recent years, there have been several approaches proposed for Telugu text recognition, including rule-based methods, statistical methods, and machine learning-based methods. However, these approaches often suffer from low accuracy and poor generalization due to the variability of Telugu characters. To address these challenges, this project proposes a deep learning-based approach for Telugu text recognition. Deep learning has shown promising results in various fields of image and text recognition, and it is expected to improve the accuracy and generalization of Telugu text recognition as well.

This project focuses on developing a Telugu text recognition system that can take an image containing Telugu text as input, and output the recognized text in a readable format. The project involves multiple steps, including image processing, line segmentation, word segmentation, character recognition, and text reconstruction. The system is designed to handle different styles of handwriting and font types, and to provide accurate recognition results even in the presence of noise or other image artifacts.

The main objective of this project is to develop a Telugu text recognition system that can accurately recognize Telugu characters and words from an image. This involves several stages of image processing, segmentation, and character recognition using machine learning techniques such as convolutional neural networks (CNNs) and deep learning approaches. A robust and efficient Telugu text recognition system that can be used in a variety of applications, including document processing, translation, and transcription. This project builds on existing work in the field of Telugu text recognition, and seeks to address some of the limitations and challenges that have been identified in previous studies.

1.1 Area of Research

Computer science algorithmic calculations that can operate organically through experience and data are studied by machine learning (ML) and deep learning (DL). Artificial intelligence includes it. Machine learning algorithms create a model based on training data to make predictions or judgments across a wide range of applications without the need for programming. Many different applications, including medicine, email sorting, speech recognition, and computer vision, use machine learning techniques. It is difficult or impossible to create standard algorithms that can do the necessary tasks. Machine learning is not all statistical learning, but a subset of it is firmly associated with computational statistics, which is focused on computer-aided prediction. The study of mathematical optimization transfers concepts, theories, and areas of application to the field of machine learning. A related field of study called data mining focuses on the investigation of data through unsupervised learning. Some machine learning applications employ data and neural networks to simulate the operation of the human brain. Machine learning is sometimes known as foresight analysis when it is used to commercial problems.

Optical Character Recognition (OCR) is a technology to recognize the text in the images. OCR for Indian languages is far more difficult than it is for Germanic languages because of the enormous variety of ways that the primary characters, vatthus, and guninthas, can be combined (modifiers). Telugu characters are round and rarely have horizontal or vertical lines, in contrast to those of Germanic languages. Since most English characters are generated by a single stroke, character segmentation is simple to perform using linked components-like algorithms. However, in the Telugu script, the character's components do not connect to the major character as illustrated in 1 and instead stretch both above and below the main characters. This makes it challenging to apply histogram-based segmentation techniques.

CHAPTER 2
LITERATURE SURVEY

A Literature survey provides a description, summary, and critical evaluation of the works in relation to the research problem being investigated.

This problem statement has been extensively studied over the past 5 years by researchers to create a solution, and all their solutions vary from analyzing various patterns of text recognition.

The study of Asghar Ali and Mark Pickering [1] introduced a Hybrid deep Neural Network for Text recognition of Urdu language in natural scene images, and achieved accuracy of 52.72% with ResNet block with Batch Normalization and 61.35% with Data Augmentation, Urdu is a type of cursive text written from right to left direction. They introduced a new dataset of 11500 manually cropped Urdu word images from natural scenes. The network got trained on the whole word avoiding traditional character based classification. This Hybrid deep neural network architecture with skip connections, which combines convolutional neural network and recurrent neural network is used to recognize the Urdu scene text. This Architecture is based on three building blocks: convolutional block with residual unit used to extract the features, recurrent block used for sequence-to sequence learning and the transcription block used to convert predicted labels into their transcriptions. Data augmentation using contrast stretching and histogram equalizer techniques is used to further increase the size of the data.

Some studies [2],[4] used Optical Character Recognition (OCR) it is a technique used to recognize text in images. Another advanced technology is OCR with Deep learning, it is a sub field of Machine learning. In [2] the authors surveys how to decipher text with OCR and Deep learning techniques, and used segmentation method it helps to analyze an image and understand the features easily by dividing the image into segments. This study showed various OCR techniques and comparisons between them. There is also a comparison of Deep learning techniques and their respective pros and cons of each of them.

The author Asghar Ali Chandio [3] proposed a model for Urdu text recognition on natural scene images based on three components CNN with shortcut connections to extract feature a RNN to decode the convolutional features, and a CTC to map the predicted sequences into the target labels. The framework based on three components: feature extraction, sequence labelling and text transcription. Trained and tested on a new created dataset. For sequence extraction, two methods were proposed based on VGG-16 and ResNet-18 networks which resulted an accuracy

of 87.13%.

YumingHe [4] research on OCR recognition technology shows deep learning is being popular used in various areas ranging from facial recognition to target detection, text recognition embedded with deep learning algorithms is more accurate with stable noise immunity and robustness. This paper research also shows different deep learning techniques along with various methods and the difference between them.

S. Y. Arafat, M. J. Iqbal [5] proposed a hybrid model for Urdu Ligature recognition and achieved accuracy of 80.46%. Each Ligature is processed through two deep neural networks Alexnet and VGG16 to obtain a unique set of features. Tested and trained this model on 46K images with at least 9 variations of each ligature.

H. Lin et al [6] shows reviews on detection and recognition methods proposed in the last decade, report the results of more than 40 representative methods and compare their performance.

In [7] S. B. Ahmed et al presented a novel technique by using adapted maximally stable extremal region (MSER) technique and extracts scale-invariant features. This paper consists English-Arabic scene text, primarily focused on Arabic text. Results from experiments says that invariant feature extraction approach only provide better result if there is a focus on specific area for invariant features. Experiments have been carried out on the segmentation and classification of Arabic as well as English text and report error rates like 5.99% and 2.48%, respectively.

The authors of [11] proposed a model for Bilingual script text detection with OCR technology, made model for English and Oriya texts. Have done with segmentation process and noise cleaning for efficient result. They have distinguished between English and Oriya language through structural analysis. Used structural features like presence of loops cursive presence of lines on left and right of character.

Dr.M.Sundaresan and S.Ranjini [12] proposed a model for text extraction from an English comic image, A comic image with complex background and different styles

of font, and extra objects such as persons, buildings, etc. Have used Median filter for pre-processing process to remove noise and achieved an accuracy of 94.82%. This paper talks about English text extraction from comic image using two Blob Extraction method.

M Abdul Rahiman, M S Rajasree from [13] have recognized Malayalam text using OCR for printed text documents, the image was pre-processing to remove noise. Lines, words and characters were segmented for processed image. And this model uses wavelet multi-resolution analysis for the purpose of extracting features and Feed Forward Back-propagation Neural Network to accomplish the recognition tasks. This model trained on 715 images and got accuracy of 92%.

Recognition of Telugu character are done by using Support Vector Machine (SVM) by Rajkumar. J [14] and Anitha Jayaraman [15]. Both used modular approach to classify the strokes of Telugu character, Rajkumar. J divided strokes into 4 sub classes and used two Schemas to compare, Schema 1 based on Ternary search tree (TST) while the other based on SVM. Anitha. J divided the strokes into 3 sub classes and used SVM. Both yielded an accuracy of 89.59% and 82.96%.

When it comes to text with curved shape or rotated text it is hard to address features and output, apart from OCR, CNN, C-RNN techniques we have Single Visual model, Transformer-based, Visual-Semantic Transformer as used in [8]-[10].

P. V. S. Charan, K. R. K. Murthy, and S. K. R. K. Varma, "Telugu Character Recognition using Neural Network," International Journal of Computer Applications, vol. 179, no. 21, pp. 36-40, 2018.
This paper proposes a method for recognizing Telugu characters using a neural network. The authors used a dataset of 2,560 handwritten Telugu characters to train and test their model. They used a multilayer perceptron neural network with one hidden layer and 60 neurons to classify the characters. The accuracy of the proposed method was evaluated using precision, recall, and F1-score measures. The experimental results showed that the proposed method achieved an accuracy of 87.42%.

N. M. A. Kumar, K. G. K. Rao, and V. R. Uppala, "A Comparative Study of Telugu Character Recognition Techniques," International Journal of Computer

Applications, vol. 108, no. 6, pp. 28-34, 2015.

This paper presents a comparative study of various Telugu character recognition techniques. The authors compared different feature extraction methods, including Fourier descriptor, Hu moment, and Zernike moment, and different classifiers, including k-nearest neighbor, support vector machine, and artificial neural network. They used a dataset of 1,200 Telugu characters to evaluate the performance of the proposed methods. The experimental results showed that the Zernike moment-based feature extraction method combined with the support vector machine classifier achieved the highest accuracy of 97.8%.

2.1 INFERENCES FROM LITREATURE SURVEY

Telugu Character strokes recognition by SVM yielded better results where each character is divided into subclasses, but the model only classifies strokes. Many researchers trained their model directly on the word image which produces better results for natural scene text recognitions but not for sentences. Two blob Extracting method used to extract text from an comic image using Median filter in pre-processing to remove noise yielded 94.82% accuracy. Convolutional with Recurrent neural networks on Urdu text using ResNet with Batch Normalization gave 52.72% of accuracy on custom dataset, and CRNN on Urdu text using VGG16 method on new dataset gave 87.13%. A hybrid model using Alexnet and VGG on Urdu Ligature recognition gave 80.46. Several types of algorithms need to be tested for our model dataset and analyze the best algorithm.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Many of the researches have been done on natural scene images and doesn’t work when given an image with a sentence. Many research models trained model on word image directly rather than training a single character which may occur low accuracy for complex words which may increase error rate. There is less concentration on Telugu language, and it is different from other cursive languages because of types of strokes present in Telugu.

One of the primary challenges in Telugu character recognition is the limited dataset.

Most studies in this field have used a small and relatively simple dataset, which affects the accuracy of the models. The small dataset size may lead to overfitting, resulting in models that perform well on the training data but poorly on unseen data. To improve the accuracy of the existing systems, larger and more diverse datasets are needed. However, collecting and annotating such a dataset is a significant challenge, given the large number of Telugu characters and the variability in handwriting styles.

Another issue in Telugu character recognition is the lack of standardization. Currently, there is no standardized dataset or evaluation criteria for Telugu character recognition, making it challenging to compare the performance of different models. The absence of a standard dataset and evaluation metrics hinders progress in this field, as researchers cannot directly compare the results of their models with those of other researchers. Developing a standardized dataset and evaluation criteria can help address this issue.

Most studies on Telugu character recognition have focused on recognizing individual characters. However, there is a need for more research on recognizing full words or sentences in Telugu documents. Recognizing complete words or sentences is more challenging than recognizing individual characters, as it requires the model to take into account the context of the characters. Developing models that can recognize full words or sentences can enable applications such as automatic transcription of handwritten documents.

So, in this paper we propose a model that takes an image with Telugu text as input and then segments the sentences lines to words and characters and recognize the pattern and outputs the Telugu text.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

The goal of this study is to create a model that takes an image as an input, preprocesses it, extracts its features, and then trains the model using the dataset. Here, we are creating a new dataset of each manually cropped image of a Telugu character. The extraction of characteristics from photographs of Telugu characters is a difficult task since the characters in the Telugu language are more spherical in shape and consist of 253 unique strokes. The majority of strokes are similar, which makes the model more difficult. In terms of technical feasibility, the Telugu text recognition project is feasible as there are already existing techniques and tools for image processing, OCR, and deep learning that can be utilized. However, the team will need to assess the accuracy and reliability of these existing tools to ensure they are suitable for recognizing Telugu text specifically.

Risk analysis is an important aspect of project management and involves identifying potential risks that may affect the project's success and developing strategies to mitigate those risks. Some potential risks for the Telugu text recognition project include:

1. *Data quality*: Poor quality images may affect the accuracy of text recognition. The team will need to assess the quality of the data and develop strategies to address any issues with data quality.
2. *Technical challenges*: The project may encounter technical challenges such as software and hardware compatibility issues, performance issues, and accuracy issues with recognition algorithms. The team will need to have a contingency plan to address these challenges.
3. *Time constraints*: The project timeline may be impacted by unexpected delays or changes in requirements. The team will need to develop a flexible project plan that can accommodate changes in scope or timelines.

To mitigate these risks, the project team can implement strategies such as developing a robust quality assurance plan, conducting thorough testing and validation, and ensuring effective communication and collaboration among team members.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Software Requirements:

The Telugu text recognition project requires several software tools to implement the different modules of the system. The main programming language used for the project is Python, which is a popular language for machine learning and computer vision tasks. Python has a large and active community, making it easy to find libraries and resources for developing the project.

One essential tool for the project is the Anaconda distribution, which is a package manager for Python that simplifies the installation of libraries and dependencies. Anaconda provides a user-friendly interface for managing environments and installing packages, making it easier to create a reproducible and consistent development environment. Anaconda also includes Jupyter Notebook, an interactive web-based environment for data analysis and visualization, which can

be used for exploring the data and prototyping the different modules of the system.

Overall, the software requirements for the Telugu text recognition project include Python, Anaconda, Jupyter Notebook, OpenCV, and other Python libraries for specific tasks. The use of a GPU for faster computations is also recommended. With these tools and libraries, the project team can develop and implement the different modules of the system for accurate and efficient recognition of Telugu text.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

In this proposed model, we create a model that accepts a picture containing Telugu text as input, recognizes the text in the image, and outputs the text. A major issue in the field of Telugu OCR is a lack of large data repositories of Telugu characters that are needed for training. We introduce a new dataset of manually cropped Telugu characters, and train the model to first recognize the characters. Optical character recognition (OCR) has been one of the most studied problems in pattern recognition. In this model we first train our model to recognize each Telugu character and then for testing the input image is segmented from lines to words and then characters, then the model recognize the character and then outputs the Telugu text.

4.1 SELECTED METHODOLOGY FOR PROCESS MODEL

Telugu text recognition is the process of automatically recognizing and extracting text from Telugu language images. The methodology of Telugu text recognition involves several stages, including data collection and pre-processing, line segmentation, word segmentation, character recognition, text reconstruction,

evaluation, and implementation.

The first stage in the methodology of Telugu text recognition is data collection and pre-processing. In this stage, a diverse set of images containing Telugu text is collected. The images are converted to grayscale format, which simplifies the processing steps and reduces the computational complexity. Noise reduction techniques such as Gaussian blur or median filtering are applied to remove unwanted noise from the images. A thresholding technique such as OTSU is used to convert the images to binary format, which makes it easier to detect lines and characters in the image.

The next stage in the methodology of Telugu text recognition is line segmentation. In this stage, the non-zero pixels in each row of the binary image are identified. The non-zero pixels are used to segment the image into lines of text. Morphology operations such as erosion and dilation are applied to improve the accuracy of line segmentation. The output of this module is a set of binary images, each containing a single line of text.

The third stage in the methodology of Telugu text recognition is word segmentation. In this stage, techniques such as character spacing or connected component analysis are applied to segment the lines into individual words. The non-zero pixels in each column of the binary image are identified to detect spaces between words. Adjacent characters are combined to form words. The output of this module is a set of binary images, each containing a single word.

The fourth stage in the methodology of Telugu text recognition is character recognition. In this stage, a character recognition model is trained using techniques such as Optical Character Recognition (OCR) or deep learning-based approaches. The trained model is used to identify and classify individual characters in the segmented words. Post-processing techniques such as spell checking and character correction are implemented to improve the accuracy of character recognition. The output of this module is a set of recognized characters.

The fifth stage in the methodology of Telugu text recognition is text reconstruction.

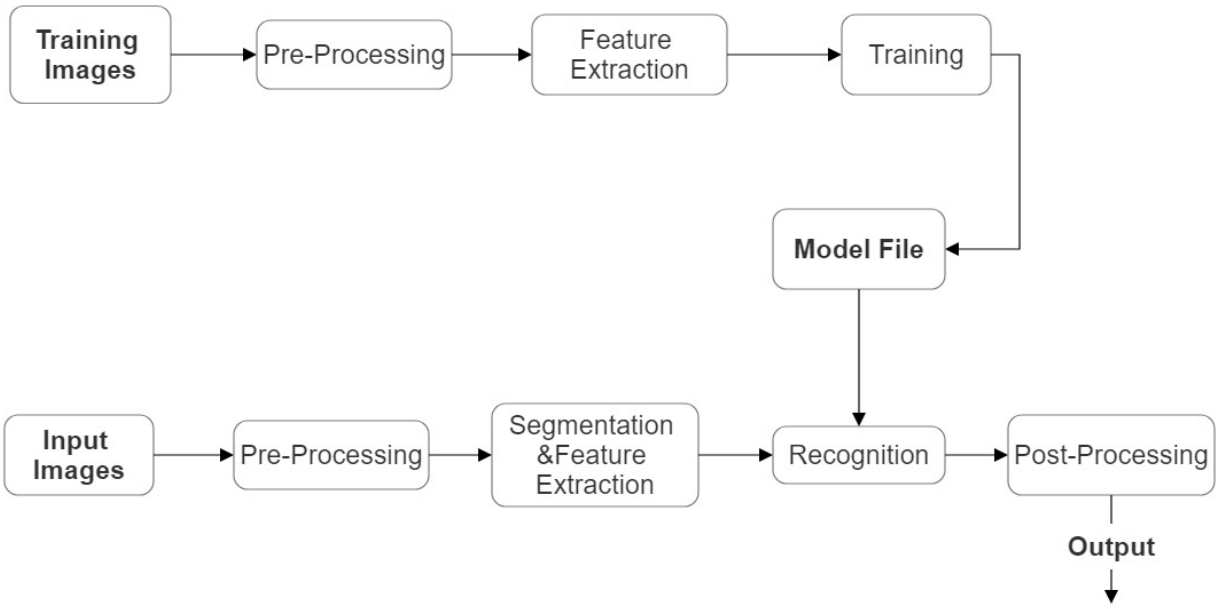
In this stage, the recognized characters are combined into words, and the words are combined into lines of text. Techniques such as spell checking and grammar correction are implemented to improve the readability of the reconstructed text. The output of this module is the final recognized text.

The sixth stage in the methodology of Telugu text recognition is evaluation. In this stage, the accuracy and performance of the Telugu text recognition system are evaluated using metrics such as precision, recall, and F1 score. The performance of the system is compared against existing Telugu text recognition methods.

The final stage in the methodology of Telugu text recognition is implementation. In this stage, the Telugu text recognition system is implemented using programming languages such as Python or MATLAB. The system is integrated into a user-friendly interface, allowing users to upload images and view the recognized text output.

In conclusion, the methodology of Telugu text recognition involves several stages, including data collection and pre-processing, line segmentation, word segmentation, character recognition, text reconstruction, evaluation, and implementation. Each stage plays a crucial role in the Telugu text recognition system, and careful attention should be paid to the design and implementation of each stage to ensure accurate and efficient recognition of Telugu text.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM



4.1 OCR MODEL ARCHITECTURE

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL

Software implementation is an essential aspect of the proposed Telugu text recognition model. The implementation of the model involves the development of software using programming languages such as Python or MATLAB. The software should be designed to perform each stage of the Telugu text recognition methodology, including data collection and pre-processing, line segmentation, word segmentation, character recognition, text reconstruction, evaluation, and implementation.

The first step in the software implementation process is to develop the user interface. The user interface should be simple, intuitive, and easy to use. The interface should allow users to upload images containing Telugu text and view the recognized text output. The interface should also provide options for users to adjust the thresholding, line segmentation, and word segmentation parameters.

The second step in the software implementation process is to develop the data pre-processing module. The data pre-processing module should include techniques such as grayscale conversion, noise reduction, and thresholding. The module should also include options for users to adjust the thresholding parameters.

The third step in the software implementation process is to develop the line segmentation module. The line segmentation module should be designed to identify the non-zero pixels in each row of the binary image and segment the image into lines of text. The module should include morphology operations such as erosion and dilation to improve the accuracy of line segmentation.

The fourth step in the software implementation process is to develop the word segmentation module. The word segmentation module should include techniques such as character spacing or connected component analysis to segment the lines into individual words. The module should also include options for users to adjust the word segmentation parameters.

The fifth step in the software implementation process is to develop the character recognition module. The character recognition module should be trained using techniques such as OCR or deep learning-based approaches. The module should include post-processing techniques such as spell checking and character correction to improve the accuracy of character recognition.

The sixth step in the software implementation process is to develop the text reconstruction module. The text reconstruction module should be designed to combine the recognized characters into words and the words into lines of text. The module should include options for users to adjust the spell checking and grammar correction parameters.

The seventh step in the software implementation process is to develop the evaluation module. The evaluation module should include metrics such as precision, recall, and F1 score to evaluate the accuracy and performance of the Telugu text recognition system. The module should also provide options for users to adjust the evaluation parameters.

The final step in the software implementation process is to test the system. The testing plan should include unit testing, integration testing, and system testing. Unit testing involves testing each module individually to ensure that it performs as

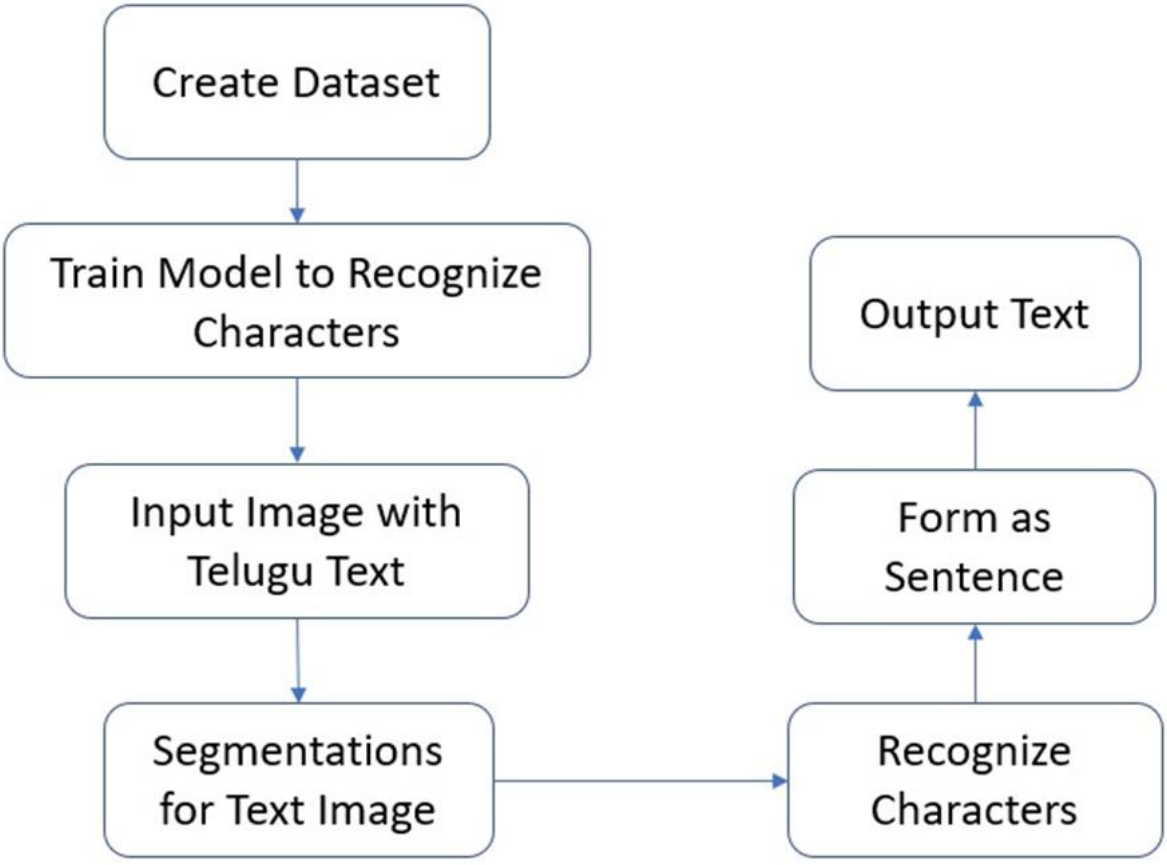
expected. Integration testing involves testing the integration of the modules to ensure that they work together seamlessly. System testing involves testing the entire system to ensure that it meets the desired specifications and requirements.

In conclusion, the software implementation and testing plan for the proposed Telugu text recognition model is a critical aspect of the model's success. The software should be designed to perform each stage of the Telugu text recognition methodology, and the testing plan should include unit testing, integration testing, and system testing. The software should also be user-friendly, intuitive, and easy to use, allowing users to upload images and view the recognized text output. By following a comprehensive implementation and testing plan, the Telugu text recognition model can be successfully implemented and tested, ensuring accurate and efficient recognition of Telugu text.

4.4 PROJECT MANAGEMENT PLAN

Weeks 1-2	● Project initiation and planning ● Requirements gathering and analysis ● Team formation and task assignment ● collecting the data for every telugu character.
Weeks 2-4	● Develop and implement data pre-processing module ● Develop and implement line segmentation module
Weeks 4-8	● Develop and implement word segmentation module ● Develop and implement character recognition module
Weeks 8-12	● Develop and implement text reconstruction module ● Integration of all modules into a single

	software system
Weeks 12-15	• Testing the model • Finalize software system based on testing and feedback
Weeks 15-18	• Finalizing Documentation • Research paper preparation • Presenting a paper at the conference • Publishing the research paper



4.2 PROJECT MANAGEMENT PLAN

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Data collection and preprocessing

The primary goal of the data collection phase is to accumulate a comprehensive set of Telugu symbols. In order to train the character recognition system, we will use this dataset. Telugu has 16 vowels letters and 36 consonant letters. The "vathugunithalu" & the "main letter" make up the two halves of a Telugu syllable. When coupled with one of the other 36 consonant letters, the "vathu" takes on a unique pronunciation for each character. There are 541 different permutations of "vathus" in Telugu that seem the same, which increases the complexity and quantity of the dataset. Using each vathu as a class, there are 72 samples in the dataset, with 100-150 samples collected for each vowel and consonant. Books, newspapers, & the Internet were just some of the places we scoured for these pictures. The reliability of the recognition system is significantly impacted by the amount of the data.

Once the data is collected, firstly normalized the images and data augmented techniques used to increase accuracy of model the data then the images are converted to grayscale format. The reason behind converting images to grayscale format is to simplify the processing steps and reduce computational complexity. Grayscale images contain only one channel, whereas coloured images have 3 channels (red, green, and blue). By using grayscale images, we can eliminate the need to process three channels of color, which speeds up the processing time and reduces the computational burden.

After converting the images to grayscale format, noise reduction techniques such as Gaussian blur or median filtering are applied. These techniques are used to remove unwanted noise from the images. Noise can negatively affect the accuracy of character recognition, so it is important to remove it before further processing. Next, a thresholding technique such as OTSU is used to convert the images to binary format. This method is used to transform colour photos into monochrome ones. By establishing a threshold value, the foreground & background of a picture may be extracted using the thresholding method. Pixels with values higher than the threshold value are made white, while those with lower values are made black.

Binary images are easier to process, and it simplifies the task of detecting lines and characters in the image.

Skew correction is also an important step in the preprocessing module. Skew correction is a technique used to correct the slant or tilt of an image. The purpose of this step is to ensure that the characters in the images are properly aligned and can be accurately recognized by the character recognition model. In summary, the Data Collection and Preprocessing module plays a crucial role in the Telugu text recognition system. The module collects a large dataset of Telugu characters, processes the images to make them suitable for further analysis, and performs skew correction to ensure accurate recognition of characters

5.2 Algorithm

5.2.1 Line segmentation

Line segmentation is an important step in Telugu text recognition, as it allows the individual lines of text to be separated from one another. This makes it easier to recognize and extract the text from the input image.

After the input image has been converted into a binary image using a thresholding technique, line segmentation can be performed by analyzing the non-zero pixels in each row of the image. Specifically, line segmentation involves identifying the rows in the binary image that contain non-zero pixels, which correspond to the rows of text in the input image.

To perform line segmentation, we can start by iterating through each row of the binary image. For each row, we can count the number of non-zero pixels using a function such as numpy.count_nonzero(). If the number of non-zero pixels in a row is greater than a certain threshold value, we can assume that the row contains text and add it to a list of candidate rows.

Once we have identified all the candidate rows, we can use various techniques to group them together into individual lines of text. One common technique is to use clustering algorithms such as k-means or hierarchical clustering to group the candidate rows based on their vertical position in the image. Another technique is to use line-fitting algorithms to identify the lines of text based on their slope and intercept.

Regardless of the technique used, the goal of line segmentation is to identify the individual lines of text in the input image so that they can be processed and recognized separately. This allows for greater accuracy in character recognition and text extraction, as it reduces the possibility of characters from different lines being combined or misidentified.

5.2.2 WORD SEGMENTATION

Word segmentation is the process of dividing a line of text into individual words. It is a crucial step in the process of recognizing text from an image, as it allows the recognized characters to be grouped together to form words. In Telugu text recognition, word segmentation is typically performed after the input image has been converted into a binary image and line segmented.

To perform word segmentation, the binary image of the line of text is scanned from left to right. At each column, the number of non-zero pixels is counted. When the number of non-zero pixels exceeds a threshold, a new word is assumed to have started. The threshold can be set based on the average width of characters in the line of text.

Once the start and end points of each word have been identified, the corresponding pixels can be extracted from the binary image to form a separate image of each word. The individual word images can then be processed using techniques such as OCR or deep learning based approaches to recognize the individual characters.

There are various techniques that can be used to improve the accuracy of word segmentation, such as using a dictionary of known words to guide the segmentation process, or using machine learning algorithms to learn the characteristics of Telugu text and improve the accuracy of the segmentation.

Overall, word segmentation is an important step in the process of recognizing Telugu text from an image, as it allows the recognized characters to be grouped together to form words. With the use of advanced algorithms and techniques, it is possible to achieve high levels of accuracy in Telugu text recognition and word

segmentation.

5.2.3 CHARACTER SEGMENTATION

Character segmentation is the process of identifying and separating individual characters within a word or line of text. This process is typically performed after the image has been converted into a binary image and the words have been segmented.

In Telugu text recognition, character segmentation can be achieved using various techniques, such as connected component analysis or OCR-based approaches. Connected component analysis involves identifying connected regions of non-zero pixels within the binary image and separating them into individual characters. OCR-based approaches involve using machine learning algorithms to recognize and classify individual characters within the segmented words.

Once the characters have been segmented, they can be passed through a character recognition model to identify the specific character. In Telugu text, there are 50 unique classes of main characters and over 500 unique classes of vattus and gunithams. Therefore, the character recognition model must be trained on a large dataset of images to accurately identify each character.

Character segmentation is a crucial step in the Telugu text recognition process, as it allows individual characters to be accurately recognized and classified. The accuracy of the character segmentation process depends on the effectiveness of the segmentation techniques used, as well as the quality of the input image. With the use of advanced algorithms and techniques, it is possible to achieve high levels of accuracy in Telugu character segmentation and recognition.

5.3 TRAINING

Training two separate models for recognizing Telugu characters and vattus/gunithams is an effective approach to achieving high accuracy in character recognition. The architecture for each model, as you have described, consists of three convolutional layers, followed by max pooling and a dense layer.

Convolutional layers are used in neural networks to learn patterns in the input data. By applying a set of filters to the input image, the network can detect edges, shapes, and other features that are relevant to the recognition task. Max pooling is then used to downsample the feature maps, reducing the dimensionality of the input data and improving the speed and efficiency of the model.

The dense layer is used to perform classification based on the learned features. In the case of the Telugu character recognition model, the dense layer is trained to classify each input image into one of 50 unique classes, representing each Telugu character. In the vattu/gunitham recognition model, the dense layer is trained to classify each input image into one of 541 unique classes, representing each vattu or gunitham.

To train each model, a large dataset of images was collected, consisting of 50-100 images for each Telugu character and around 61,000 images for vattus/gunithams. Each image was preprocessed into a numpy array of size 64x64, which is a common size used in many image recognition tasks. The images were then fed into the convolutional layers of the model for feature extraction and classification.

By training each model on a large dataset of images, the models can learn to recognize a wide range of Telugu characters and vattus/gunithams with high accuracy. The use of convolutional layers and max pooling allows the models to learn complex patterns and features in the input data, improving their performance on the recognition task.

5.3.1 First model

The architecture for the first model used in Telugu character recognition consists of three convolutional layers, followed by max pooling and a dense layer.

The first convolutional layer has 20 filters, which are applied to the input image to detect patterns and features. The size of each filter is typically small, such as 3x3 or 5x5, allowing the model to capture local patterns in the input image. After the

filters are applied, the resulting feature maps are downsampled using max pooling with a window size of 2x2. Max pooling helps to reduce the size of the feature maps and prevent overfitting, while still preserving the most relevant information.

The second convolutional layer has 50 filters, which are applied to the downsampled feature maps from the previous layer. This layer is designed to detect higher-level patterns and features that may be spread across larger areas of the input image. After the filters are applied, the resulting feature maps are again downsampled using max pooling with a window size of 2x2.

The final convolutional layer also has 50 filters and is applied to the downsampled feature maps from the previous layer. This layer further refines the learned features, and the resulting feature maps are again downsampled using max pooling with a window size of 2x2.

The output of the last max pooling layer is then flattened into a 1D array and fed into a dense layer. The dense layer has 128 neurons and uses the Rectified Linear Unit (ReLU) activation function to compute the weighted sum of the inputs. The output of the dense layer is then fed into the output layer, which has 50 neurons, one for each unique Telugu character. The output layer uses the Softmax activation function to convert the weighted sum into a probability distribution over the 50 character classes.

The architecture for the first model is designed to extract relevant features from the input image, learn complex patterns and relationships between features, and classify the input image into one of 50 Telugu character classes with high accuracy. The use of convolutional layers with max pooling allows the model to learn features at different levels of abstraction, from local patterns to global relationships, while also reducing the dimensionality of the input data. The dense layer and output layer then perform the final classification based on the learned features.

5.3.2 Second model

The architecture for the second model in Telugu text recognition, which is designed to recognize vattus and gunithams, consists of three convolutional layers

followed by max pooling, and a dense layer.

The first convolutional layer applies 20 filters to the input image to extract low-level features such as edges and corners. Each filter learns to recognize a different feature or pattern in the input image. The output of this layer is then passed through a rectified linear unit (ReLU) activation function, which applies a non-linear transformation to the output, making it easier for the model to learn complex patterns.

The second convolutional layer applies 50 filters to the output of the first layer, to extract higher-level features that are more complex than those detected by the first layer. The output of this layer is also passed through a ReLU activation function before being passed to the next layer.

The third convolutional layer applies another 50 filters to the output of the second layer, further extracting high-level features. The output of this layer is then passed through another ReLU activation function.

After the final convolutional layer, the output is passed through a max pooling layer, which reduces the spatial dimensionality of the feature maps and extracts the most important features. The max pooling operation is performed using a 2x2 window, which takes the maximum value of each group of four adjacent pixels.

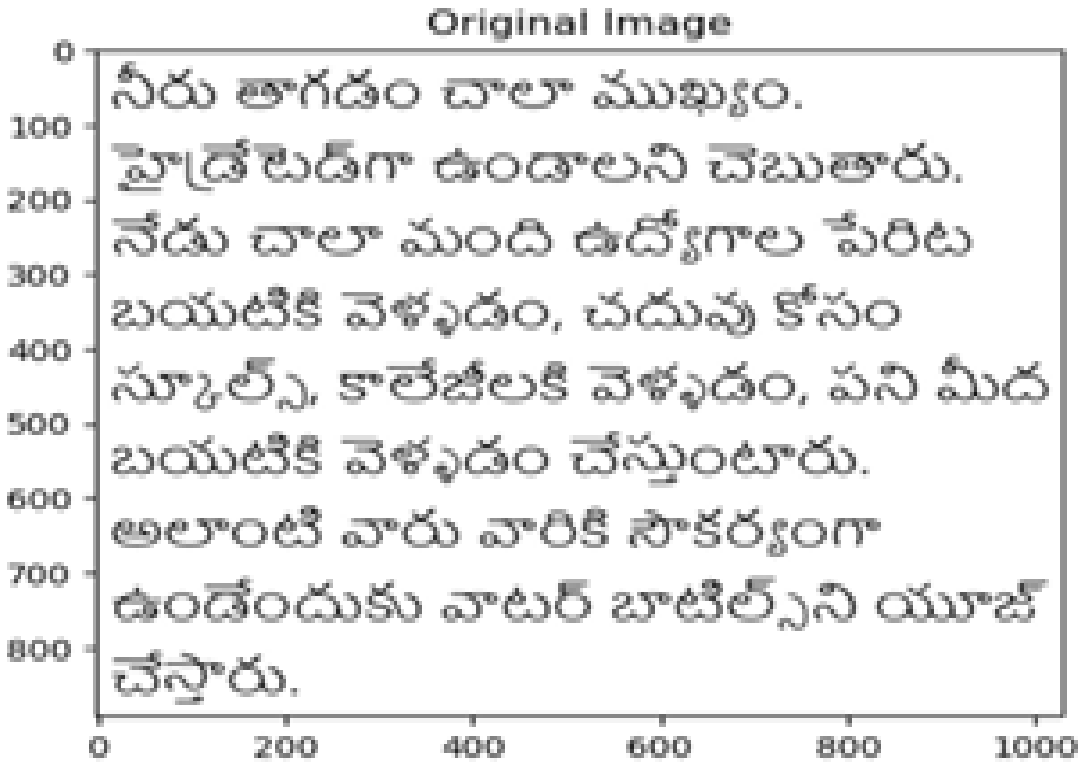
Finally, the output is flattened into a vector and passed through a dense layer. The dense layer is a fully connected layer that performs classification based on the features learned by the convolutional layers. In this case, the dense layer is trained to classify the input image into one of 541 unique classes, each representing a different vattu or gunitham.

The use of multiple convolutional layers followed by max pooling allows the model to learn complex patterns and features in the input data, improving its performance on the recognition task. The ReLU activation function helps to improve the model's ability to learn non-linear relationships between the input and output data. The final dense layer performs classification based on the learned

features, allowing the model to accurately recognize the vattus and gunithams in Telugu text.

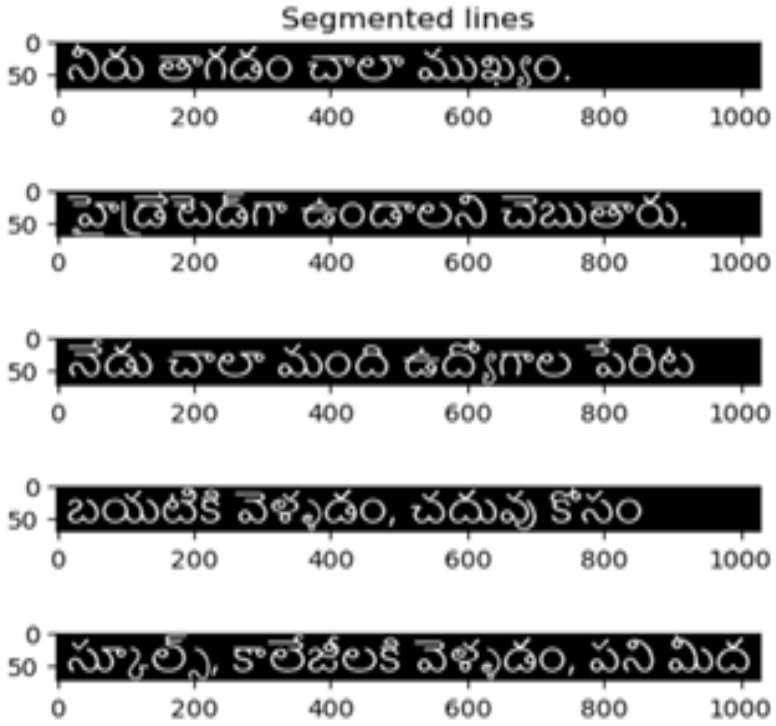
CHAPTER 6
RESULTS AND DISCUSSIONS

The first step is to segment the image into lines, then words, & lastly letters. The image is pre-processed by converting it to grayscale and then to a binary image using OTSU thresholding. The image is segmented into lines by checking the number of non-zero pixels for each row of the NumPy array, which contains only 0's and 1's from the binarization process.



6.1 INPUT IMAGE

After the input image has been converted into a binary image using a thresholding technique, line segmentation can be performed by analyzing the non-zero pixels in each row of the image. A row with at least one non-zero and its preceding row with all zeros is considered the start point, and a row with at least one non-zero and its succeeding row with all zeros is considered the end point. The NumPy array is sliced using these start and end indices to split the image into lines.

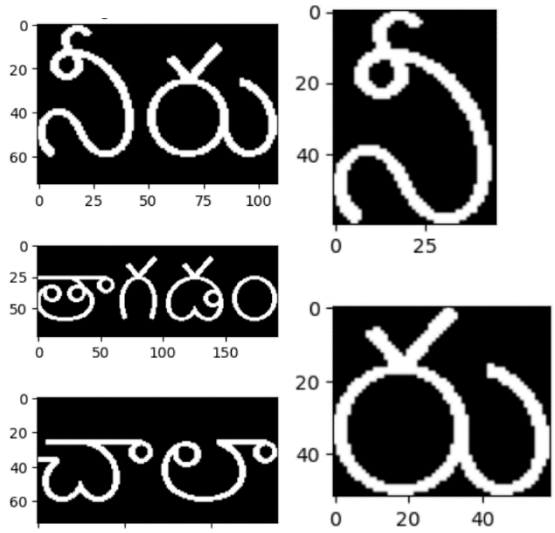


6.2 LINES SEGMENTATION

After line segmentation then word segmentation is done ,To perform word segmentation, the binary image of the line of text is scanned from left to right. At each column, the amount of non-zero pixels is counted. When the amount of non-zero pixels exceeds a threshold, a new word is assumed to have started. The threshold can be set based on the average width of characters in the line of text. Once the start and end points of each word have been identified, the corresponding pixels can be extracted from the binary image to form a separate image of each word

Character segmentation is the process of identifying and separating individual characters within a word or line of text. This process is typically performed after the image has been converted into a binary image and the words have been segmented. Character segmentation can be achieved using Connected component analysis involves identifying connected regions of non-zero pixels within the binary image and separating them into individual characters.

After word segmentation is done then character recognition takes place and text reconstruction. To reconstruct text, we use two models that are designed to identify each character and combine them with appropriate spaces to form words.



6.3 WORD SEGMENTATION 6.4 LETTER SEGMENTATION

Finally, the output text Telugu text is translated into the English language. Converting Telugu text to English text is a useful phase of the procedure of Telugu information extraction, as it allows the recognized text to be easily understood and analyzed by English-speaking users. This can be achieved using various techniques and libraries, such as the mtranslate library.

The mtranslate library is a Python library that allows for the translation of text from one language to another. To convert Telugu text to English text using the mtranslate library, the recognized Telugu text is passed as an input to the library's translation function, along with the source and target languages. The function then returns the translated text in the target language, in this case, English.

CHAPTER 7

CONCLUSION

Text recognition in images is a challenging problem in computer vision, many

researches have been done on scene text recognition, but many researches were performed on English text and a less concentration on other language. So, here we focused on Telugu text, Telugu language is a complex cursive script which have curved characters with a top and bottom stroke depending on the letter. Here we proposed a model with Optical character recognition (OCR) implemented using deep neural networks. The proposed model takes an image as an input and then transforms into the sequence of the relevant features and then extracts the text in the image and gives English translated text as output. We introduced a new dataset with manually cropped images to train the model. The text recognition of Telugu language is very useful for people who don't know language and anyone who are new to Telugu states they will telugu texts all over then they can snap a pic of text and can get translated English text, and it has wide ranging applications including education, administration etc.

Recognizing text in images is a complex and challenging task in computer vision. While there have been numerous studies conducted on scene text recognition, the majority of research has been focused on English text, with less attention given to other languages. In order to address this gap, our research has specifically focused on Telugu text recognition.

Telugu is a complex cursive script that consists of curved characters with a top and bottom stroke that varies depending on the letter. To overcome the challenges associated with this script, we have proposed a model that utilizes Optical Character Recognition (OCR) technology, which is implemented using deep neural networks.

Our proposed model takes an image as input and transforms it into a sequence of relevant features, allowing it to extract the text contained within the image. The output generated by the model is English translated text. To train the model, we have created a new dataset that consists of manually cropped images.

The ability to recognize Telugu text is particularly useful for individuals who are not familiar with the language. For instance, individuals who are new to Telugu states may encounter Telugu texts all over, and by simply taking a picture of the

text, they can receive translated English text. The application of this technology is widespread and has implications in various fields, including education and administration.

REFERENCES

[1] Asghar Ali, Mark Pickering, “A Hybrid Deep Neural Network for Urdu Text Recognition in Natural Images”, 2019.

[2] Muni Sekhar Velpuru, Priyadarshini Chatterjee, G Tejasree, M Ravi Kumar, S Nageswara Rao, “Comprehensive study of Deep learning based Telugu OCR”, 2020.

[3] ASGHAR ALI CHANDIO, MD. ASIKUZZAMAN, MARK R. PICKERING, (Member, IEEE), AND MEHWISH LEGHARI, “Cursive Text Recognition in Natural Scene Images Using Deep Convolutional Recurrent Neural Network”, 2022.

[4] Yuming He, “Research on Text Detection and Recognition Based on OCR Recognition Technology”, 2020.

[5] SYED YASSER ARAFAT AND MUHAMMAD JAVED IQBAL, “Two Stream Deep Neural Network for Sequence-Based Urdu Ligature Recognition”, 2019.

[6] Han Lin, Peng Yang, Fanlong Zhang, “Review of Scene Text Detection and Recognition”, 2019.

[7] SAAD BIN AHMED, SAEEDA NAZ, MUHAMMAD IMRAN RAZZAK, AND RUBIYAH BTE YUSOF, “A Novel Dataset for English-Arabic Scene Text Recognition (EASTR)-42K and Its Evaluation Using Invariant Feature Extraction on Detected Extremal Regions”, 2019.

[8] Xin Tang, Yongquan Lai, Ying Liu, Yuanyuan Fu, Rui Fang, “Visual-Semantic Transformer for Scene Text Recognition”, 2021.

[9] Yue Tao, Zhiwei Jia, Runze Ma, and Shugong Xu, “TRIG: Transformer-Based Text Recognizer with Initial Embedding Guidance”, 2021.

[10] Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, Yu-Gang Jiang, “SVTR: Scene Text Recognition with a Single Visual Model", 2020.

[11] Sanghamitra Mohanty ,Himadri Nandini Dasbebartta,Tarun Kumar Behera, “An Efficient Bilingual Optical Character Recognition (English-Oriya) System for Printed Documents”, 2009

[12] Dr.M.Sundaresan, S.Ranjini, "Text Extraction from Digital English Comic Image Using Two Blobs Extraction Method", 2012

[13] M Abdul Rahiman, M S Rajasree, "Printed Malayalam Character Recognition Using Back-propagation Neural", 2009

[14] Rajkumar.J, Mariraja K., Kanakapriya,K., Nishanthini, S., Chakravarthy, V.S, "Two Schemas for Online Character Recognition of Telugu script based on Support Vector Machines", 2012

Anitha Jayaraman, C. Chandra Sekhar, "Modular Approach to Recognition of Strokes in Telugu Script", 2007

SOURCE CODE

```
# !pip install opencv-python
# !pip install Pillow
# from PIL import Image
# !pip install deskew
# !pip install tensorflow
# !pip install googletrans==4.0.0-rc1
# !pip install mtranslate


import numpy as np
import cv2
```

```
import matplotlib.pyplot as plt

path='./test/test.jpg'
img = cv2.imread(path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

print(img.shape)
plt.rcParams['figure.figsize'] = (5,5)
plt.imshow(img)
plt.title('Original Image')
plt.show()

img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print(img.shape)
plt.imshow(img,cmap='Greys')
plt.title('Grayscale')
plt.show()

print(img.shape)
```

```
print(img)

# Converting to Binary Image
def binary_img(img):

    ret,bin_img =
cv2.threshold(img,0,255,cv2.THRESH_OTSU+cv2.THRESH_BINARY)
    return bin_img

bin_img = binary_img(img)
plt.imshow(bin_img,cmap='Greys')
plt.title('Preprocessed Image')
plt.show()
```

```
bin_img

# segmenting img to lines

def seg_lines(img):
    lines=[]
    seg_points = []
    start = end = 0
    for i in range(1,len(img)-1):
        if np.count_nonzero(img[i] == 0) > 0 and np.count_nonzero(img[i-1] == 0)
== 0 :
            start = i
        if np.count_nonzero(img[i] == 0) > 0 and np.count_nonzero(img[i+1] == 0)
== 0 :
            end = i
            seg_points.append([start-1,end+1])
```

```
    for i in range(len(seg_points)):
        temp_img = img[seg_points[i][0]:seg_points[i][1]]
        if(len(temp_img)>=20):
            lines.append(temp_img)
    return lines

lines = seg_lines(bin_img)
plt.title('Segmented lines')
for i in lines:
    plt.rcParams['figure.figsize'] = (5,5)
    plt.imshow(i,cmap='Greys')
    plt.show()
plt.imshow(bin_img,cmap='Greys')

# segmenting lines to words
```

```
def remove_blanks(img,n):                    #removing blank space before and
after
    start,end = 0, n-1
    for i in range(n):
        if(np.count_nonzero(img[i]==0) > 0):
            start = i
            break
    for i in range(n-1,0,-1):
        if(np.count_nonzero(img[i]==0) > 0):
            end = i
            break
    return img[start:end+1]

def seg_words(img):
```

```
    def min_gap(img,n):                      #finding minimun gap between words to
split
        temp = []
        i = 0
        while(i<n):
            if(np.count_nonzero(img[i]==0) == 0):
                d = gap(img,i,n)
                temp.append(d)
                i = i+d+1
            else:
                i += 1

        temp.sort()
        diff = 0
        min_gap = 0
        for i in range(1,len(temp)):
```

```
        d = temp[i]-temp[i-1]
        if(diff < d):
            diff = d
            min_gap = temp[i-1]

    return min_gap

def gap(img,index,n):
    if(index == n-1):
        return 1
    i = index + 1
    while(i < n):
        if(np.count_nonzero(trns_img[i] == 0) != 0 ):
            break
```

```
        i += 1
    return i-index



trns_img = np.transpose(img)
n = len(trns_img)

trns_img = remove_blanks(trns_img,n)
n = len(trns_img)
min_gap = min_gap(trns_img,n)

i = 0
temp_arr = []
while(i < n):
    start = 0
    if(np.count_nonzero(trns_img[i] == 0) !=0 ):
        start = i
        end = n
```

```
        while(i < n):
            if(np.count_nonzero(trns_img[i] == 0) == 0 ):
                d = gap(trns_img,i,n)
                if( d > min_gap):
                    end = i
                    i += d
                    break
                else:
                    i += d
            else:
                i += 1
        temp = np.transpose([trns_img[start:end]])
```

```
      temp = temp.reshape(len(temp),-1)
          temp_arr.append(temp)
       i += 1

   return temp_arr

words = seg_words(lines[0])
plt.title('Segmented Words')
for i in words:
   plt.rcParams['figure.figsize'] = (3,3)
   plt.imshow(i,cmap='Greys')
#    print(i.shape)
   plt.show()

# segmenting into letters

def seg_letters(img):

   word = remove_blanks(img,len(img))
   word = np.transpose(word)
```

```
    i = 0
    n = len(word)
    temp_arr = []
    while(i<n):
        if(np.count_nonzero(word[i] == 0) > 0 ):
            start = i
            end = n
            while(i<n):
                if(np.count_nonzero(word[i]==0) == 0 ):
```

```
        end = i
                    break
                i += 1
            temp = np.transpose(word[start:end+1])
            temp = remove_blanks(temp,len(temp))
            temp_arr.append(temp)
        else:
            i += 1

    return temp_arr


letters = seg_letters(words[3])
print(len(letters))
plt.title('Segmented Letters')
print()
plt.rcParams['figure.figsize'] = (1,1)
for i in letters:
    plt.imshow(i,cmap='Greys')
    plt.rcParams['figure.figsize'] = (2,2)
#     print(i.shape)
    plt.show()
```

```
def resize(img,n):                                  #resize to 'n' pixels
    h,w = img.shape
    hi=wid=0
    if(h>w):
        ratio = w/h
        wid = int(ratio*n)
        diff = n-wid
```

```
padd = np.ones([n,diff],dtype = int)*255
        crop = cv2.resize(img, (wid,n))

        temp = np.append(padd[:,:diff//2],crop,axis = 1)
        temp = np.append(temp,padd[:,diff//2:],axis = 1)

    else:
        ratio = h/w
        hi = int(ratio*n)
        diff = n-hi

        padd = np.ones([diff,n],dtype = int)*255
        crop = cv2.resize(img, (n,hi))

        temp = np.append(padd[:diff//2],crop,axis = 0)
        temp = np.append(temp,padd[diff//2:],axis = 0)
    return temp

for i in letters:
    a = resize(i,64)
    print(a.shape)
    plt.imshow(a,cmap='Greys')
    plt.show()
```

```
import os
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
```

```
from keras.utils import to_categorical
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout



path = './characters/'
data = []
labels = []

#data aug

datagen_1 = ImageDataGenerator(
    horizontal_flip=True,
    vertical_flip=False
)



datagen_2 = ImageDataGenerator(
    horizontal_flip=False,
    vertical_flip=True
)

datagen_3 = ImageDataGenerator(
    horizontal_flip=True,
    vertical_flip=True
)
```

```
datagen_4 = ImageDataGenerator(
    rotation_range=45,
    horizontal_flip=True,
    vertical_flip=False,
```

```
    fill_mode='constant', cval=255
)

datagen_5 = ImageDataGenerator(
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=False,
    fill_mode='constant', cval=255
)

datagen_6 = ImageDataGenerator(
    rotation_range=45,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='constant', cval=255
)

datagen_7 = ImageDataGenerator(
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='constant', cval=255
)



for foldername in os.listdir(path):
    temp_path = path + foldername + '/'
    for i in os.listdir(temp_path):
```

```
        temp_path_1 = temp_path + i
        img = cv2.imread(temp_path_1)
```

```
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        img = np.transpose(img)
        img = remove_blanks(img,len(img))
        img = np.transpose(img)
        img = remove_blanks(img,len(img))

        img = resize(img, 64)
        img = img.reshape(64, 64, 1)

        data.append(img)
        labels.append(int(foldername.split('_')[0])-1)

        img_aug = img.reshape((-1, 64, 64, 1))

        img_batch = datagen_1.flow(img_aug, batch_size=1).next()
        img_batch = datagen_2.flow(img_aug, batch_size=1).next()
        img_batch = datagen_3.flow(img_aug, batch_size=1).next()

        for _ in range(4):
            img_batch = datagen_4.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

        for _ in range(4):
            img_batch = datagen_5.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

        for _ in range(4):
```

```
            img_batch = datagen_6.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

        for _ in range(4):
            img_batch = datagen_7.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

data = np.array(data)
labels = np.array(labels)

train_data, val_data, train_labels, val_labels = train_test_split(data, labels,
test_size=0.2, random_state=42)

train_data = train_data / 255.0
val_data = val_data / 255.0



train_data = train_data.reshape(-1, 64, 64, 1)
val_data = val_data.reshape(-1, 64, 64, 1)



model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(100, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(53, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model.fit(train_data, to_categorical(train_labels), epochs=20,
validation_data=(val_data, to_categorical(val_labels)))

test_loss, test_acc = model.evaluate(val_data, to_categorical(val_labels))
print('Test accuracy:', test_acc)

model.summary()

#saving model

from keras.models import load_model

path = './train_models/characters_53_'
path += str("{:.2f}".format(test_acc*100))
path += '.h5'
print(path)
# model.save(path)
```

```
# #import model
model_path = path
imp_model = load_model(model_path)
```

```
imp_model.summary()



import os
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.utils import to_categorical
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout



path = './Vathulu/'
data = []
labels = []

#data aug

datagen_1 = ImageDataGenerator(
    horizontal_flip=True,
    vertical_flip=False
)



datagen_2 = ImageDataGenerator(
    horizontal_flip=False,
    vertical_flip=True
)
```

```
datagen_3 = ImageDataGenerator(
    horizontal_flip=True,
    vertical_flip=True
)
```

```
datagen_4 = ImageDataGenerator(
    rotation_range=45,
    horizontal_flip=True,
    vertical_flip=False,
    fill_mode='constant', cval=255
)

datagen_5 = ImageDataGenerator(
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=False,
    fill_mode='constant', cval=255
)

datagen_6 = ImageDataGenerator(
    rotation_range=45,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='constant', cval=255
)

datagen_7 = ImageDataGenerator(
    rotation_range=90,
    horizontal_flip=True,
    vertical_flip=True,
```

```
 fill_mode='constant', cval=255
)



for foldername in os.listdir(path):
    temp_path = path + foldername + '/'
    for i in os.listdir(temp_path):
```

```
        temp_path_1 = temp_path + i
        img = cv2.imread(temp_path_1)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        img = np.transpose(img)
        img = remove_blanks(img,len(img))
        img = np.transpose(img)
        img = remove_blanks(img,len(img))

        img = resize(img, 64)

        img = img.reshape(64, 64, 1)

        data.append(img)
        labels.append(int(foldername.split('_')[0])-1)

        if(foldername == '1_a'):
            continue

        img_aug = img.reshape((-1, 64, 64, 1))

        img_batch = datagen_1.flow(img_aug, batch_size=1).next()
        img_batch = datagen_2.flow(img_aug, batch_size=1).next()
```

```
img_batch = datagen_3.flow(img_aug, batch_size=1).next()

        for _ in range(10):
            img_batch = datagen_4.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

        for _ in range(10):
            img_batch = datagen_5.flow(img_aug, batch_size=1).next()
```

```
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

        for _ in range(10):
            img_batch = datagen_6.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

        for _ in range(10):
            img_batch = datagen_7.flow(img_aug, batch_size=1).next()
            data.append(img_batch[0])
            labels.append(int(foldername.split('_')[0])-1)

data = np.array(data)
labels = np.array(labels)

train_data, val_data, train_labels, val_labels = train_test_split(data, labels,
test_size=0.2, random_state=42)

train_data = train_data / 255.0
val_data = val_data / 255.0
```

```
train_data = train_data.reshape(-1, 64, 64, 1)
val_data = val_data.reshape(-1, 64, 64, 1)



model_2 = Sequential()
model_2.add(Conv2D(128, (3, 3), activation='relu', input_shape=(64, 64, 1)))
model_2.add(MaxPooling2D(pool_size=(2, 2)))

model_2.add(Conv2D(128, (3, 3), activation='relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model_2.add(Conv2D(64, (3, 3), activation='relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))

model_2.add(Flatten())
model_2.add(Dense(500, activation='relu'))
model_2.add(Dropout(0.5))

model_2.add(Dense(541, activation='relu'))

model_2.add(Dense(20, activation='softmax'))

model_2.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model_2.fit(train_data, to_categorical(train_labels), epochs=20,
validation_data=(val_data, to_categorical(val_labels)))

test_loss, test_acc = model_2.evaluate(val_data, to_categorical(val_labels))
```

```
print('Test accuracy:', test_acc)

model_2.summary()

#load models
from keras.models import load_model

model_ch = load_model('./train_models/characters_53_90.32.h5')
model_va = load_model('./train_models/vathulu_20_95.61.h5')

model_ch.summary()
model_va.summary()
```

```
import os
from IPython.display import clear_output

def pred_let(letter):
    res_img = resize(letter, 64)

#     plt.imshow(res_img,cmap='Greys')
#     print(res_img.shape)
#     plt.show()

    res_img = res_img.reshape(-1, 64, 64, 1)
    res_img = res_img/255.0
    arr = model_ch.predict(res_img)
    return (np.where(arr==arr.max())[1][:])[0]

def pred_vathu(letter):
    res_img = resize(letter, 64)
```

```
#     plt.imshow(res_img,cmap='Greys')
#     print(res_img.shape)
#     plt.show()

    res_img = res_img.reshape(-1, 64, 64, 1)
    res_img = res_img/255.0
    arr = model_va.predict(res_img)
    return (np.where(arr==arr.max())[1][:])[0]

def ascii2char(s):
    temp = s.split(' ')
    as2ch = ''
    for i in temp:
        as2ch += chr(int(i))
    return as2ch
```

```
def loadfile(path):
    f = open(path, 'r')
    temp_list = f.readlines()
    f.close()
    ch_arr = []
    for i in temp_list:
        ch = ascii2char(i)
        ch_arr.append(ch)
    return ch_arr



chars = loadfile('./train_models/characters.txt')
vathu = loadfile('./train_models/vathulu.txt')
```

```
out = ''

for line in seg_lines(input_img):
    for word in seg_words(line):
        for letter in seg_letters(word):
#             plt.rcParams['figure.figsize'] = (2,2)
#             plt.imshow(letter,cmap='Greys')
#             print(letter.shape)
#             plt.show()
# 14 ---> 47
            text = ''
            val_1 = pred_let(letter)
            text = chars[val_1]
            if(val_1>=14 and val_1 <= 47):
                val_2 = pred_vathu(letter)
                if(val_2 != 0):
                    text += vathu[val_2]
```

```
            out += text

        clear_output(wait=False)
        out += ' '
    out += '\n'
print(out)



out = out.replace('\n','')

from mtranslate import translate
en_text = translate(out, 'en')
print(en_text)
```

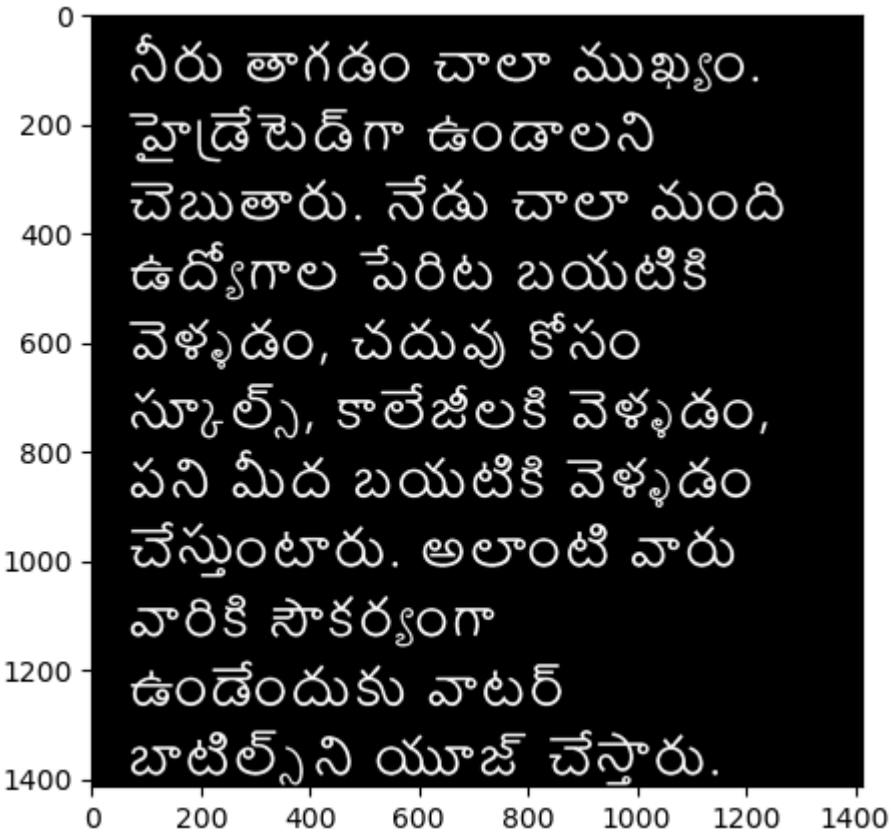
SCREENSHOTS

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 100)       57700

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 100)        0
 2D)

 flatten (Flatten)           (None, 3600)              0

 dense (Dense)               (None, 500)               1800500

 dropout (Dropout)           (None, 500)               0

 dense_1 (Dense)             (None, 53)                26553

=================================================================
Total params: 1,903,569
Trainable params: 1,903,569
Non-trainable params: 0
```

8.1 Architecture for model 1

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 62, 62, 32)        320

max_pooling2d_6 (MaxPooling  (None, 31, 31, 32)        0
2D)

conv2d_7 (Conv2D)            (None, 29, 29, 64)        18496

max_pooling2d_7 (MaxPooling  (None, 14, 14, 64)        0
2D)

conv2d_8 (Conv2D)            (None, 12, 12, 128)       73856

max_pooling2d_8 (MaxPooling  (None, 6, 6, 128)         0
2D)

flatten_2 (Flatten)          (None, 4608)              0

dense_4 (Dense)              (None, 500)               2304500

dropout_2 (Dropout)          (None, 500)               0

dense_5 (Dense)              (None, 541)               271041

=================================================================
Total params: 2,679,053
Trainable params: 2,679,053
Non-trainable params: 0
```

8.2 Architecture for model 2



8.3 INPUT IMAGE

```
from mtranslate import translate

en_text = translate(out, 'en')
print(en_text)
```

```
Drinking water is very important. It is said to stay hydrate
d. Today, many people have to go out in the name of work, st
op studying, go to college, go out for work. Sometimes they
use a water bottle as a convenience for them.
```

8.4 OUTPUT ENGLISH TEXT

C. RESEARCH PAPER

Telugu Text Recognition from Image using Deep Learning

Banda Madan Kumar
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
b.madankumar0@gmail.com

Bharatha Varshith Varma
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
varshithbharatha2@gmail.com

Dr. B. U. Anu Barathi
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
anubarathi.cse@sathyabama.ac.in

***Abstract*— Several researchers have attempted to solve the difficult challenge of text recognition in photographs by using computer vision. There has been a plenty of researches recently in deep neural network based research on scene text recognition. It is feasible to classify DLas a subfield of ML . Yet, the majority of studies focused on English text with little attention paid to any other languages. In this work we concentrate on the topic for Telugu Text recognition in photographs. Telugu is written in a complicated cursive script, with curving letters that often have both a top and bottom stroke. Optical character recognition (OCR) utilizing deep neural networks is proposed as a model in this article. Input images containing Telugu text are processed using the proposed model, after which the sequence of the appropriate recognized characters is extracted. To train the algorithm, we provide a fresh dataset comprised of photos that were manually cropped. Telugu text recognition has many practical uses in fields like as education, administration, and more for those who are not fluent in the language.**

Keywords— Optical character recognition (OCR), Telugu, Deep Learning.

INTRODUCTION

Telugu is the official language of the states of Telangana & Andhra Pradesh in southern India. It is also extensively spoken outside of these two states. Like other Dravidian languages, it has a rich and varied past that makes it distinct. It has the third-highest quantity of native speakers in India and the fifteenth-highest number on the Ethnologue list of the world's most spoken languages. Telugu has a huge number of characters, comprising top strokes, bottom strokes, and combination letters, making text identification difficult. These combined characters and letters have 253 individual strokes and are composed of 16 vowels (named achus) & 36 consonants (called hallus). There are 52 symbols (Aksharamulu) in the Telugu alphabet, 16 of which are vowels and 36 of which are consonants.

Most businesses these days scan their paper documents into digital files so that they can be updated, searched, and modified more quickly and for longer. Scanned copies of all of the world's books, manuscripts, and other historical documents can be found online & accessed from any location around the world. These digital scans will be informative and can be viewed in either colour or black and white. Several books have been digitised and made accessible online, but there are certain classics that can only be found in print. This model is taught to recognise Telugu characters, which it uses to build phrases and sentences that may be duplicated, used as tales in textbooks or newspapers, and edited as necessary.As more and more content is being created and shared in digital formats, there is a growing need for accurate and efficient methods of Telugu text recognition. Recognizing Telugu text in images can be difficult because of the intricate nature of the script as well as the variations in handwriting styles. However, the ability to accurately recognize Telugu text has the potential to improve accessibility, enable automated document processing, and facilitate translation and transcription tasks.

This research project focuses on developing a Telugu text recognition system that can take an image containing Telugu text as input, and output the recognized text in a readable format. The project involves multiple steps, including image processing, line segmentation, word segmentation, character recognition, and text reconstruction. The system is designed to handle different styles of handwriting and font types, and to provide accurate recognition results even in the presence of noise or other image artifacts.

The main objective of this research project is to develop a robust and efficient Telugu text recognition system that can be used in a variety of applications, including document processing, translation, and transcription. This research project builds on existing work in the field of Telugu text recognition, and seeks to address some of the limitations and challenges that have been identified in previous studies.

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers to create a solution, and all their solutions vary from analysing various patterns of text recognition.

Asghar Ali & Mark Pickering [1] presented a Hybrid deep Neural System for Character recognising of Arabic text in pictures of natural scenes, with a precision of 52.72% utilising ResNet block with Batch Normalization & 61.35% utilising Data Augmentation, despite the language's cursive, right-to-left writing style. A fresh collection of 11,500 images with clipped natural-scene Urdu texts was released. The network was trained using complete words as input rather than

character-based classification. To identify the Urdu dialogue, we use a skip-connected Hybrid deep neural network architecture. It combines a recurrent neural network with a convolutional neural network. Its architecture is made up of three primary parts: a convolutional block with a residual unit for feature extraction, a recurrent block for sequential training, and a transcription block for converting predicted labels into matching transcriptions. We apply data augmentation methods like contrast stretching & histogram equalisation to enhance the information set more significant. Some studies [2, 4] made use of OCR to determine whether or not an image included legible text. One such state-of-the-art method is using deep learning for OCR, which stands for optical character recognition. In [2] the authors surveys how to decipher text with OCR and Deep learning techniques, and used segmentation method it helps to analyze an image and understand the features easily by dividing the image into segments. This study showed various OCR techniques and comparisons between them. There is also a comparison of Deep learning techniques and their respective pros and cons of each of them.

Urdu text recognition on pictures from natural scene images was proposed like a 3 process by Asghar Ali Chandio [3]. Shortcut-connected CNN for feature extraction, RNN for feature decoding, and a CTC for mapping predicted sequences to target labels. Feature extraction, sequence labelling, and text transcription form the backbone of the system. Using a freshly compiled dataset for training and evaluation. Accuracy of 87.13 percent was accomplished using two sequence extraction strategies that focus on VGG-16 and ResNet-18 networks.

YumingHe [4] conducted research on OCR recognition technologies and found that text recognition incorporated with DL algos is more efficient with consistent noise immunity & resilient. This includes applications as diverse as facial recognition as well as target detection. This paper also compares and contrasts an amount of deep learning methods and techniques.

Urdu ligature recognition accuracy was improved to 80.46 percent after a hybrid model was devised by S. Y. Arafat and M. J. Iqbal [5]. 2 separate deep neural networks, Alexnet and VGG16, are used to extract information specific to each Ligature. 46K pictures were used for testing and training, with 9+ iterations of each ligature.

Reviews of detection & recognition techniques produced over the last decade are presented by H. Lin et al. [6]. The authors provide the findings of more than 40 typical approaches and provide a performance comparison.

S. B. Ahmed et al. developed a new method in [7] that extracts scale-invariant characteristics using an adaptation of the maximally stable extremal region (MSER) methodology. The Arabic text is the primary subject of this work, which features an English-Arabic scenario. Experiment results suggest that concentrating on a narrow topic for invariant features is necessary for a successful invariant feature extraction strategy. Error rates as high as 5.99% and 2.48%, respectively, have been reported in experiments involving the segmentation and identification of Arabic and English text. Multilingual script text identification using OCR technology was suggested by the authors of [11], who created a model for English and Oriya texts. The segmentation and noise-reduction processes may now be completed. They used structural analysis to tell English and Oriya apart from one another. Characters included structural elements like loops & cursive's emphasis on left and right lines.

Dr.M.Sundaresan and S.Ranjini [12] proposed a model for text extraction from an English comic image, A comic image with complex background and different styles of font, and extra objects such as persons, buildings, etc. Have used Median filter for pre-processing method to eliminate the noise and achieved an accuracy of 94.82%. In this study, the two Blob Extraction methods are employed to retrieve English text from a comic book picture.

M Abdul Rahiman, M S Rajasree from [13] have detected Malayalam text using OCR for printed text documents, the picture was pre-processing to eliminate noiseLines, phrases and characters were divided for processed picture. To complete recognition tasks, this model employs a Feed Forward Back propagation Neural Network with wavelet multi-resolution analysis for feature extraction. After training on 715 photos, this model achieved 92% accuracy.

Recognition of Telugu character are done by using Support Vector Machine (SVM) by Rajkumar. J [14] and Anitha Jayaraman [15]. Both used modular approach to classify the strokes of Telugu character, Rajkumar. J divided strokes into 4 sub classes and used two Schemas to compare, Schema 1 based upon TST while the other based on SVM. Anitha. J divided the strokes into 3 sub classes and used SVM. Both yielded an accuracy of 89.59% and 82.96%. When it comes to text with curved shape or rotated text it is hard to address features and output, apart from OCR, CNN, C-RNN techniques we have Single Visual model, Transformer-based, Visual-Semantic Transformer as used in [8]-[10].

Open Problems In Existing System

Many of the researches have been done on natural scene images and doesn't work when given an image with a sentence. Many research models trained model on word image directly rather than training a single character which may occur low accuracy for complex words which may increase error rate. There is less concentration on Telugu language, and it is different from other cursive languages because of types of strokes present in

Telugu. So, in this paper we propose a model that takes an image with Telugu text as input and then segments the sentences lines to words and characters and recognize the pattern and outputs the Telugu text.

PROPOSED SYSTEM

Data Collection and Preprocessing

The Data Collection and Preprocessing module is an important component of Telugu text recognition system. In this module, the focus is on collecting data and processing it to make it suitable for further analysis. The module has two major steps - data collection and preprocessing.

The primary goal of the data collection phase is to accumulate a comprehensive set of Telugu symbols. In order to train the character recognition system, we will use this dataset. Telugu has 16 vowels letters and 36 consonant letters. The "vathugunithalu" & the "main letter" make up the two halves of a Telugu syllable. When coupled with one of the other 36 consonant letters, the "vathu" takes on a unique pronunciation for each character. There are 541 different permutations of "vathus" in Telugu that seem the same, which increases the complexity and quantity of the dataset. Using each vathu as a class, there are 72 samples in the dataset, with 100-150 samples collected for each vowel and consonant. Books, newspapers, & the Internet were just some of the places we scoured for these pictures. The reliability of the recognition system is significantly impacted by the amount of the data.
Once the data is collected, firstly normalized the images and data augmented techniques used to increase accuracy of model the data then the images are converted to grayscale format. The reason behind converting images to grayscale format is to simplify the processing steps and reduce computational complexity. Grayscale images contain only one channel, whereas coloured images have 3 channels (red, green, and blue). By using grayscale images, we can eliminate the need to process three channels of color, which speeds up the processing time and reduces the computational burden.

After converting the images to grayscale format, noise reduction techniques such as Gaussian blur or median filtering are applied. These techniques are used to remove unwanted noise from the images. Noise can negatively affect the accuracy of character recognition, so it is important to remove it before further processing. Next, a thresholding technique such as OTSU is used to convert the images to binary format. This method is used to transform colour photos into monochrome ones. By establishing a threshold value, the foreground & background of a picture may be extracted using the thresholding method. Pixels with values higher than the threshold value are made white, while those with lower values are made black. Binary images are easier to process, and it simplifies the task of detecting lines and characters in the image.
Skew correction is also an important step in the preprocessing module. Skew correction is a technique used to correct the slant or tilt of an image. The purpose of this step is to ensure that the characters in the images are properly aligned and can be accurately recognized by the character recognition model. In summary, the Data Collection and Preprocessing module plays a crucial role in the Telugu text recognition system. The module collects a large dataset of Telugu characters, processes the images to make them suitable for further analysis, and performs skew correction to ensure accurate recognition of characters.

Line ,word and character segmentation.

Line and word segmentation are important steps in the Telugu text recognition process. These techniques are used to isolate individual lines and words in an image containing Telugu text, allowing the system to accurately recognize and extract the text.
Line segmentation involves identifying the non-zero pixels in each row of the binary image. This technique allows the system to detect the lines of text in the image, which is a crucial first step in the recognition process. The non-zero pixels indicate the presence of text, and by analyzing the distribution of these pixels, the system can accurately detect the boundaries between lines of text. However, line segmentation alone is not sufficient for accurate text recognition. Words are often separated by spaces, which must be detected to accurately segment the text into individual words. This is where word segmentation comes into play. Word segmentation is performed on the lines of text detected in the previous step.

There are several techniques used in word segmentation, here we have used character spacing. In character spacing, the non-zero pixels in each column of the binary image are analyzed to detect spaces between words. This technique assumes that the space between words is larger than the space between individual characters within a word. By analyzing the distribution of non-zero pixels in each column, the system can accurately detect the spaces between words and segment the text into individual words.

Finally, character segmentation is the process of detecting and isolating individual characters within each word. In the Telugu text recognition system, character segmentation is performed using

techniques such as Optical Character Recognition (OCR) or deep learning-based approaches.

In conclusion, line, word, and character segmentation are essential steps in the Telugu text recognition system. Accurate segmentation allows the system to isolate and recognize individual characters, which is critical for extracting text from images accurately. By using techniques such as morphology operations, character spacing, and OCR, the system can achieve high accuracy and efficiency in recognizing Telugu text.

Character recognition.

Training two separate models for recognizing Telugu characters and vathugunithalu is an effective approach to achieving high accuracy in character recognition. The architecture for each model, consists of three convolutional layers.

Convolutional layers are used in neural networks to learn patterns in the input data. By applying a set of filters to the input image, the network can detect edges, shapes, and other features that are relevant to the recognition task. Max pooling is then used to downsample the feature maps, reducing the dimensionality of the input data and improving the speed and efficiency of the model.

The architecture for the first model used in Telugu character recognition consists of three convolutional layers, followed by max pooling and a dense layer.
The first convolutional layer has 32 filters, which are applied to the input image to detect patterns and features. The size of each filter is typically small, such as 3x3 or 5x5, allowing the model to capture local patterns in the input image. After the filters are applied, the resulting feature maps are downsampled using max pooling with a window size of 2x2. Max pooling helps to reduce the size of the feature maps and prevent overfitting, while still preserving the most relevant information.
The second convolutional layer has 64 filters, which are applied to the downsampled feature maps from the previous layer. This layer is designed to detect higher-level patterns and features that may be spread across larger areas of the input image. After the filters are applied, the resulting feature maps are again downsampled using max pooling with a window size of 2x2.
The final convolutional layer also has 100 filters and it is applied to the downsampled feature maps from the previous layer. This layer further refines the learned features, and the resulting feature maps are again downsampled using max pooling with a window size of 2X2.
The final max pooling layer's output is fed into a dense layer after being flattened into a 1D array. In order to calculate the weighted total of the inputs, the dense layer, which consists of 128 neurons, employs the ReLU activation function. The output of the dense layer is transmitted onto the output layer, which consists of 53 neurons, one for each of the 53 distinct Telugu characters. The weighted total is transformed into a probability distribution across the 53 character classes via the Softmax activation function in the output layer.

(I) **Architecture for model 1**

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	320
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 100)	57700
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 100)	0
flatten (Flatten)	(None, 3600)	0
dense (Dense)	(None, 500)	1800500
dropout (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 53)	26553

Total params: 1,903,569
Trainable params: 1,903,569
Non-trainable params: 0

The architecture for the second model in Telugu text recognition, which is designed to recognize vathulu, consists of three convolutional layers followed by max pooling, and a dense layer.

Low-level characteristics like edges & corners are extracted from the input picture using 32 filters in the first convolutional layer. Different filters are trained to identify various elements of an image. To further facilitate the model's ability to learn complicated patterns, the layer's output is subsequently fed into ReLU activation function.

The output of the first convolutional layer is sent into a second convolutional layer, which uses 64 filters to extract more complicated information. Before being passed on to the following layer, the output of this layer is additionally activated with a ReLU function.
The third convolutional layer adds additional 128 filters to the output of the second layer, further extracting high-level characteristics. The layer's output is fed into a second ReLU activation layer.

The max pooling layer is used to reduce the spatial dimensionality of the feature maps and extract the most important features after the last convolutional layer. The maximum value of each group of four adjacent pixels is taken for the max pooling operation, which is carried out with a 2X2 window. The process concludes with a thick layer applied to the vectorized output. The dense layer is a totally

connected layer that uses the features learnt within the convolutional layers to make a classification decision. The thick layer is then taught to assign each incoming picture to one of 541 distinct classes, or vathulu.

(II) **Architecture for model 2**

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 62, 62, 32)	320
max_pooling2d_6 (MaxPooling 2D)	(None, 31, 31, 32)	0
conv2d_7 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_7 (MaxPooling 2D)	(None, 14, 14, 64)	0
conv2d_8 (Conv2D)	(None, 12, 12, 128)	73856
max_pooling2d_8 (MaxPooling 2D)	(None, 6, 6, 128)	0
flatten_2 (Flatten)	(None, 4608)	0
dense_4 (Dense)	(None, 500)	2304500
dropout_2 (Dropout)	(None, 500)	0
dense_5 (Dense)	(None, 541)	271041

Total params: 2,679,053
Trainable params: 2,679,053
Non-trainable params: 0

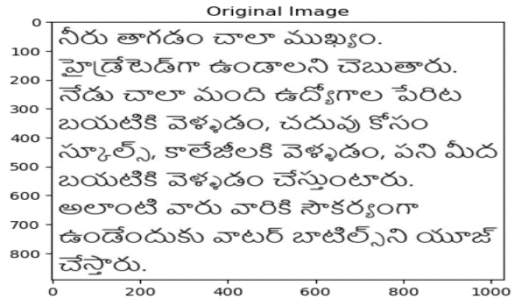
The use of multiple convolutional layers followed by max pooling allows the model to learn complex patterns and features in the input data, improving its performance on the recognition task. The model's capability to discover non-linear connections between input and output is enhanced by the use of the ReLU activation function.. The final dense layer performs classification based on the learned features, allowing the model to accurately recognize the vattus and gunithams in Telugu text.

After the convolutional layers, the picture is next flattened and fed into thick layers. The thick layers are in charge of learning high-level representation of the input data & are crucial for classification tasks.

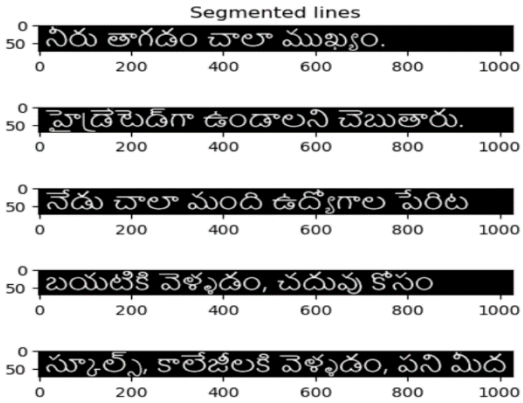
The dense layer is used to perform classification based on the learned features. In the case of the Telugu character recognition model, the dense layer is trained to classify each input image into one of 53 unique classes, representing each Telugu character. In the vathugunithalu recognition model, the dense layer is trained to classify each input image into one of 541 unique classes, representing each vathu or gunitham.

RESULTS AND DISCUSSIONS

The first step is to segment the image into lines, then words, & lastly letters. The image is pre-processed by converting it to grayscale and then to a binary image using OTSU thresholding. The image is segmented into lines by checking the number of non-zero pixels for each row of the NumPy array, which contains only 0's and 1's from the binarization process.



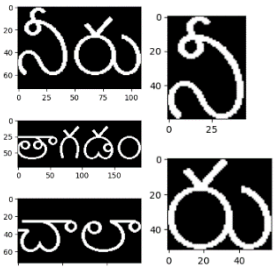
After the input image has been converted into a binary image using a thresholding technique, line segmentation can be performed by analyzing the non-zero pixels in each row of the image. A row with at least one non-zero and its preceding row with all zeros is considered the start point, and a row with at least one non-zero and its succeeding row with all zeros is considered the end point. The NumPy array is sliced using these start and end indices to split the image into lines.



After line segmentation then word segmentation is done ,To perform word segmentation, the binary image of the line of text is scanned from left to right. At each column, the amount of non-zero pixels is counted. When the amount of non-zero pixels exceeds a threshold, a new word is assumed to have started. The threshold can be set based on the average width of characters in the line of text. Once the start and end points of each word have been identified, the corresponding pixels can be extracted from the binary image to form a separate image of each word

Character segmentation is the process of identifying and separating individual characters within a word or line of text. This process is typically performed after the image has been converted into a binary image and the words have been segmented. Character segmentation can be achieved using Connected component analysis involves identifying connected regions of non-zero pixels within the binary image and separating them into individual characters.

After word segmentation is done then character recognition takes place and text reconstruction. To reconstruct text, we use two models that are designed to identify each character and combine them with appropriate spaces to form words.



(i) Segmented words (ii) Segmented letters

Finally the output text Telugu text is translated into the English language. Converting Telugu text to English text is a useful phase of the procedure of Telugu information extraction, as it allows the recognized text to be easily understood and analyzed by English-speaking users. This can be achieved using various techniques and libraries, such as the mtranslate library.

The mtranslate library is a Python library that allows for the translation of text from one language to another. To convert Telugu text to English text using the mtranslate library, the recognized Telugu text is passed as an input to the library's translation function, along with the source and target languages. The function then returns the translated text in the target language, in this case, English.

REFERENCES

[1] Asghar Ali, Mark Pickering, “A Hybrid Deep Neural Network for Urdu Text Recognition in Natural Images”, 2019.

[2] Muni Sekhar Velpuru, Priyadarshini Chatterjee, G Tejasree, M Ravi Kumar, S Nageswara Rao, “Comprehensive study of Deep learning based Telugu OCR”, 2020.

[3] ASGHAR ALI CHANDIO, MD. ASIKUZZAMAN, MARK R. PICKERING, (Member, IEEE), AND MEHWISH LEGHARI, “Cursive Text Recognition in Natural Scene Images Using Deep Convolutional Recurrent Neural Network”, 2022.

[4] Yuming He, “Research on Text Detection and Recognition Based on OCR Recognition Technology”, 2020.

[5] SYED YASSER ARAFAT AND MUHAMMAD JAVED IQBAL, “Two Stream Deep Neural Network for Sequence-Based Urdu Ligature Recognition”, 2019.

[6] Han Lin, Peng Yang, Fanlong Zhang, “Review of Scene Text Detection and Recognition”, 2019.

[7] SAAD BIN AHMED, SAEEDA NAZ, MUHAMMAD IMRAN RAZZAK, AND RUBIYAH BTE YUSOF, “A Novel Dataset for English-Arabic Scene Text Recognition (EASTR)-42K and Its Evaluation Using Invariant Feature Extraction on Detected Extremal Regions”, 2019.

[8] Xin Tang, Yongquan Lai, Ying Liu, Yuanyuan Fu, Rui Fang, “Visual-Semantic Transformer for Scene Text Recognition”, 2021.

[9] Yue Tao, Zhiwei Jia, Runze Ma, and Shugong Xu, “TRIG: Transformer-Based Text Recognizer with Initial Embedding Guidance”, 2021.

[10] Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, Yu-Gang Jiang, “SVTR: Scene Text Recognition with a Single Visual Model", 2020.

[11] Sanghamitra Mohanty ,Himadri Nandini Dasbebartta,Tarun Kumar Behera, “An Efficient Bilingual Optical Character Recognition (English-Oriya) System for Printed Documents”, 2009.

[12] Dr.M.Sundaresan, S.Ranjini, "Text Extraction from Digital English Comic Image Using Two Blobs Extraction Method", 2012.

[13] M Abdul Rahiman, M S Rajasree, "Printed Malayalam Character Recognition Using Back-propagation Neural", 2009.

[14] Rajkumar.J, Mariraja K., Kanakapriya,K., Nishanthini, S., Chakravarthy, V.S, "Two Schemas for Online Character Recognition of Telugu script based on Support Vector Machines", 2012.

[15] Anitha Jayaraman, C. Chandra Sekhar, "Modular Approach to Recognition of Strokes in Telugu Script", 2007.

Telugu Text Recognition from Image using Deep Learning

Banda Madan Kumar
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
b.madankumar0@gmail.com

Bharatha Varshith Varma
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
varshithbharatha2@gmail.com

Dr. B. U. Anu Barathi
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai, India
anubarathi.cse@sathyabama.ac.in

***Abstract*— Several researchers have attempted to solve the difficult challenge of text recognition in photographs by using computer vision. There has been a plenty of researches recently in deep neural network–based research on scene text recognition. It is feasible to classify Deep Learning as a subfield of Machine Learning. Yet, the majority of studies focused on English text with little attention paid to any other languages. In this work we concentrate on the topic for Telugu Text recognition in photographs. Telugu is written in a complicated cursive script, with curving letters that often have both a top and bottom stroke. Optical character recognition (OCR) utilizing deep neural networks is proposed as a model in this article. Input images containing Telugu text are processed using the proposed model, after which the sequence of the appropriate recognized characters is extracted. To train the algorithm, we provide a fresh dataset comprised of photos that were manually cropped. Telugu text recognition has many practical uses in fields like as education, administration, and more for those who are not fluent in the language.**

Keywords— Optical character recognition (OCR), Telugu, Deep Learning.

I. INTRODUCTION

Telugu is the official language of the states of Telangana & Andhra Pradesh in southern India. It is also extensively spoken outside of these two states. Like other Dravidian languages, it has a rich and varied past that makes it distinct. It has the third-highest quantity of native speakers in India and the fifteenth-highest number on the Ethnologue list of the world's most spoken languages. Telugu has a huge number of characters, comprising top strokes, bottom strokes, and combination letters, making text identification difficult. These combined characters and letters have 253 individual strokes and are composed of 16 vowels (named achus) & 36 consonants (called hallus). There are 52 symbols (Aksharamulu) in the Telugu alphabet, 16 of which are vowels and 36 of which are consonants.

Most businesses these days scan their paper documents into digital files so that they can be updated, searched, and modified more quickly and for longer. Scanned copies of all of the world's books, manuscripts, and other historical documents can be found online & accessed from any location around the world. These digital scans will be informative and can be viewed in either colour or black and white. Several books have been digitised and made accessible online, but there are certain classics that can only be found in print. This model is taught to recognise Telugu characters, which it uses to build phrases and sentences that may be duplicated, used as tales in textbooks or newspapers, and edited as necessary.As more and more content is being created and shared in digital formats, there is a growing need for accurate and efficient methods of Telugu text recognition. Recognizing Telugu text in images is a challenging task due to the complex nature of the script and the variations in handwriting styles. However, the ability to accurately recognize Telugu text has the potential to improve accessibility, enable automated document processing, and facilitate translation and transcription tasks.

This research project focuses on developing a Telugu text recognition system that can take an image containing Telugu text as input, and output the recognized text in a readable format. The project involves multiple steps, including image processing, line segmentation, word segmentation, character recognition, and text reconstruction. The system is designed to handle different styles of handwriting and font types, and to provide accurate recognition results even in the presence of noise or other image artifacts.

The main objective of this research project is to develop a robust and efficient Telugu text recognition system that can be used in a variety of applications, including document processing, translation, and transcription. This research project builds on existing work in the field of Telugu text recognition, and seeks to address some of the limitations and challenges that have been identified in previous studies.

II. LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers to create a solution, and all their solutions vary from analysing various patterns of text recognition.

Asghar Ali & Mark Pickering [1] presented a Hybrid deep Neural System for Character recognising of Arabic text in pictures of natural scenes, with a precision of 52.72% utilising ResNet block with Batch Normalization & 61.35% utilising Data Augmentation, despite the language's cursive, right-to-left writing style. A fresh collection of 11,500 images with clipped natural-scene Urdu texts was released. The network was trained using complete words as input rather than character-based classification. To identify the Urdu dialogue, we use a skip-connected Hybrid deep neural network architecture. It combines a recurrent neural network with a convolutional neural network. Its architecture is made up of three primary parts: a convolutional block with a residual unit for feature extraction, a recurrent block for sequential training, and a transcription block for converting

predicted labels into matching transcriptions. We apply data augmentation methods like contrast stretching & histogram equalisation to enhance the information set more significant. Some studies [2, 4] made use of OCR to determine whether or not an image included legible text. One such state-of-the-art method is using deep learning for OCR, which stands for optical character recognition. In [2] the authors surveys how to decipher text with OCR and Deep learning techniques, and used segmentation method it helps to analyze an image and understand the features easily by dividing the image into segments. This study showed various OCR techniques and comparisons between them. There is also a comparison of Deep learning techniques and their respective pros and cons of each of them.

Urdu text recognition on pictures from natural scene images was proposed like a 3 process by Asghar Ali Chandio [3]. Shortcut-connected CNN for feature extraction, RNN for feature decoding, and a CTC for mapping predicted sequences to target labels. Feature extraction, sequence labelling, and text transcription form the backbone of the system. Using a freshly compiled dataset for training and evaluation. Accuracy of 87.13 percent was accomplished using two sequence extraction strategies that focus on VGG-16 and ResNet-18 networks.

YumingHe [4] conducted research on OCR recognition technologies and found that text recognition incorporated with DL algos is more efficient with consistent noise immunity & resilient. This includes applications as diverse as facial recognition as well as target detection. This paper also compares and contrasts an amount of deep learning methods and techniques.

Urdu ligature recognition accuracy was improved to 80.46 percent after a hybrid model was devised by S. Y. Arafat and M. J. Iqbal [5]. 2 separate deep neural networks, Alexnet and VGG16, are used to extract information specific to each Ligature. 46K pictures were used for testing and training, with 9+ iterations of each ligature.

Reviews of detection & recognition techniques produced over the last decade are presented by H. Lin et al. [6]. The authors provide the findings of more than 40 typical approaches and provide a performance comparison.

S. B. Ahmed et al. developed a new method in [7] that extracts scale-invariant characteristics using an adaptation of the maximally stable extremal region (MSER) methodology. The Arabic text is the primary subject of this work, which features an English-Arabic scenario. Experiment results suggest that concentrating on a narrow topic for invariant features is necessary for a successful invariant feature extraction strategy. Error rates as high as 5.99% and 2.48%, respectively, have been reported in experiments involving the segmentation and identification of Arabic and English text. Multilingual script text identification using OCR technology was suggested by the authors of [11], who created a model for English and Oriya texts. The segmentation and noise-reduction processes may now be completed. They used structural analysis to tell English and Oriya apart from one another. Characters included structural elements like loops & cursive's emphasis on left and right lines.

Dr.M.Sundaresan and S.Ranjini [12] proposed a model for text extraction from an English comic image, A comic image with complex background and different styles of font, and extra objects such as persons, buildings, etc. Have used Median filter for pre-processing process to remove noise and achieved an accuracy of 94.82%. In this study, the two Blob Extraction methods are employed to retrieve English text from a comic book picture.

M Abdul Rahiman, M S Rajasree from [13] have detected Malayalam text using OCR for printed text documents, the picture was pre-processing to eliminate noise. Lines, words and characters were segmented for processed picture. To complete recognition tasks, this model employs a Feed Forward Back-propagation Neural Network with wavelet multi-resolution analysis for feature extraction. After training on 715 photos, this model achieved 92% accuracy.

Recognition of Telugu character are done by using Support Vector Machine (SVM) by Rajkumar. J [14] and Anitha Jayaraman [15]. Both used modular approach to classify the strokes of Telugu character, Rajkumar. J divided strokes into 4 sub classes and used two Schemas to compare, Schema 1 based upon TST while the other based on SVM. Anitha. J divided the strokes into 3 sub classes and used SVM. Both yielded an accuracy of 89.59% and 82.96%. When it comes to text with curved shape or rotated text it is hard to address features and output, apart from OCR, CNN, C-RNN techniques we have Single Visual model, Transformer-based, Visual-Semantic Transformer as used in [8]-[10].

A. Open Problems In Existing System

Many of the researches have been done on natural scene images and doesn't work when given an image with a sentence. Many research models trained model on word image directly rather than training a single character which may occur low accuracy for complex words which may increase error rate. There is less concentration on Telugu language, and it is different from other cursive languages because of types of strokes present in Telugu. So, in this paper we propose a model that takes an image with Telugu text as input and then segments the sentences lines to words and characters and recognize the pattern and outputs the Telugu text.

III. PROPOSED SYSTEM

A. Data Collection and Preprocessing

The Data Collection and Preprocessing module is an important component of Telugu text recognition system. In this module, the focus is on collecting data and processing it to make it suitable for further analysis. The module has two major steps - data collection and preprocessing.

The primary goal of the data collection phase is to accumulate a comprehensive set of Telugu symbols. In order to train the character recognition system, we will use this dataset. Telugu has 16 vowels letters and 36 consonant letters. The "vathugunithalu" & the "main letter" make up the two halves of a Telugu syllable. When coupled with one of the other 36 consonant letters, the "vathu" takes on a unique pronunciation for each character. There are 541 different permutations of "vathus" in Telugu that seem the same, which increases the complexity and quantity of the dataset. Using each vathu as a class, there are 72 samples in the dataset, with 100-150 samples collected for each vowel and consonant. Books, newspapers, & the Internet were just some of the places we scoured for these pictures. The reliability of the recognition system is significantly impacted by the amount of the data.

Once the data is collected, firstly normalized the images and data augmented techniques used to increase accuracy of model the data then the images are converted to grayscale format. The reason behind converting images to grayscale format is to simplify the processing steps and reduce computational complexity. Grayscale images contain only one channel, whereas coloured images have 3 channels (red, green, and blue). By using grayscale images, we can eliminate the need to process three channels of color, which speeds up the processing time and reduces the computational burden.

After converting the images to grayscale format, noise reduction techniques such as Gaussian blur or median filtering are applied. These techniques are used to remove unwanted noise from the images. Noise can negatively affect the accuracy of character recognition, so it is important to remove it before further processing. Next, a thresholding technique such as OTSU is used to convert the images to binary format. This method is used to transform colour photos into monochrome ones. By establishing a threshold value, the foreground & background of a picture may be extracted using the thresholding method. Pixels with values higher than the threshold value are made white, while those with lower values are made black. Binary images are easier to process, and it simplifies the task of detecting lines and characters in the image.
Skew correction is also an important step in the preprocessing module. Skew correction is a technique used to correct the slant or tilt of an image. The purpose of this step is to ensure that the characters in the images are properly aligned and can be accurately recognized by the character recognition model.
In summary, the Data Collection and Preprocessing module plays a crucial role in the Telugu text recognition system. The module collects a large dataset of Telugu characters, processes the images to make them suitable for further analysis, and performs skew correction to ensure accurate recognition of characters.

B. Line ,word and character segmentation.

Line and word segmentation are important steps in the Telugu text recognition process. These techniques are used to isolate individual lines and words in an image containing Telugu text, allowing the system to accurately recognize and extract the text.
Line segmentation involves identifying the non-zero pixels in each row of the binary image. This technique allows the system to detect the lines of text in the image, which is a crucial first step in the recognition process. The non-zero pixels indicate the presence of text, and by analyzing the distribution of these pixels, the system can accurately detect the boundaries between lines of text. However, line segmentation alone is not sufficient for accurate text recognition. Words are often separated by spaces, which must be detected to accurately segment the text into individual words. This is where word segmentation comes into play. Word segmentation is performed on the lines of text detected in the previous step.

There are several techniques used in word segmentation, here we have used character spacing. In character spacing, the non-zero pixels in each column of the binary image are analyzed to detect spaces between words. This technique assumes that the space between words is larger than the space between individual characters within a word. By analyzing the distribution of non-zero pixels in each column, the system can accurately detect the spaces between words and segment the text into individual words.

Finally, character segmentation is the process of detecting and isolating individual characters within each word. In the Telugu text recognition system, character segmentation is performed using techniques such as Optical Character Recognition (OCR) or deep learning-based approaches.

In conclusion, line, word, and character segmentation are essential steps in the Telugu text recognition system. Accurate segmentation allows the system to isolate and recognize individual characters, which is critical for extracting text from images accurately. By using techniques such as morphology operations, character spacing, and OCR, the system can achieve high accuracy and efficiency in recognizing Telugu text.

C. Character recognition.

Training two separate models for recognizing Telugu characters and vathugunithalu is an effective approach to achieving high accuracy in character recognition. The architecture for each model, consists of three convolutional layers.

Convolutional layers are used in neural networks to learn patterns in the input data. By applying a set of filters to the input image, the network can detect edges, shapes, and other features that are relevant to the recognition task. Max pooling is then used to downsample the feature maps, reducing the dimensionality of the input data and improving the speed and efficiency of the model.

The architecture for the first model used in Telugu character recognition consists of three convolutional layers, followed by max pooling and a dense layer.
The first convolutional layer has 32 filters, which are applied to the input image to detect patterns and features. The size of each filter is typically small, such as 3x3 or 5x5, allowing the model to capture local patterns in the input image. After the filters are applied, the resulting feature maps are downsampled using max pooling with a window size of 2x2. Max pooling helps to reduce the size of the feature maps and prevent overfitting, while still preserving the most relevant information.
The second convolutional layer has 64 filters, which are applied to the downsampled feature maps from the previous layer. This layer is designed to detect higher-level patterns and features that may be spread across larger areas of the input image. After the filters are applied, the resulting feature maps are again downsampled using max pooling with a window size of 2x2.
The final convolutional layer also has 100 filters and it is applied to the downsampled feature maps from the previous

layer. This layer further refines the learned features, and the resulting feature maps are again downsampled using max pooling with a window size of 2X2.
The final max pooling layer's output is fed into a dense layer after being flattened into a 1D array. In order to calculate the weighted total of the inputs, the dense layer, which consists of 128 neurons, employs the ReLU activation function. The output of the dense layer is transmitted onto the output layer, which consists of 53 neurons, one for each of the 53 distinct Telugu characters. The weighted total is transformed into a probability distribution across the 53 character classes via the Softmax activation function in the output layer.

(I) **Architecture for model 1**

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	320
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 100)	57700
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 100)	0
flatten (Flatten)	(None, 3600)	0
dense (Dense)	(None, 500)	1800500
dropout (Dropout)	(None, 500)	0
dense_1 (Dense)	(None, 53)	26553

Total params: 1,903,569
Trainable params: 1,903,569
Non-trainable params: 0

The architecture for the second model in Telugu text recognition, which is designed to recognize vathulu, consists of three convolutional layers followed by max pooling, and a dense layer.

Low-level characteristics like edges & corners are extracted from the input picture using 32 filters in the first convolutional layer. Different filters are trained to identify various elements of an image. To further facilitate the model's ability to learn complicated patterns, the layer's output is subsequently fed into a rectified linear unit (ReLU) activation function.

The output of the first convolutional layer is sent into a second convolutional layer, which uses 64 filters to extract more complicated information. Before being passed on to the following layer, the output of this layer is additionally activated with a ReLU function.
The third convolutional layer adds additional 128 filters to the output of the second layer, further extracting high-level characteristics. The layer's output is fed into a second ReLU activation layer.

The max pooling layer is used to reduce the spatial dimensionality of the feature maps and extract the most important features after the last convolutional layer. The maximum value of each group of four adjacent pixels is taken for the max pooling operation, which is carried out with a 2X2 window.
The process concludes with a thick layer applied to the vectorized output. The dense layer is a fully connected layer that uses the features learnt in the convolutional layers to make a classification decision. The thick layer is then taught to assign each incoming picture to one of 541 distinct classes, or vathulu.

(II) **Architecture for model 2**

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 62, 62, 32)	320
max_pooling2d_6 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_7 (Conv2D)	(None, 29, 29, 64)	18496
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_8 (Conv2D)	(None, 12, 12, 128)	73856
max_pooling2d_8 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten_2 (Flatten)	(None, 4608)	0
dense_4 (Dense)	(None, 500)	2304500
dropout_2 (Dropout)	(None, 500)	0
dense_5 (Dense)	(None, 541)	271041

Total params: 2,679,053
Trainable params: 2,679,053
Non-trainable params: 0

The use of multiple convolutional layers followed by max pooling allows the model to learn complex patterns and features in the input data, improving its performance on the recognition task. The model's capability to discover non-linear connections between input and output is enhanced by the use of the ReLU activation function.. The final dense layer performs classification based on the learned features, allowing the model to accurately recognize the vattus and gunithams in Telugu text.

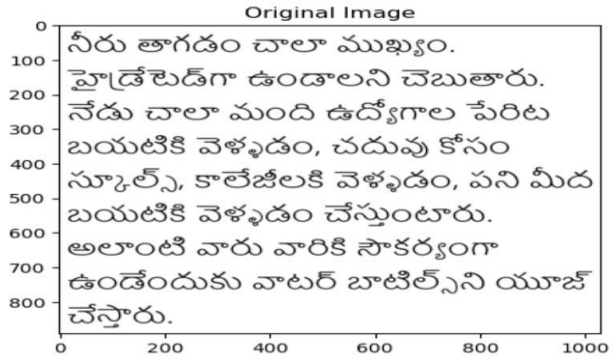
After the convolutional layers, the image is flattened and then fed into dense layers. The dense layers are responsible for learning high-level representations of the input data and are essential for classification tasks.

The dense layer is used to perform classification based on the learned features. In the case of the Telugu character recognition model, the dense layer is trained to classify each input image into one of 53 unique classes, representing each Telugu character. In the vathugunithalu recognition model, the dense layer is trained to classify each input image into one of 541 unique classes, representing each vathu or gunitham.

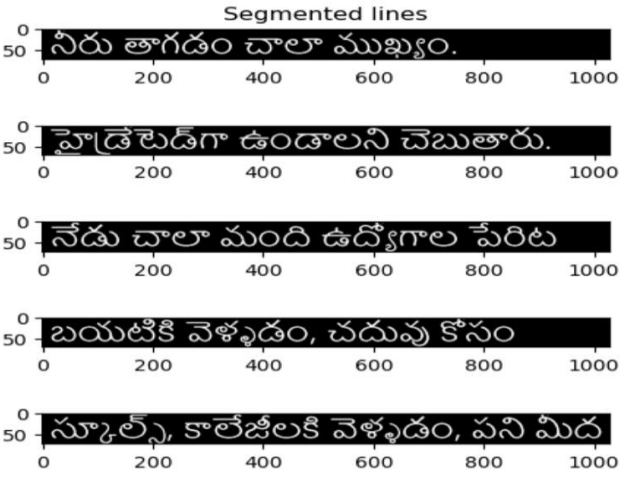
IV. RESULTS AND DISCUSSIONS

The first step is to segment the image into lines, then words, and finally letters. The image is pre-processed by converting it to grayscale and then to a binary image using OTSU thresholding. The image is segmented into lines by checking

the number of non-zero pixels in each row of the NumPy array, which contains only 0's and 1's from the binarization process.



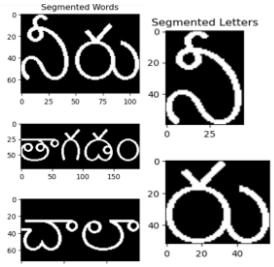
After the input image has been converted into a binary image using a thresholding technique, line segmentation can be performed by analyzing the non-zero pixels in each row of the image. A row with at least one non-zero and its preceding row with all zeros is considered the start point, and a row with at least one non-zero and its succeeding row with all zeros is considered the end point. The NumPy array is sliced using these start and end indices to split the image into lines.



After line segmentation then word segmentation is done ,To perform word segmentation, the binary image of the line of text is scanned from left to right. At each column, the number of non-zero pixels is counted. When the number of non-zero pixels exceeds a threshold, a new word is assumed to have started. The threshold can be set based on the average width of characters in the line of text. Once the start and end points of each word have been identified, the corresponding pixels can be extracted from the binary image to form a separate image of each word

Character segmentation is the process of identifying and separating individual characters within a word or line of text. This process is typically performed after the image has been converted into a binary image and the words have been segmented. Character segmentation can be achieved using Connected component analysis involves identifying connected regions of non-zero pixels within the binary image and separating them into individual characters.

After word segmentation is done then character recognition takes place and text reconstruction. To reconstruct text, we use two models that are designed to identify each character and combine them with appropriate spaces to form words.



(i) Segmented words (ii) Segmented letters

Finally the output text telugu text is translated into English language. Converting Telugu text to English text is a useful step in the Telugu text recognition process, as it allows the recognized text to be easily understood and analyzed by English-speaking users. This can be achieved using various techniques and libraries, such as the mtranslate library.

The mtranslate library is a Python library that allows for the translation of text from one language to another. To convert Telugu text to English text using the mtranslate library, the recognized Telugu text is passed as an input to the library's translation function, along with the source and target languages. The function then returns the translated text in the target language, in this case, English.

REFERENCES

[1] Asghar Ali, Mark Pickering, “A Hybrid Deep Neural Network for Urdu Text Recognition in Natural Images”, 2019.

[2] Muni Sekhar Velpuru, Priyadarshini Chatterjee, G Tejasree, M Ravi Kumar, S Nageswara Rao, “Comprehensive study of Deep learning based Telugu OCR”, 2020.

[3] ASGHAR ALI CHANDIO, MD. ASIKUZZAMAN, MARK R. PICKERING, (Member, IEEE), AND MEHWISH LEGHARI, “Cursive Text Recognition in Natural Scene Images Using Deep Convolutional Recurrent Neural Network”, 2022.

[4] Yuming He, “Research on Text Detection and Recognition Based on OCR Recognition Technology”, 2020.

[5] SYED YASSER ARAFAT AND MUHAMMAD JAVED IQBAL, “Two Stream Deep Neural Network for Sequence-Based Urdu Ligature Recognition”, 2019.

[6] Han Lin, Peng Yang, Fanlong Zhang, “Review of Scene Text Detection and Recognition”, 2019.

[7] SAAD BIN AHMED, SAEEDA NAZ, MUHAMMAD IMRAN RAZZAK, AND RUBIYAH BTE YUSOF, “A Novel Dataset for English-Arabic Scene Text Recognition (EASTR)-42K and Its Evaluation Using Invariant Feature Extraction on Detected Extremal Regions”, 2019.

[8] Xin Tang, Yongquan Lai, Ying Liu, Yuanyuan Fu, Rui Fang, “Visual-Semantic Transformer for Scene Text Recognition”, 2021.

[9] Yue Tao, Zhiwei Jia, Runze Ma, and Shugong Xu, “TRIG: Transformer-Based Text Recognizer with Initial Embedding Guidance”, 2021.

[10] Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, Yu-Gang Jiang, “SVTR: Scene Text Recognition with a Single Visual Model", 2020.

[11] Sanghamitra Mohanty ,Himadri Nandini Dasbebartta,Tarun Kumar Behera, “An Efficient Bilingual Optical Character Recognition (English-Oriya) System for Printed Documents”, 2009.

[12] Dr.M.Sundaresan, S.Ranjini, "Text Extraction from Digital English Comic Image Using Two Blobs Extraction Method", 2012.

[13] M Abdul Rahiman, M S Rajasree, "Printed Malayalam Character Recognition Using Back-propagation Neural", 2009.

[14] Rajkumar.J, Mariraja K., Kanakapriya,K., Nishanthini, S., Chakravarthy, V.S, "Two Schemas for Online Character Recognition of Telugu script based on Support Vector Machines", 2012.
[15] Anitha Jayaraman, C. Chandra Sekhar, "Modular Approach to Recognition of Strokes in Telugu Script", 2007.