A Major Project Report On

**An Automated Annotation System for Document Text Images**

Submitted in fulfillment of the requirements for the award of the

#### Bachelor of Technology

In

#### Department of Computer Science and Engineering

##### By

**B. JHANSI LAKSHMI 20241A0566**

**B. SRIJA 20241A0564**

**MARIA JABEEN 20241A0593**

**K. KUSHI REDDY 20241A0583**

Under the Esteemed guidance of

## Dr. ASHLIN DEEPA R N

## Associate Professor



**Department of Computer Science and Engineering**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

**Bachupalli, Kukatpally, Hyderabad,Telangana,India,500090**

**2023-2024**

I



## GOKARAJU RANGARAJU

**INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous)**

## CERTIFICATE

This is to certify that the major project entitled “**An Automated Annotation System for Document Text Images**” is submitted by **B. Jhansi Lakshmi (20241A0566), B. Srija (20241A0564), Maria Jabeen(20241A0593), K Kushi Reddy(20241A0583),** in fulfillment of the award of a degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2023-2024.**

INTERNAL GUIDE HEAD OF THE DEPARTMENT

**Dr. ASHLIN DEEPA R N Dr. B. SANKARA BABU**

**Associate Professor Professor**

**EXTERNAL EXAMINER**

II



Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **Dr. Aslin Deepa R N**, **Associate Professor**, Department of CSE for his support in the completion of our project report. We wish to express our honest and sincere thanks to **Dr. B. Sankara Babu, HOD,** Department of CSE, and to our principal **Dr. J. Praveen** for providing the facilities to complete our major project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

**B. Jhansi Lakshmi (20241A0566)**

**B. Srija (20241A0564)**

**Maria Jabeen(20241A0593)**

**K Kushi Reddy(20241A0583)**

III

## DECLARATION

We hereby declare that the major project entitled “**An Automated Annotation System for Document Text Images**” is the work done during the period from **2023-2024** and is submitted in the fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad).** The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

**B. Jhansi Lakshmi (20241A0566)**

**B. Srija (20241A0564)**

**Maria Jabeen(20241A0593)**

**K Kushi Reddy(20241A0583**

IV

|  | Table of Contents |  |
| --- | --- | --- |
| Chapter | TITLE | Page No |
|  | Abstract | 1 |
| 1 | Introduction | 2-3 |
| 2 | System Requirements | 4-8 |
|  | 2.1 Software Requirements | 4-5 |
|  | 2.2 Hardware Requirements | 5-6 |
|  | 2.3 Data Set | 6-8 |
| 3 | Literature Survey | 9-11 |
| 4 | Propose Approach , Modules Description, and UML Diagrams | 12-24 |
|  | 4.1 Modules | 13-22 |
|  | 4.2 UML Diagrams | 23-24 |
| 5 | Implementation, Experimental Results &Test Cases | 25-39 |
| 6 | Conclusion and Future Scope | 40 |
| 7 | References | 42-47 |
|  | Appendix   1. Full Paper-Publication Proof 2. Snapshot Of the Result | 48-49 |

V

|  | **LIST OF FIGURES** |  |
| --- | --- | --- |
|  |  |  |
| **Fig No** | **Fig Title** | **Page No** |
| 1 | Architecture of our model | 12 |
| 2 | Example of De-GAN output | 14 |
| 3 | Working of CRAFT | 15 |
| 4 | Example of CRAFT output | 15 |
| 5 | Flow Of Image in the CRNN model | 18 |
| 6 | Text recognition system | 20 |
| 7 | Post-Ocr module architecture | 22 |
| 8 | Use Case Diagram | 23 |
| 9 | Class Diagram | 23 |
| 10 | Data Flow Diagram | 24 |
| 11 | De-GAN Code Module 1 | 26 |
| 12 | De-GAN Code Module 2 | 26 |
| 13 | CRAFT Code Module 1 | 27 |
| 14 | CRAFT Code Module 2 | 28 |
| 15 | CRAFT Code Module 3 | 29 |
| 16 | Recognition Code Module | 30 |
| 17 | Annotation Code Module | 31 |
| 18 | Improper word detection by CRAFT | 31 |
| 19 | Improper word detection by CRAFT | 32 |
| 20 | Results with different dataset sizes | 34 |
| 21 | Paper Acceptance for Review paper | 48 |
| 22 | Result | 49 |

|  | **LIST OF TABLES** |  |
| --- | --- | --- |
| **Table No** | **Table Name** | **Page No** |
| 1 | De-GAN Module Test Cases | 36 |
| 2 | CRAFT Module Test Cases | 37 |
| 3 | Recognition Module Test Cases | 38 |
| 4 | Annotation Module Test Cases | 38-39 |
| 5 | Post-OCR Module Test Cases | 39 |

VI

**Abstract**

Annotation, the process of adding details or labels to text. Unlike manual annotation, which is slow and error-prone, our aim is to develop a robust system capable of annotating Telugu document text images with minimal human intervention. Our focus has been on achieving precise word-level annotations for Telugu handwritten documents. This automated approach not only saves time and resources compared to manual annotation but also improves the accuracy of the annotations.

Existing OCR systems for Telugu face challenges related to accuracy and speed. While some systems are capable of recognizing printed Telugu text with reasonable accuracy, handwritten Telugu text presents greater difficulty due to variations in handwriting styles and quality. Our project addresses the challenges in OCR for Indian languages, aiming to enhance the precision and efficiency of annotation processes.

Our project employs a multi-stage approach that integrates various neural network architectures. We utilize Generative Adversarial Networks (DE-GAN) for image denoising, Character-Region Awareness for Text detection (CRAFT) for text detection, and a combination of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks for text recognition. Additionally, we employ an attention-based encoder-decoder model for post-OCR processing to increase recognition accuracy.

Our curated dataset comprises 17,449 denoised Telugu word-level images, derived from 437 handwritten pages, collected with the help of our friends and faculty of GRIET, contributing in writing the dataset from 8th Physics Grade Telugu school textbooks where each word image has been cleaned, recognized, and annotated. We achieved an impressive 86.92% accuracy in recognizing Telugu word-level images and a 97.37% accuracy in recognizing Telugu character-level images. These results demonstrate significant progress in the fields of image annotation and character recognition, particularly for Telugu handwritten text.

**Chapter 1**

**Introduction**

Optical Character Recognition (OCR) technology converts various document types into editable and searchable data. This process involves transforming images of typed, handwritten, or printed text into machine-encoded text using several techniques. OCR is crucial for historical records and the field has seen significant advancements [16].

OCR digitizes physical documents, making them easier to store, retrieve, and share electronically. This enhances searchability, improving accessibility and usability. OCR also reduces manual data entry, increasing efficiency and reducing errors, while playing a key role in archiving and preserving historical documents. Additionally, OCR supports language processing tasks like text-to-speech [31], translation [32], and text mining.

OCR for Indian languages has progressed but faces unique challenges due to their complexity and diversity. Indian scripts include Devanagari [33], Bengali [34], Tamil [35], Telugu [36], Kannada [37], Gujarati [38], Malayalam, Odia, Punjabi etc. Each script's writing styles complicate the OCR process. Challenges include the complexity of scripts, variability in fonts and handwriting compared to Latin scripts [39]. Many Indian languages are low-resource, lacking digital data for training effective OCR systems [40]. Indian languages are also morphologically rich, adding complexity to OCR systems, especially in multilingual regions where documents contain multiple scripts [41]. While OCR primarily focuses on printed text [15], Handwritten Text Recognition (HTR) specializes in handwritten content.

An optimized OCR system for handwritten Marathi text is detailed in [17]. HTR technology, using neural networks and machine learning, transcribes handwritten content, preserving historical documents' authenticity [4]. Telugu OCR [36] is essential for digitizing Telugu literature and official records, supporting language preservation and academic research. Developing robust HTR systems for Indian languages remains challenging, as shown by the online Telugu character-level handwritten text recognizer proposed in [18].

Regional OCR systems are vital for processing documents in various Indian languages, enabling digitization of diverse content and meeting local needs. Synthetic data generation [42] is crucial for training OCR systems for low-resource languages, improving model accuracy and handling variability in fonts and handwriting styles.

Preprocessing images before OCR is essential. We use DE-GAN[26] for denoising images and CRAFT[27] for text detection at the page level. This improves the input data quality, leading to more accurate OCR results.

Automated Annotation Systems [5] surpass manual annotation in efficiency and user-friendliness. They use NLP, machine learning, and data mining to streamline tasks like text analysis, information retrieval, and data organization. An automatic annotation-based approach for digitized text recognition is proposed in [10], improving accuracy and efficiency in annotating handwritten documents, preserving historical and cultural information.

Post-OCR correction methods [19] demonstrate significant improvements over baseline models. Single-source and multi-source variants improve accuracy, with multi-source variants benefiting from diverse information sources and high-resource translations, as shown in their superior performance for Ainu [30]. These findings highlight the effectiveness of model adaptations in enhancing recognition capabilities across various linguistic contexts.

We created a Telugu dataset from Telangana state board 8th-grade physics chapters 1 and 2, and did DE-GAN[26], CRAFT[27], and recognition of data collected from school textbooks

**Chapter 2**

**System Requirements**

**2.1 Software Requirements**

This section details the software requirements for each module involved in the project, including DE-GAN, CRAFT, Recognition, Annotation, and POST-OCR.

**DE-GAN Module:**

The DE-GAN module requires the following Python packages:

* imageio (v2.8.0)
* matplotlib (v3.1.3)
* numpy (v1.19.2) and numpy-base (v1.19.2)
* Pillow (v8.1.0)
* scipy (v1.4.1)
* TensorBoard (v1.13.1) and tensorboard-plugin-wit (v1.6.0.post3)
* TensorFlow (v1.13.1) and tensorflow-estimator (v1.13.0)
* tqdm (v4.46.0)

**CRAFT Module:**

The CRAFT (Character Region Awareness for Text detection) module requires:

* PyTorch (v0.4.1.post2) and torchvision (v0.2.1)
* opencv-python (v3.4.2.17)
* scikit-image (v0.14.2)
* scipy (v1.1.0)

**Recognition Module:**

The Recognition module for text recognition involves:

* future (v0.18.2)
* scikit-image (v0.19.3)
* colorama (v0.4.6)
* lmdb (v1.4.0)
* opencv-python-headless (v4.7.0.68)
* PyTorch (v1.13.1) and torchvision (v0.14.1)
* six (v1.16.0)
* matplotlib

**Annotation Module:**

For annotating images with recognized texts, the Annotation module requires:

* Python (v3.8)
* opencv-python-headless (v4.7.0.68)
* Pillow (v8.4.0)
* numpy (v1.21.2)

The POST-OCR, responsible for processing OCR results, requires:

* pdf2image
* google-cloud-vision

**2.2 Hardware Requirements**

This section outlines the hardware specifications recommended for effectively running the modules and its components.

Essential Hardware required are:

* CUDA-Enabled System:
  + GPU (Graphics Processing Unit)
    - Recommended GPUs: NVIDIA GTX 1080, RTX 2080, or the newer RTX 30 series GPUs
  + CUDA Version: CUDA 10.x or 11.x
* Memory (RAM):
  + Minimum: 4 GB RAM
  + Recommended: 8 GB RAM or more
* Storage:
  + SSD (Solid State Drive)
  + Recommended Storage: At least a 256 GB SSD

Additional Considerations:

* CPU (Central Processing Unit)
  + Recommendation: i5/i7
* Operating System:
  + Linux, Windows, and macOS
  + Recommendation: Ubuntu 18.04 or 20.04 LTS

**2.3 Data Set**

In the realm of text recognition, the availability of robust datasets is crucial. Datasets form the backbone of machine learning models, providing the essential training material needed to develop accurate and reliable OCR systems. However, existing online Telugu datasets are often simplistic and preprocessed, leading to a significant gap in the development of effective Telugu OCR solutions. To address this, we focused on a novel approach centered around school education, creating a dataset that thoroughly addresses these shortcomings.

This document presents a comprehensive overview of our compiled Telugu Handwritten Text Dataset. The dataset is crafted from 8th-grade Physics textbooks, with contributors transcribing content into paragraphs and chapters. It is collected with the help of faculty members and friends from GRIET, who contributed by writing 8th-grade Physics school notebooks, specifically Chapters 1 and 2, ensuring a comprehensive range of word lengths, from as short as two characters to considerably longer constructs.

The significance of this dataset extends beyond mere collection; it presents a diverse and rich collection of denoised word images, systematically categorized for training, testing, and validation. Designed for Telugu handwritten text recognition tasks, this dataset not only fills a critical gap but also opens new avenues for research and development in OCR technologies for regional languages. Through this initiative, we aim to enhance the accuracy and efficiency of Telugu text recognition systems, thereby contributing significantly to the field of computational linguistics and educational technology.

**Dataset Composition**

* **Page-Level Images:** 437 high-resolution images representing full pages of handwritten Telugu text. These images capture a diverse range of writing styles and content.
* **Word-Level Images:** 17,449 individual word images extracted from the page-level images. Each word image undergoes a denoising process to enhance clarity and readability.
* **Distinct Classes:** 1,015 unique categories comprising the extracted word images. This classification ensures the dataset represents a broad spectrum of Telugu vocabulary.

**Dataset Breakdown**

The dataset is divided into three distinct sets for training, testing, and validation:

1. **Training Set (12,524 Images, 562 Classes):** This set comprises the majority of the denoised word images through De-GAn, distributed across a significant portion of the total classes. By exposing the models to a vast and varied set of examples, the training set enables them to learn effective recognition patterns.
2. **Testing Set (3,485 Images, 893 Classes):** The testing set serves to evaluate the performance of trained models on unseen data. It includes denoised word images spanning a wider range of classes compared to the training set. This comprehensive evaluation helps assess the model's ability to generalize its recognition capabilities to new and potentially challenging scenarios.
3. **Validation Set (1,440 Images, 262 Classes):** The validation set plays a crucial role in fine-tuning the model during the training process. It allows researchers to adjust model parameters to prevent overfitting, where the model performs well on the training data but fails to generalize to unseen data.

**Significance of the Dataset**

This dataset offers a significant contribution to the field of Telugu handwritten text recognition. Its rich collection of denoised word images, categorized into distinct classes, provides a valuable resource for researchers and developers in the following ways:

* **Training and Evaluation:** The dataset's structure facilitates the training and evaluation of machine learning models designed for Telugu handwritten text recognition. Researchers can leverage the diverse range of images to ensure their models achieve high accuracy and generalization capabilities.
* **Advancement of OCR and Language Processing:** By enabling the development of robust systems, this dataset contributes to the field of OCR and language processing technologies for the Telugu language.

The Telugu Handwritten Text Dataset serves as a valuable resource for researchers and developers working on handwritten text recognition. With its comprehensive collection of denoised and categorized word images, this dataset fosters advancements in OCR and language processing technologies for the Telugu language.

**Chapter 3**

**Literature Survey**

In the current digital landscape, the demand for swift and accurate document handling has become paramount across various sectors ranging from administrative offices to academic institutions and libraries. This necessity has propelled the development of Optical Character Recognition (OCR) systems, which serve as indispensable tools for efficiently digitizing documents and extracting text from them. OCR systems are designed to seamlessly convert the text contained within document images, whether printed or handwritten, into machine-readable format, thereby facilitating easy storage, retrieval, and manipulation of textual information.

However, the effectiveness of OCR systems hinges not only on their ability to accurately recognize and transcribe text but also on their capacity to understand and contextualize the content they process. This is where the concept of automatic annotation of document text images becomes crucial. Automatic annotation involves the systematic addition of metadata, semantic context, and other relevant information to the text extracted from document images. By annotating text images automatically, OCR systems can enhance their accuracy and usability in several ways.

Choi and Kim (2012) delve into automatic image annotation using semantic text analysis[3]. Their work sheds light on the fusion of image analysis and natural language processing, revealing how semantic comprehension of text can enrich the annotation of images . By bridging these disciplines, they demonstrate the profound impact of linguistic understanding on visual interpretation, showcasing the potential for more nuanced and contextually informed annotations. Through their innovative approach, they underscore the transformative possibilities of integrating semantic analysis into image processing workflows.

Hu et al. (2015) introduce DocRicher[2], an automatic annotation system that leverages social media to annotate text documents . While [3] provided the importance of integrating semantic analysis into OCR systems,this innovative approach emphasizes the importance of integrating external contextual information, like social media trends or user-generated content, to annotate and enrich texts effectively. By incorporating these additional sources of context, the method enhances the relevance and depth of annotations, providing a more nuanced understanding of the text. Through this integration, OCR systems can generate more accurate and contextually relevant annotations, improving the overall usability of the technology.

Building upon the integration of external contextual information, the study by Mondal et al. (2023)[1] focuses on handwritten document images and proposes a method for automatic annotation at the word level . Automatic annotation systems are essential for optimizing OCR technology by enriching document text images with metadata, semantic context, and linguistic information. By automatically associating meaningful annotations with text images, these systems not only enhance recognition accuracy but also facilitate tasks such as context preservation and information retrieval. Their contribution to refining OCR capabilities underscores their critical role in streamlining document digitization and text extraction processes in various fields.

The advances described previously are further enhanced by the latest technological innovations. A hybrid CNN and RNN architecture[6,7], for example, tackles the challenges of recognizing handwritten and printed text .Language models like BERT and GPT for Indian languages, offering insights into their adaptability and performance in multilingual NLP tasks. [8] addresses this challenge by introducing an innovative approach using a Vertical Attention Network (VAN). VAN employs an attention mechanism focused on vertical text regions, enabling it to capture the intricate layout and structure of handwritten paragraphs, thereby eliminating the need for explicit segmentation.

In a study on Telugu language OCR[28,29], the VGGNET-16[28] architecture was employed, which comprises a total of six layers along with one input layer and one output layer, as depicted in Figure 3. The first layer of the architecture includes two convolutional layers followed by one max pooling layer. The second layer is structured similarly with two convolutional layers and one max pooling layer. The third layer consists of three convolutional layers and one max pooling layer. The fourth and fifth layers also each contain three convolutional layers and one max pooling layer. Finally, the last layer is made up of three dense layers. By training and testing data using the VGGNET-16 model, good accuracy was achieved in recognizing Telugu script.

The introduction of new metrics and learning approaches signifies another step forward. Traditional metrics like Word Error Rate (WER) and Character Error Rate (CER) are widely used in OCR experiments for assessing recognition accuracy [13]. The concept of bWER[14], a metric designed to assess word recognition independently of reading order, offers a refined evaluation method . Additionally, dealing with limited labeled data, especially for symbols or alphabets in regional languages, presents a common challenge in Handwritten Text Recognition (HTR). A few-shot learning approach[9] for HTR starts with a minimal number of labeled examples for each symbol and progressively refines its understanding as more labeled data becomes accessible .

After the evaluation of the model using the metrics , post OCR correction[19] is done .The goal of post correction is to reduce recognition errors in the initial transcription—often caused by low-quality scanning, physical deterioration of paper documents, or diverse layouts and typefaces (Dong and Smith, 2018). Our work focuses on utilizing post-correction to address the lack of OCR training data for endangered languages. By applying post-correction techniques, we aim to improve the accuracy of OCR systems even when training data is limited, ensuring better recognition and preservation of these languages.

In conclusion, the advancements in Optical Character Recognition (OCR) technology, particularly in automatic annotation systems, signify a pivotal shift in document handling paradigms across various sectors. By integrating semantic analysis, external contextual information, and innovative methodologies, OCR systems are not only enhancing text extraction accuracy but also revolutionizing document digitization processes. The implementation of post-OCR correction plays a crucial role in this transformation, particularly for underrepresented languages, by significantly enhancing the quality of text recognition and ensuring the accurate preservation of diverse linguistic heritage.

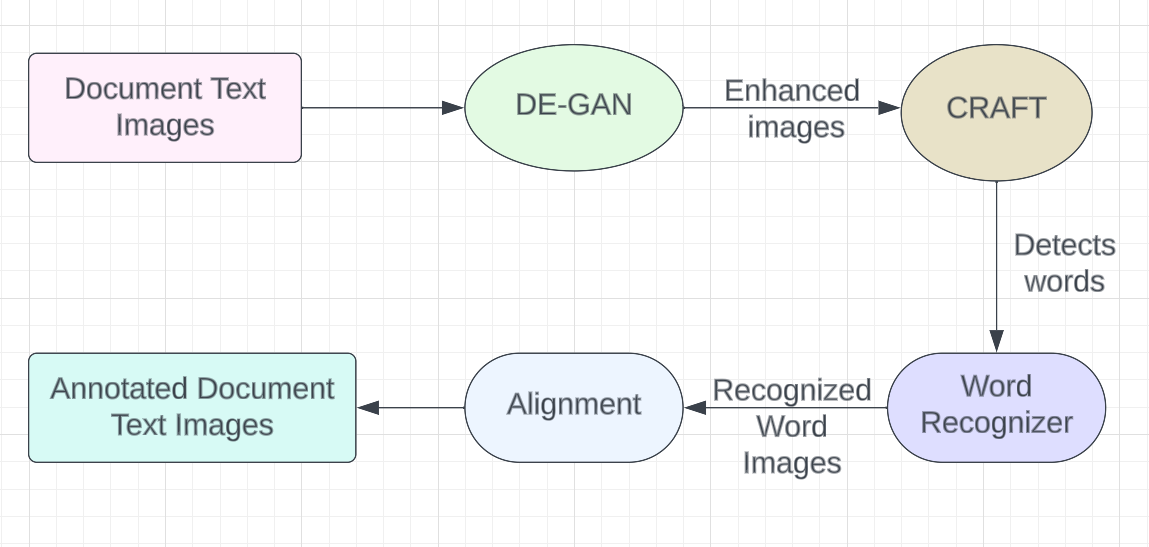
**Chapter 4**

**Proposed Approach:**

Annotation is the practice of supplementing text with extra details or labels to offer context, enhance comprehension, or assist in improvement of Natural Language Processing (NLP) models. In our project, we delve into the steps we have undertaken to achieve annotation specifically tailored for the Telugu language. It comprises four main modules aimed at enhancing the efficiency and accuracy of text recognition tasks, particularly in the context of handwritten documents. The modules involve denoising, localizing individual character regions, Convolutional Recurrent Neural Networks (CRNN) for accurate text extraction from images and precisely annotating relevant regions within the image, enhancing interpretability and enabling further analysis.

Our exploration extends to the algorithms we have employed, namely DEGAN[26] for preprocessing, CRAFT[27] for word detection, CRNN[1] for word recognition and post-OCR[19] for text correction and for enhancing accuracy. Through examination and comparison with alternative models, we aim to increase the efficiency of these approaches in the context of annotating Telugu language text.

**Architecture:**

****

**Fig 1: Architecture of our model**

**Modules Description and UML Diagrams:**

**4.1 Modules**

There are four main modules in our project.

**De-GAN Module:**

De-GAN[26] utilizes Generative Adversarial Networks (GANs) for denoising handwritten documents.It identifies and removes various types of noise, such as background artifacts and smudges. It enhances the overall quality of scanned documents, making them more legible and suitable for annotation.

The De-GAN model utilizes TensorFlow to create and implement a Generative Adversarial Network (GAN) for image enhancement tasks. The model uses below functions to enhance image.

1. generator\_model Function:

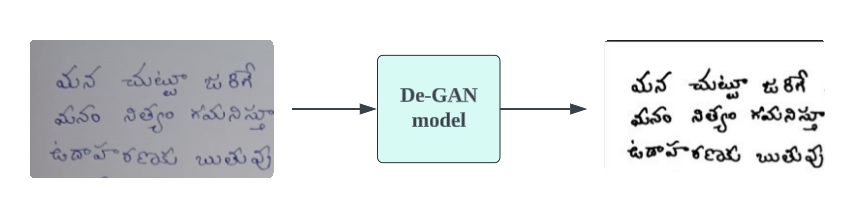
This function is responsible for creating a U-Net style generator model. U-Net architecture is commonly used for tasks like image segmentation and image enhancement. The generator model is designed to enhance images by encoding and decoding features through convolutional and pooling layers. These layers help in extracting and reconstructing image details for enhancement. It consists of multiple convolutional layers with ReLU activation function for feature extraction and a 'same' padding scheme to maintain spatial dimensions during convolution operations. The model incorporates dropout layers to prevent overfitting and upsampling layers to increase the spatial resolution of the feature maps.

2. discriminator\_model Function:

This function is responsible for creating the discriminator model for the GAN architecture. The discriminator model is designed to distinguish between real and generated images. It uses convolutional layers with LeakyReLU activation, a technique that allows a small gradient when the unit is not active to prevent sparse gradients.Batch normalization is used to standardize the inputs to each layer in the network, which stabilizes and speeds up the training process. The discriminator model ends with a convolutional layer that produces a single output value indicating the authenticity of the input image.

3. get\_gan\_network Function:

This function builds the GAN network by combining the generator and discriminator models. It sets the discriminator to be non-trainable during GAN training to prevent the discriminator from being updated. The function connects the output of the generator model to the input of the discriminator model, forming the adversarial network. The GAN model is compiled with appropriate loss functions, including binary cross-entropy for the discriminator and mean squared error for the generator, as well as an optimizer (Adam optimizer with a specific learning rate).

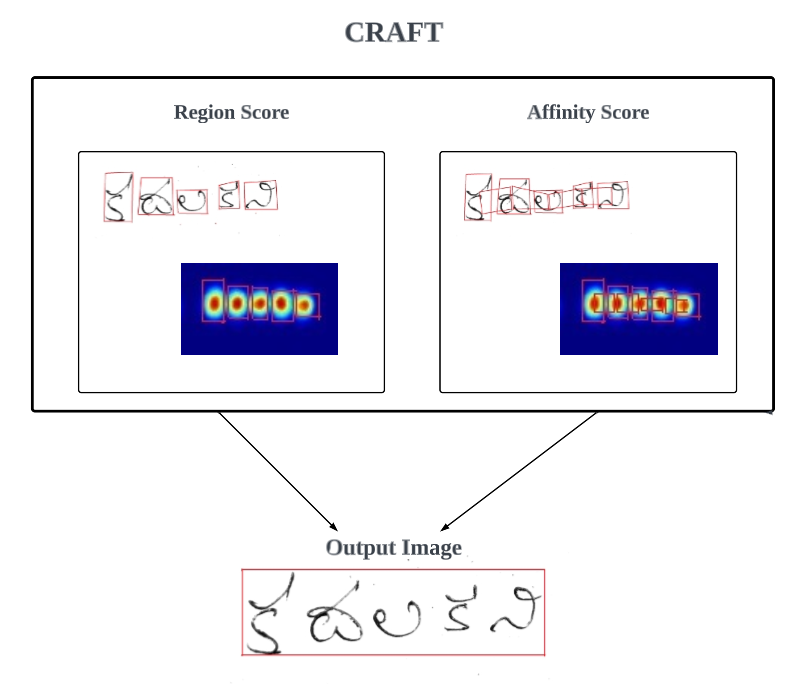


**Fig 2: Example of De-GAN[26] output**

**CRAFT Module:**

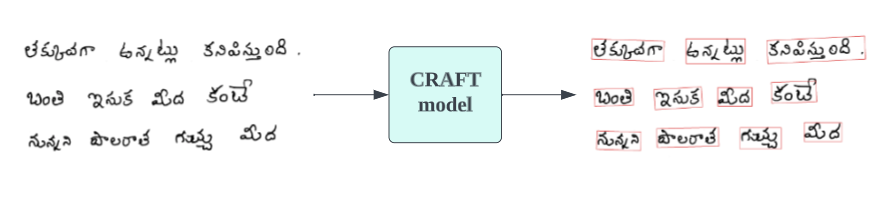
After binarizing the image using De-GAN, it is sent to the CRAFT model[27].

CRAFT is specifically designed to identify individual character regions within text and connect these detected characters to form complete text sequences. By focusing on character-level region awareness, it can easily represent text in various shapes. Unlike methods that treat text as a single entity, CRAFT hones in on the location of each character and its relationship with neighboring characters.



**Fig 3: Working of CRAFT[27]**

During training, CRAFT generates ground truth labels for two key components for each input image: the region score and the affinity score, using character-level bounding boxes. The region score indicates the probability that a specific pixel is the center of a character, while the affinity score measures the likelihood that a pixel is located at the center of the space between adjacent characters. This character-centric detection method enables convolutional filters to focus on both intra-character and inter-character details, rather than on the text as a whole, resulting in more accurate text detection.



**Fig 4: Example of CRAFT Output[27]**

The below utility scripts provide a toolkit for handling image processing tasks, specifically geared towards text detection and recognition in images.

1. crafts\_utils.py:

The crafts\_utils.py script contains functions for text detection and polygon extraction. It processes text and link score maps to identify text regions, refines detected boxes into polygons, and adjusts coordinates to match the original image size. Core functions include 'getDetBoxes\_cor' for text box detection and 'getPoly\_core' for polygon extraction, ensuring accurate text region mapping.

2. imgproc.py:

The imgproc.py script offers functions for image loading, normalization, resizing, and visualization. It includes 'loadImage' for handling various image formats, 'normalizeMeanVariance' for standardizing images, and 'resize\_aspect\_ratio' for maintaining aspect ratios while resizing. 'cvt2HeatmapImg' visualizes text regions effectively, preparing images for the detection pipeline.

3. file\_utils.py:

The file\_utils.py script focuses on file and directory management. 'get\_files' retrieves image, mask, and ground truth files, while 'list\_files' categorizes files based on extensions. The 'saveResult' function saves detection results, annotating images with detected polygons and corresponding text files, facilitating organized review and analysis.

4. test.py:

The test.py script integrates functionalities from the other scripts to create a comprehensive testing framework for text detection. It loads configurations and models, preprocesses images using 'imgproc.py', detects text regions with 'crafts\_utils.py', and saves results using 'file\_utils.py'. This script enables efficient testing and evaluation of text detection models.

Together, these scripts form a robust pipeline for text detection in images. `crafts\_utils.py` handles detection logic, `imgproc.py` provides image processing functions, `file\_utils.py` manages file operations, and `test.py` integrates these components for testing and evaluation. This toolkit ensures efficient processing and accurate results in text detection tasks.

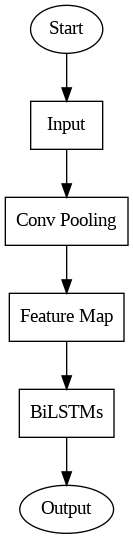
**Recognition Module :**

Our OCR system leverages a state-of-the-art text recognition model, integrating Convolutional Recurrent Neural Networks (CRNN) to accurately extract text from images. The CRNN architecture[1] is specifically designed for sequence-based prediction tasks, adeptly combining convolutional and recurrent layers to manage both spatial and sequential data aspects effectively.

The model's core comprises residual blocks, each featuring convolutional layers, batch normalization, and ReLU activation functions. These blocks are crucial for effective feature extraction, learning residual functions relative to the layer inputs, thus facilitating the training of deeper networks efficiently. Furthermore, the incorporation of affine transformations within the residual blocks enhances recognition accuracy and robustness by learning more complex feature representations. Max-pooling layers simplify the down-sampling process, reducing the spatial dimensions of the feature maps, which decreases computational load and aids in capturing the most salient features by retaining only the maximum values within each pooling window.

To capture sequence dependencies, the model integrates two bidirectional Long Short-Term Memory (LSTM) layers. These layers are vital for understanding context in sequential data, processing information in both forward and backward directions. Additionally, dropout regularization is applied to these layers to prevent overfitting, thereby enhancing the model's generalization capability. The input to the system is a denoised image, preprocessed to remove noise and improve the clarity of text features, and the output is the recognized text. Furthermore, a pretrained model was utilized for both training and testing, ensuring that the model benefits from prior learning, allowing it to achieve higher accuracy and robustness even with limited task-specific data.

The model also utilizes the Connectionist Temporal Classification (CTC) loss function during training to handle variable lengths of output sequences, making it robust to varying text lengths. CTC is specially designed for training models on sequence-to-sequence tasks with variable lengths and alignment issues, providing robustness and flexibility in handling diverse and unaligned data. It allows the model to predict a probability distribution over all possible alignments of the input and output sequences. Data augmentation techniques such as random rotation, scaling, and cropping are applied to the training data to improve robustness against different image distortions and variations. Leveraging pretrained weights from large text recognition datasets aids in achieving faster convergence and better initial performance.

****

**Fig 5: Flow Of Image in the CRNN model**

Optimized for hardware acceleration, the model runs efficiently on GPUs, leveraging parallel processing capabilities to speed up training and inference. It can also process multiple images in batches during inference, significantly reducing the time required for large-scale text extraction tasks. The diagram below illustrates the overall architecture of our CRNN recognition model, where the input denotes a denoised image that undergoes a series of convolutional and pooling operations for feature extraction, followed by sequential processing through bidirectional LSTM layers, ultimately resulting in the recognized text as the output.

By integrating these advanced components and techniques, our OCR system achieves high accuracy and efficiency in text recognition tasks, making it suitable for various applications such as document digitization and automated data entry.

**End-to-End Text Recognition Model Description:**

The end-to-end text recognition model comprises several Python scripts: config.py, lang\_train.py, utils.py, and model.py, each serving a distinct purpose in the system. Together, they provide a comprehensive solution for recognizing text from images.

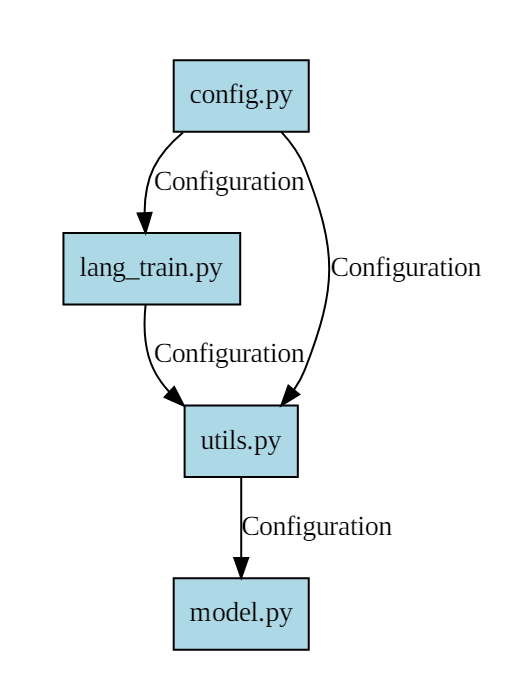
1. config.py: This script defines the configuration parameters and command-line arguments for the text recognition system. It allows users to specify various options such as dataset paths, model architecture, training parameters, and optimization settings. Key features include support for different alphabets, image preprocessing options, and training/validation intervals.

2.lang\_train.py: Responsible for training the text recognition model, lang\_train.py imports configurations from config.py and initiates the training process. It sets specific parameters such as alphabet type, input/output sizes, and the type of spatial transformer network (STN). Additionally, it defines transformation pipelines for data augmentation during training, tailored to different language groups.

3.utils.py: Similar to lang\_train.py, utils.py imports configurations from config.py but focuses on utility functions and setup. It establishes the necessary parameters for training, including alphabet type, dataset paths, and model architecture. Moreover, it configures data augmentation techniques based on the language being processed.

4.model.py: The core of the system, model.py constructs the integrated text recognition model. It defines the ModelBuilder class, which incorporates components such as convolutional recurrent neural networks (CRNN), spatial transformer networks (STN), and attention mechanisms. The model is capable of performing geometric transformations on input images before passing them through the CRNN for text recognition.

Together, these scripts form a cohesive framework for text recognition, enabling users to configure training settings, manage datasets, and build end-to-end text recognition models. By encapsulating functionality into modular components, the system offers flexibility and scalability for various text recognition tasks across different languages and datasets.



**Fig 6: Text recognition system**

**Annotation Module:**

We have implemented the annotation process by utilizing the recognized text from the OCR system and integrating it with the dimensions obtained from the CRAFT (Character Region Awareness for Text detection) module. This integration allows for precise annotation of the relevant regions within the image, thereby enhancing its interpretability and facilitating further analysis.

Using the CRAFT model, we accurately determined the coordinates of each handwritten word in the document. Once the words were recognized, they were placed on the document using these coordinates. The placement was adjusted meticulously to ensure that each recognized word was positioned directly above its corresponding handwritten word. This precise alignment not only improves the readability of the annotated document but also preserves the spatial context of the original handwritten text.

Our OCR system is complemented by an advanced post-OCR[19] correction model designed to enhance the accuracy of text extracted from images. This model employs a Character-Level Encoder-Decoder Architecture with Attention Mechanisms, targeting the reduction of errors in the initial OCR output. The primary goal is to refine the text recognition results, particularly when dealing with texts in endangered languages, where annotated data for training specific OCR systems is typically unavailable.

1. Model Initialization and Training:

The process begins with the initiation of the model, where an instance of the source model class is created. This is followed by the training phase, which utilizes both training and validation data. The model is trained in a supervised manner, with the training data comprising first pass OCR outputs as the source and the corresponding manually corrected transcriptions as the target. This approach ensures that the model learns to map erroneous OCR outputs to their correct forms. During the training process, we use pretraining with first pass OCR outputs before moving on to training with the manually corrected transcriptions. This two-step training enhances the model's ability to handle variations in the data and improves overall accuracy.

2.Encoder-Decoder Architecture with Attention:

The core of our post-correction model is a character-level encoder-decoder architecture augmented with attention mechanisms. This architecture is well-suited for sequence-to-sequence tasks, allowing the model to focus on relevant parts of the input sequence when generating each character in the output sequence. The attention mechanism is particularly beneficial in handling the irregularities and noise present in OCR outputs, by dynamically weighting the importance of different parts of the input.

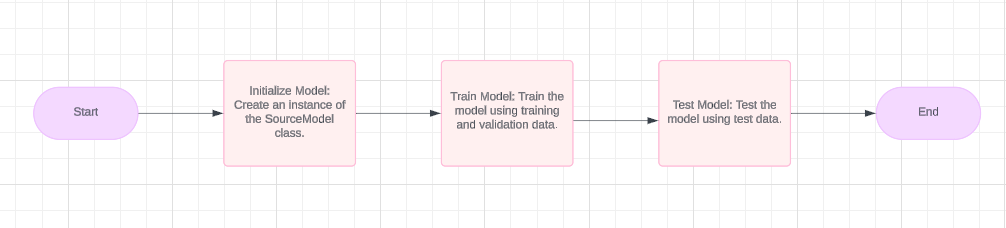
3.Handling Low-Resource Settings:

Given the scarcity of annotated data for endangered languages, our model incorporates several adaptations tailored for low-resource settings. One significant adaptation is the use of an additional encoder within a multisource framework. This additional encoder is leveraged when translations of the text in another high-resource language are available. By incorporating information from these translations, the model can further refine its corrections, leveraging the linguistic context provided by the translations.

4.Performance and Outcomes:

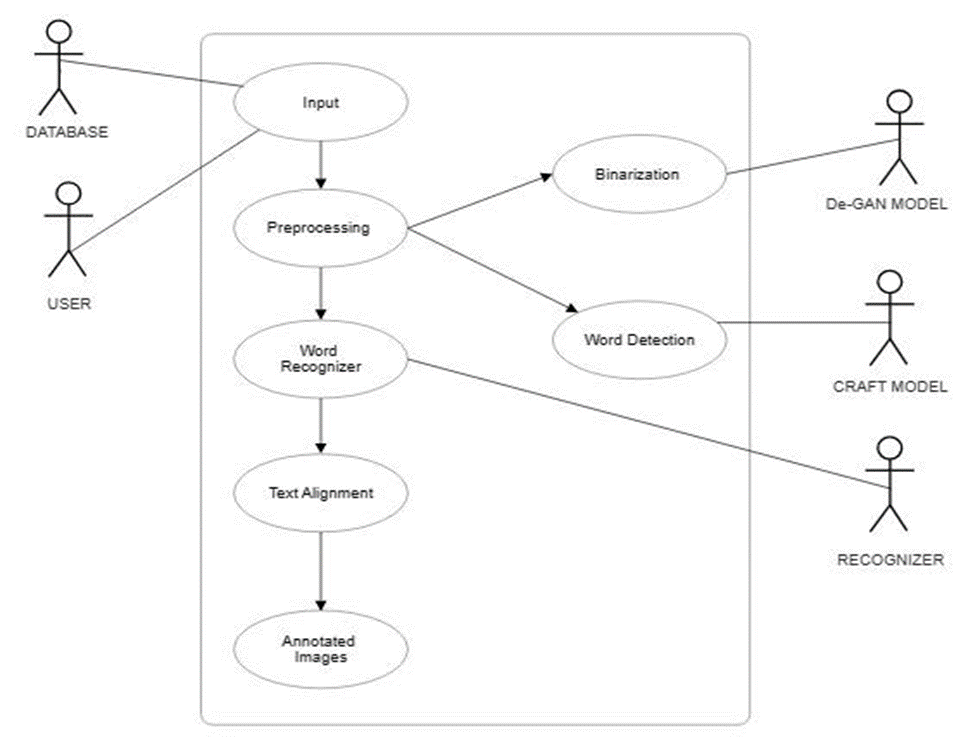
Our post-correction model significantly improves the accuracy of OCR outputs, reducing the recognition error rate. This improvement underscores the effectiveness of using a character-level encoder-decoder architecture with attention, especially when adapted for low-resource settings. The ability to correct OCR outputs accurately is crucial for making textual data in endangered languages machine-readable, thus preserving and facilitating the study of these languages.

The post-OCR correction model described here represents a robust solution to the challenges of text recognition in endangered languages. By leveraging a character-level encoder-decoder architecture with attention and incorporating adaptations for low-resource settings, the model significantly enhances the accuracy of OCR outputs. This advancement is crucial for preserving textual data in endangered languages and making it accessible for further linguistic research and documentation.

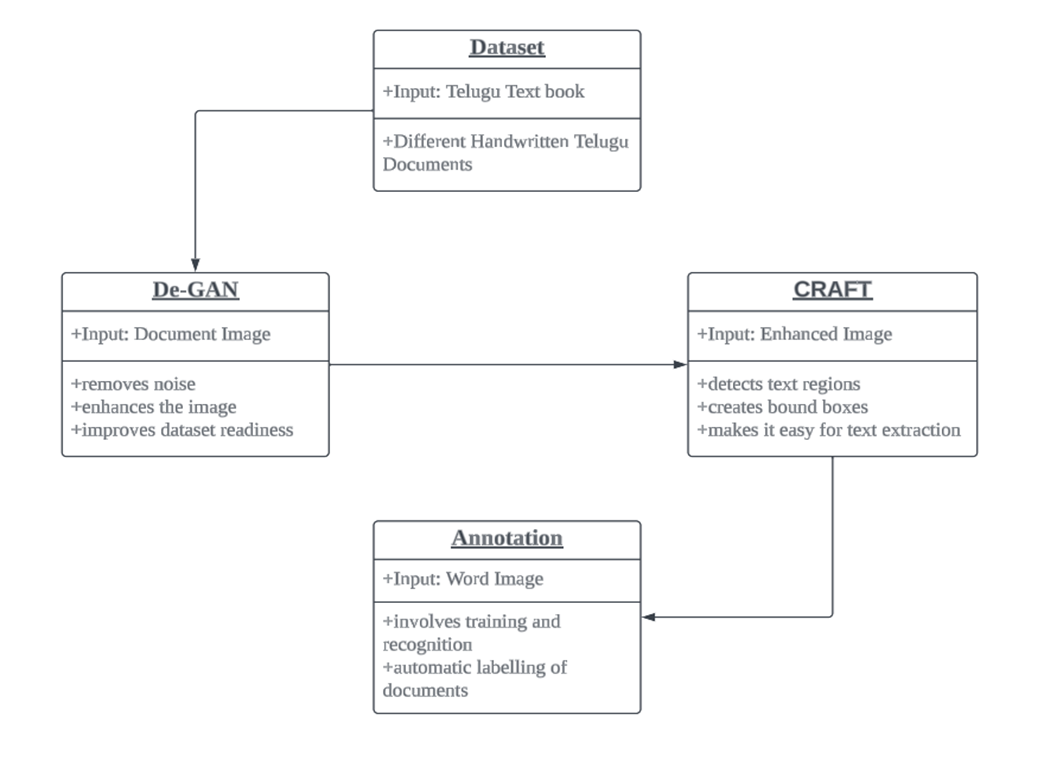


**Fig 7: Post-Ocr module architecture**

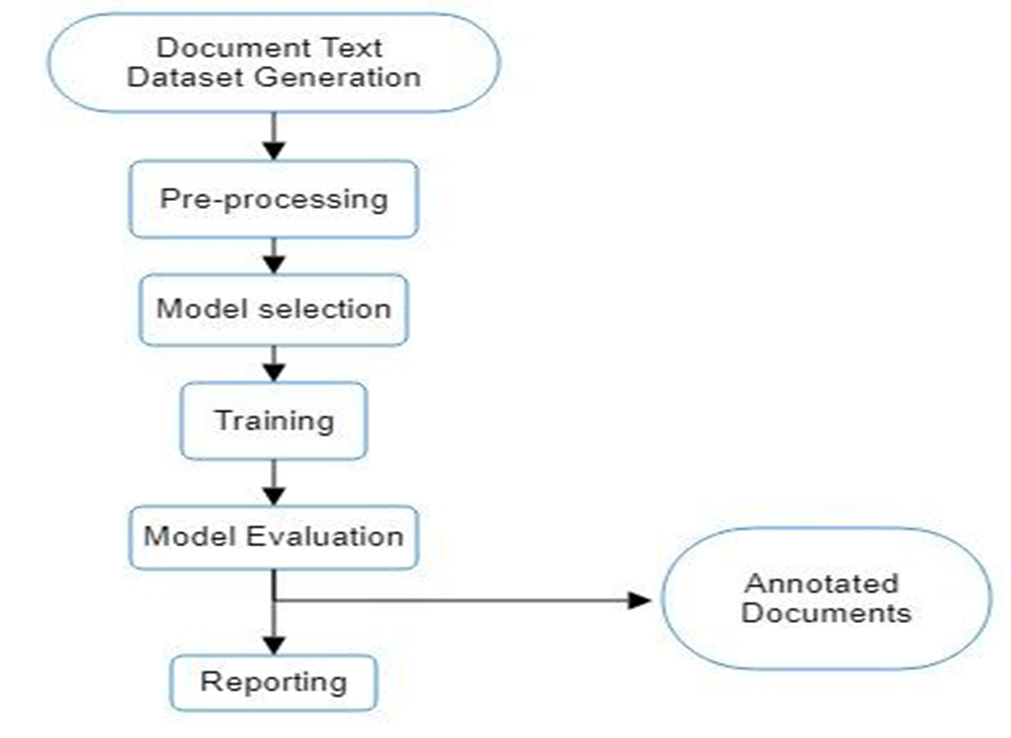
**4.2 UML Diagrams**

****

**Fig 8: Use Case Diagram**

****

**Fig 9: Class Diagram**

****

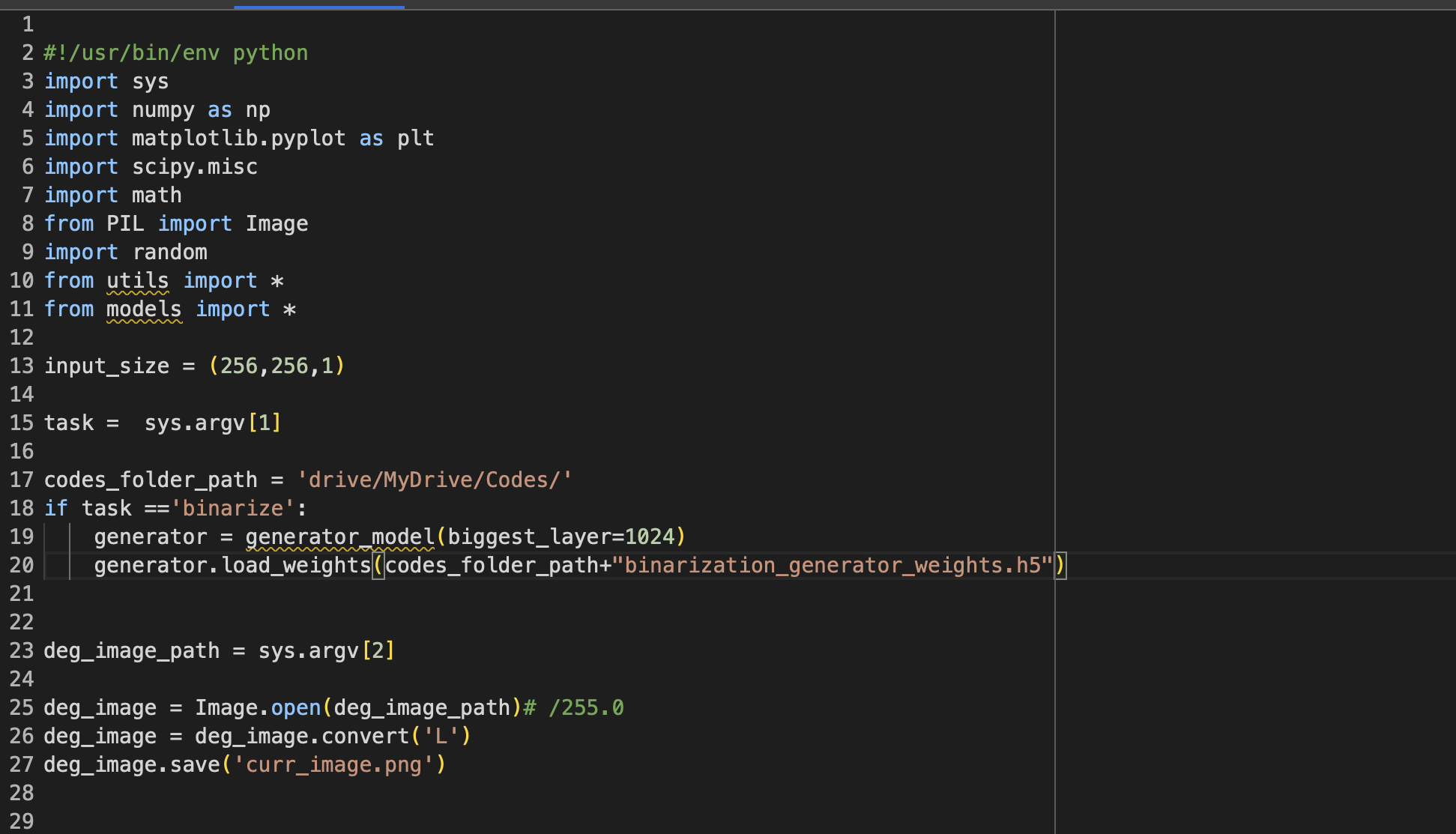
**Fig 10: Data Flow Diagram**

**Chapter 5**

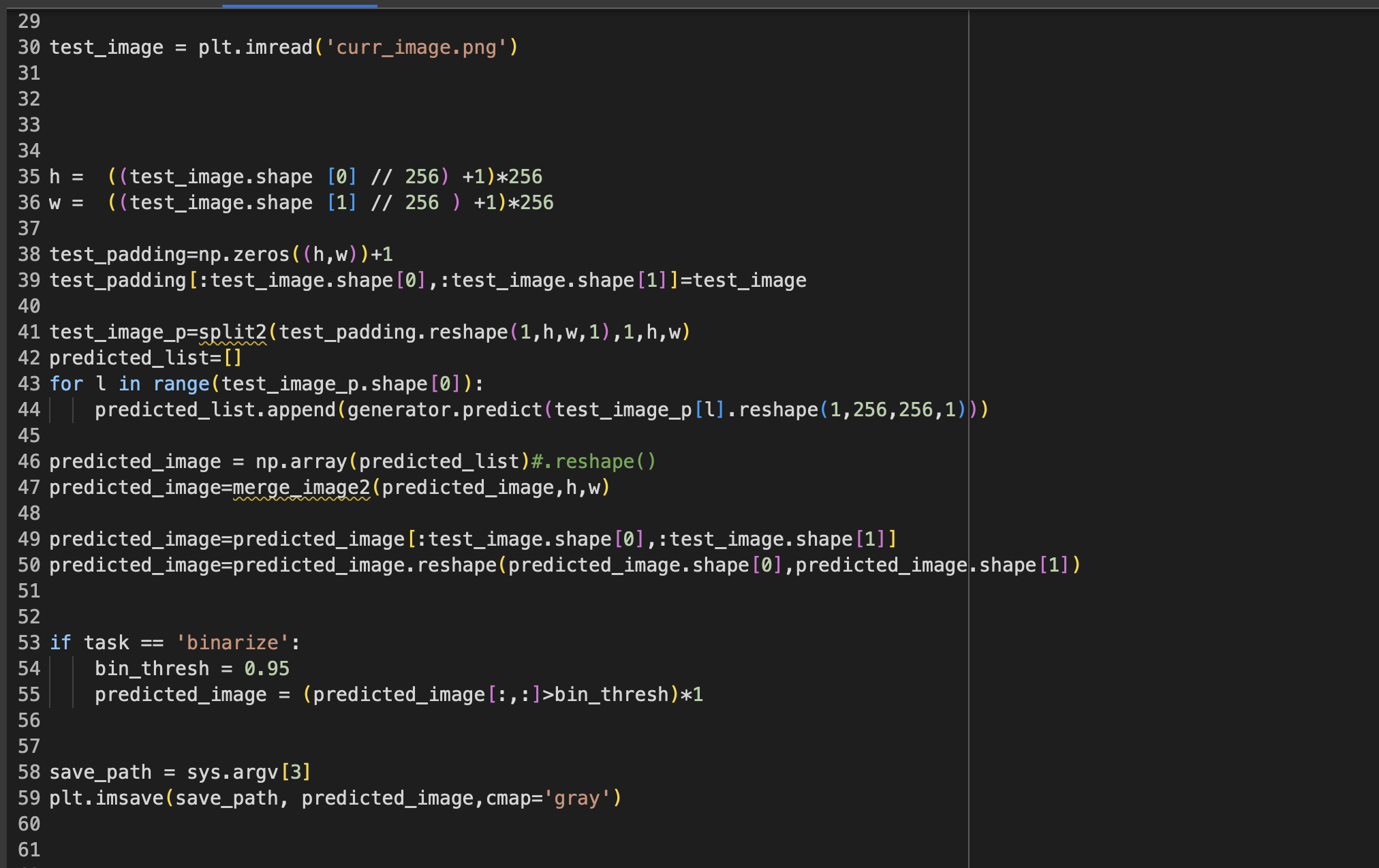
**Implementation steps:**

* The process begins with the acquisition of an image, ensuring it is taken without the influence of any filters.
* The unfiltered image is fed into the pretrained De-GAN (Denoising GAN) model.
* De-GAN specializes in denoising and enhances the image quality, generating a binarized version that eliminates background noises.
* The binarized image produced from De-GAN serves as input for the CRAFT model, which generates a bounding box image and a mask image highlighting detected text regions. Additionally, a text file is created, containing coordinates detailing the position of bounding boxes.
* After organizing bounding box images into classes, they're passed into a CRNN recognizer.
* The recognizer then generates electronic text representing the Telugu words extracted from the denoised images.
* The electronic text is then passed to the post-OCR to correct the predictions.
* Finally, this electronic text is annotated onto the denoised images, completing the workflow.

**De-GAN snapshots :**

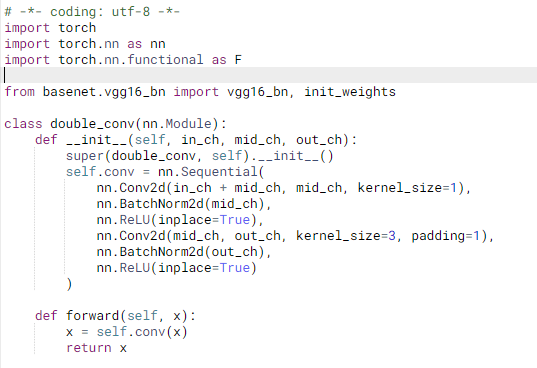


**Fig 11: De-GAN Code Module 1.**

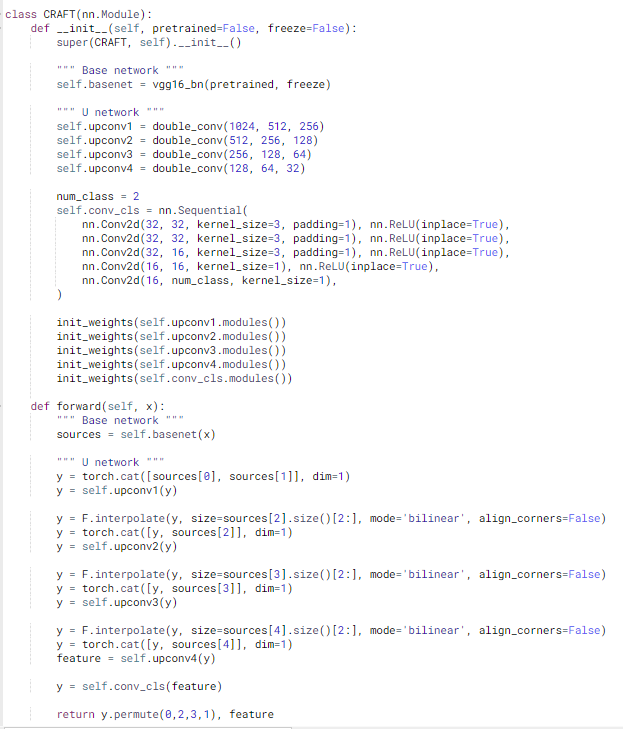


**Fig 12: De-GAN Code Module 2.**

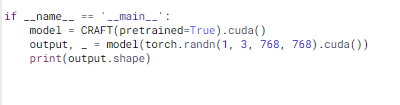
**Craft code snapshots:**



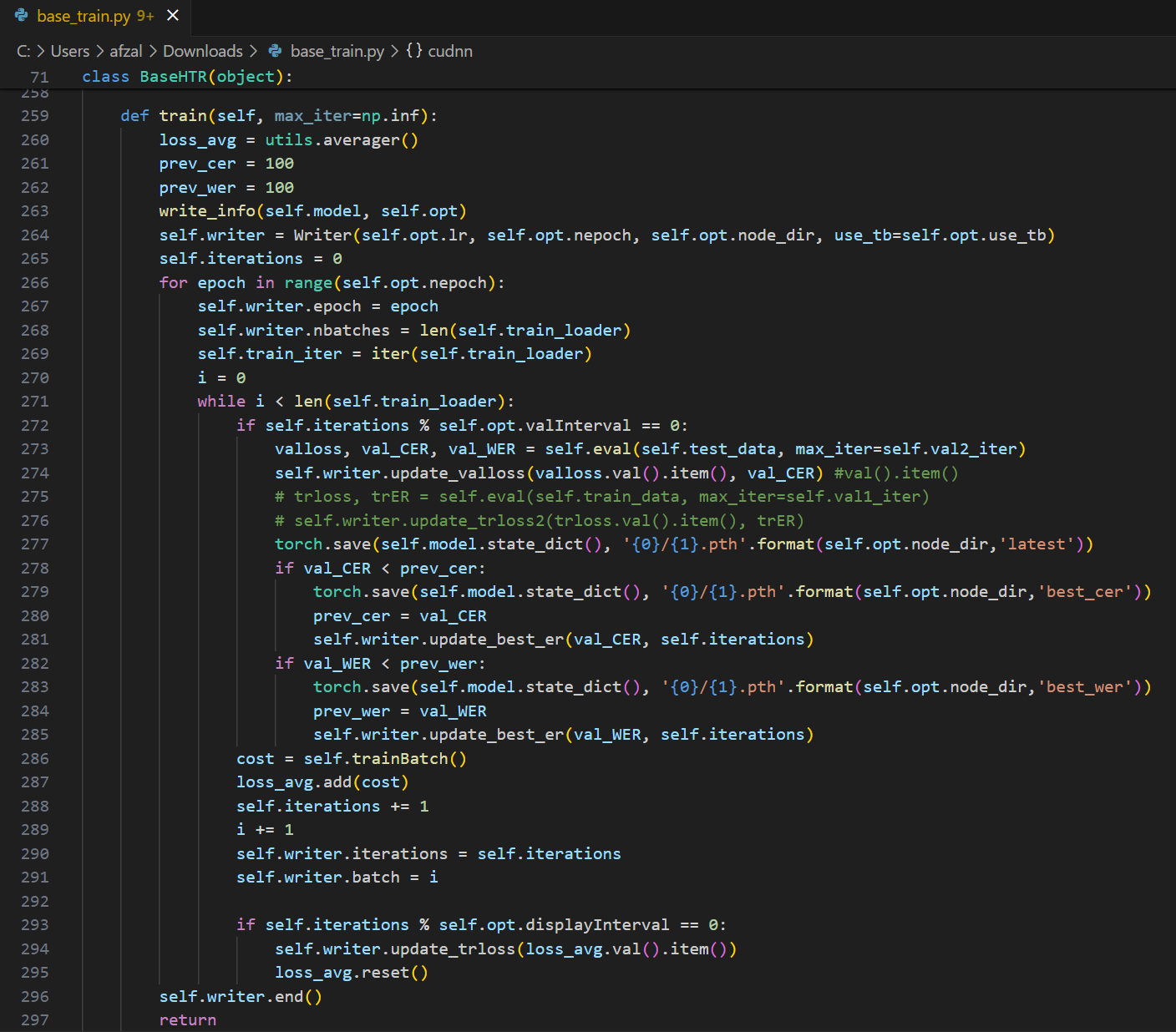
**Fig 13: CRAFT Code Module 1.**



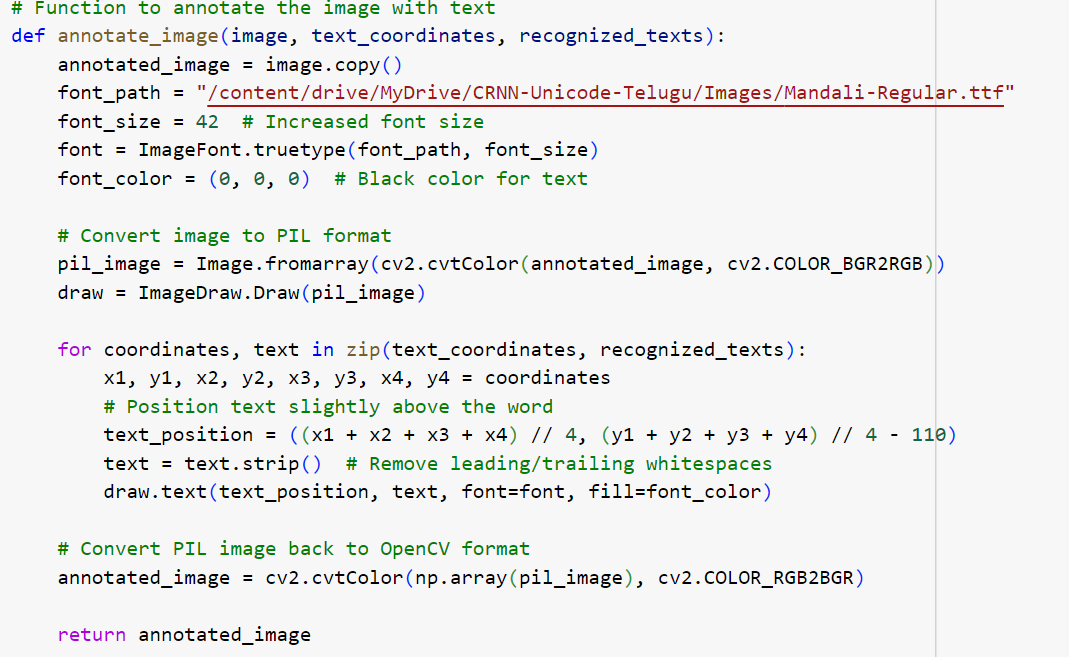
**Fig 14: CRAFT Code Module 2.**



**Fig 15: CRAFT Code Module 3.**



**Fig 16: Recognition Code Module**

****

**Fig 17: Annotation Code Module**

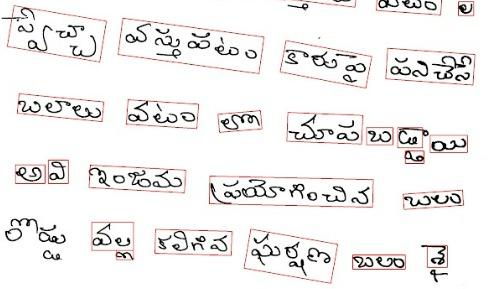
**Experimental Results**

We created our own Telugu Handwritten Text Dataset, extracted from 8th grade physics school textbook. Our friends and faculty of GRIET contributed in writing it. The dataset consists of a total 17,499 individual word images extracted from page-level images. These images are organized into 1,015 distinct classes, providing a comprehensive resource for training and evaluating OCR systems.

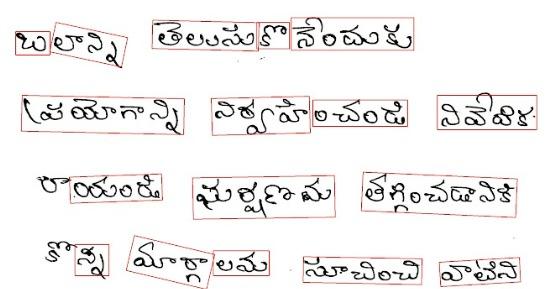
We partitioned the dataset into three subsets: 12,524 images for training, 3,485 images for testing, and 1,440 images for validation.

**Addressing Challenges in Text Recognition Using the CRAFT Model**

While utilizing the CRAFT model for text detection, several challenges can affect its performance. When the text is slanted, the model struggles to maintain accurate recognition, often failing to identify the word as a single coherent entity. This issue is particularly pronounced with longer words, which the model might fragment into multiple parts or overlook entirely. Such limitations can lead to incomplete or incorrect annotations, hindering the overall accuracy and effectiveness of the text recognition process.



**Fig 18 : Improper word detection by CRAFT**



**Fig 19 : Improper word detection by CRAFT**

We utilized CRNN architecture for the recognition phase, We trained and tested the model using our curated training, testing and validation datasets.

In the recognition phase, our testing results are as follows:

* Word Recognition Accuracy: 86.92%
* Character Recognition Accuracy: 97.37%

After the recognition phase, we implemented post-OCR[19] correction to further enhance the accuracy of our system. We performed training and testing for our own dataset. The integration of post-OCR correction not only boosts the precision of the OCR system but also enhances its robustness, making it more capable of handling real-world applications where accuracy is paramount.

For the post-OCR correction, we utilized recognized words from the OCR output and compared them with ground truth data. By analyzing the discrepancies between the recognized words and the ground truths, the algorithm adjusts and improves the OCR output. This process involved identifying common errors and patterns in the OCR results and applying specific corrections to mitigate these errors, thereby refining the overall accuracy and reliability of the system.

In the post-ocr phase, our testing results are as follows:

* VAL CER(Character Error Rate) : 0.0644
* VAL WER(Word Error Rate) : 0.2650

The recognition rates for Telugu handwritten document images were evaluated across three different datasets: 10,000 images, 5,000 images, and 17,000 images. The results are as follows:

**5,000 Images:**

* Word Recognition Accuracy: 78.29%
* Character Recognition Accuracy: 95.98%

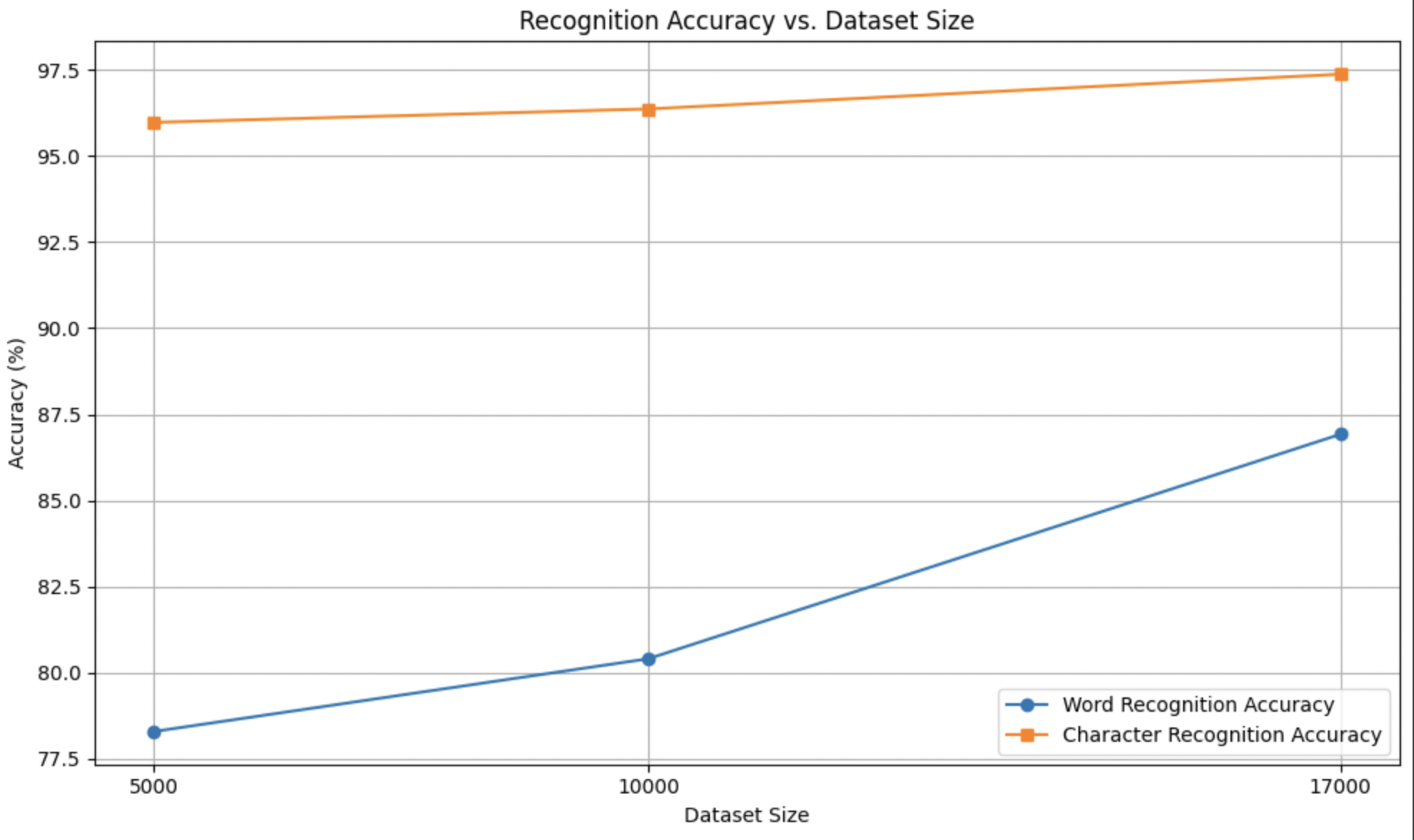
**10,000 Images:**

* Word Recognition Accuracy: 80.4%
* Character Recognition Accuracy: 96.37%

**17,000 Images:**

* Word Recognition Accuracy: 86.92%
* Character Recognition Accuracy: 97.37%

It has been observed from our dataset that increasing the size of the dataset significantly enhances the accuracy of machine learning models. Larger datasets provide a broader variety of examples, allowing models to learn more effectively and generalize better to new, unseen data. With more data, models can capture intricate patterns and nuances in the handwritten text, leading to improved performance and reliability in recognition tasks. Consequently, expanding the dataset is a crucial step in developing robust and accurate OCR systems.



**Fig 20: Results with different dataset sizes**

**Analysis**

From the data, several key observations can be made:

**1**. **Increasing Accuracy with More Data:**

There is a clear trend showing that as the number of images increases, both word and character recognition accuracies improve. This indicates that the model benefits from more training data, leading to better generalization and accuracy.

**2. Higher Character Recognition Accuracy:**

Across all datasets, character recognition accuracy consistently remains higher than word recognition accuracy. This suggests that recognizing individual characters is generally easier and more reliable than recognizing entire words, which is expected given the complexity and variability of word structures.

**3. Stability of Character Recognition:**

The character recognition accuracy shows less fluctuation compared to word recognition accuracy as the dataset size changes, indicating a more stable and robust character recognition performance.

**Summary of Percentage Changes:**

From 5,000 to 10,000 images: Approximately 2.69%

From 10,000 to 17,000 images: Approximately 8.11%

From 5,000 to 17,000 images: Approximately 11.02%

These calculations show that the word recognition accuracy improves as the dataset size increases, with the most significant improvement occurring between the datasets of 5,000 and 17,000 images. The higher character recognition accuracy highlights the need for potential enhancements in word recognition algorithms to bridge the gap between character and word-level accuracies.

**Test Cases**

**Table 1 : De-GAN Module Test Cases**

| **ID** | **Test Data** | **Output** | **Status(pass/Fail)** |
| --- | --- | --- | --- |
| **1** |  |  | **Pass** |
| **2** |  |  | **Pass** |

**Table 2 : CRAFT Module Test Cases**

| **S.No** | **Input** | **Output** | **Status(pass/Fail)** |
| --- | --- | --- | --- |
| **1** |  |  | **Pass** |
| **2** |  |  | **Pass** |
| **3** |  |  | **Fail** |

**Table 3 : Recognition Module Test Cases**

| **ID** | **Test Data** | **Output** | **Status(pass/Fail)** |
| --- | --- | --- | --- |
| **1** |  | ప్రవాహిలో | **Pass** |
| **2** |  | పెరుగుతుంది | **Pass** |
| **3** |  | ఉువ్తనాలు | **Fail** |
| **4** |  | క్రింది | **Pass** |

**Table 4 : Annotation Module Test Cases**

| **ID** | **Test Data** | **Output** | **Status(pass/Fail)** |
| --- | --- | --- | --- |
| **1** |  |  | **Pass** |
| **2** |  |  | **Pass** |

**Table 5 : Post-OCR Module Test Cases**

| **ID** | **Test Data** | **Output** | **Status(pass/Fail)** |
| --- | --- | --- | --- |
| **1** | Actual: 'ప్రభావాన్ని', Prediction: 'ప్రరావనాన్నని', | Prediction Corrected: 'ప్రభావాన్ని' | **pass** |
| **2** | Actual: 'వ్యతిరేకిస్తుందా', Prediction: 'వ్యరరశ్తుల--ో', | Prediction Corrected: 'వ్యరశ్తులో' | **fail** |

**Chapter 6**

**Conclusion and Future Scope**

In conclusion, our project represents a significant advancement in the realm of Automated Annotation Systems for document text images, offering transformative benefits across various industries. Our model gave an impressive Word Recognition Accuracy of 86.92% and Character Recognition Accuracy of 97.37%. By harnessing cutting-edge image processing techniques, including Optical Character Recognition (OCR) and deep learning algorithms such as Generative Adversarial Networks (GANs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks etc., our system significantly improves document analysis, data extraction, and information retrieval processes. Notably, our project includes a meticulously curated Telugu dataset comprising 17,449 word-level images derived from 437 handwritten pages.

Through automation, it streamlines the labor-intensive task of manual annotation, thereby optimizing efficiency, accuracy, and productivity. With applications ranging from digitizing historical archives and cataloging extensive document databases to facilitating content indexing and semantic understanding, our system stands at the forefront of technological innovation. As we continue to push the boundaries of technology, the development and adoption of Automated Annotation Systems for Document Text Images hold the potential to revolutionize the way we interact with and derive insights from textual documents, paving the way for a more efficient and insightful future.

**Acknowledgement**

We thank the Department of Science and Technology(DST), Government of India for providing financial support for our project. We also would like to thank our friends and faculty of GRIET, who contributed by writing our Telugu dataset from 8th-grade Physics school textbook.

**Publications**

1. A. D. R. N, B. Srija, M. Jabeen, K. R. Kankar, B. J. Lakshmi and Y. Vijayalata, "Synthetic Data Generation for Document Text Recognition," 2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU), Bhubaneswar, India, 2024, pp. 1-6, doi: 10.1109/IC-CGU58078.2024.10530719.
2. A. D. R. N, B. Srija, M. Jabeen, K. R. Kankar, B. J. Lakshmi and A. Negi, "A Review on Automated Annotation System for Document Text Images," 2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU), Bhubaneswar, India, 2024, pp. 1-6, doi: 10.1109/IC-CGU58078.2024.10530793.

**Chapter 7**

**References**

1. Ajoy Mondal, Krishna Tulsyan, and C.V. Jawahar. "Automatic Annotation of Handwritten Document Images at Word Level." ICVGIP '22, ACM, 2023.

2. Qiang Hu, Qi Liu, Xiaoli Wang, Anthony K.H. Tung, Shubham Goyal, and Jisong Yang. "DocRicher: An Automatic Annotation System for Text Documents Using Social Media." SIGMOD '15, ACM, 2015.

3. Choi, D., Kim, P. "Automatic Image Annotation Using Semantic Text Analysis." CD-ARES 2012, Springer, 2012.

4. Balci, Batuhan, Dan Saadati, and Dan Shiferaw. "Handwritten text recognition using deep learning." CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University, Course Project Report, Spring (2017): 752-759.

5. Mondal, Ajoy & Tulsyan, Krishna & Jawahar, C.. (2022). Automatic Annotation of Handwritten Documents in Word Level. 10.1145/3571600.3571645.

6. Jain, Mohit & Mathew, Minesh & Jawahar, C.V.. (2017). Unconstrained OCR for Urdu Using Deep CNN-RNN Hybrid Networks. 747-752. 10.1109/ACPR.2017.5.

7. Gunna, Sanjana, Rohit Saluja, and C. V. Jawahar. "Transfer learning for scene text recognition in Indian languages." International Conference on Document Analysis and Recognition. Cham: Springer International Publishing, 2021.

8. Coquenet, Denis & Chatelain, Clement & Paquet, Thierry. (2020). End-to-end Handwritten Paragraph Text Recognition Using a Vertical Attention Network.

9. Souibgui, M. A., Fornés, A., Kessentini, Y., and Megyesi, B., “Few shots are all you need: A progressive learning approach for low resource handwritten text recognition”, <i>Pattern Recognition Letters</i>, vol. 160, pp. 43–49, 2022. doi:10.1016/j.patrec.2022.06.003.

10. Sankar, K. Pramod and C. V. Jawahar. “Enabling Search over Large Collections of Telugu Document Images - An Automatic Annotation Based Approach.” Indian Conference on Computer Vision, Graphics & Image Processing (2006).

13. Bansal, Siddhant, Praveen Krishnan, and C. V. Jawahar. "Fused Text Recogniser and Deep Embeddings Improve Word Recognition and Retrieval." International Workshop on Document Analysis Systems. Cham: Springer International Publishing, 2020.

14. Vidal, Enrique and Toselli, Alejandro Héctor and Rios-Vila, Antonio and Calvo-Zaragoza, Jorge, End-to-End Page-Level Assessment Of Handwritten Text Recognition. Available at SSRN: https://ssrn.com/abstract=4347108 or<http://dx.doi.org/10.2139/ssrn.4347108>

15. B M, Sagar & Shobha, G. & P., Ramakanth. (2008). OCR for printed Kannada text to machine editable format using database approach. WSEAS Transactions on Computers. 7. 766-769.

16. Sharma, Parikshit. (2023). Advancements in OCR: A Deep Learning Algorithm for Enhanced Text Recognition. International Journal of Inventive Engineering and Sciences. 10. 1-7. 10.35940/ijies.F4263.0810823.

17. Ramteke, Surendra & Gurjar, Ajay & deshmukh, Dhiraj. (2018). A streamlined OCR system for handwritten MARATHI text document classification and recognition using SVM-ACS algorithm. International Journal of Intelligent Engineering and Systems. 11. 186-195. 10.22266/ijies2018.0630.20.

18. Kumar, K. Vijay, and R. Rajeshwara Rao. "Online handwritten character recognition for Telugu language using support vector machines." *International Journal of Engineering and Advanced Technology* 3.2 (2013): 189-192.

19. Rijhwani, Shruti, Antonios Anastasopoulos, and Graham Neubig. "OCR post correction for endangered language texts." *arXiv preprint arXiv:2011.05402* (2020).DE-GAN

20. Rigaud, Christophe & Doucet, Antoine & Coustaty, Mickaël & Moreux, Jean-Philippe. (2019). ICDAR 2019 Competition on Post-OCR Text Correction. 1588-1593. 10.1109/ICDAR.2019.00255.

21. Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. 2021. Survey of Post-OCR Processing Approaches. ACM Comput. Surv. 54, 6, Article 124 (July 2022), 37 pages. <https://doi.org/10.1145/3453476>

22. Sarah Schulz and Jonas Kuhn. 2017. [Multi-modular domain-tailored OCR post-correction](https://aclanthology.org/D17-1288). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2726, Copenhagen, Denmark. Association for Computational Linguistics.

23. Amrith Krishna, Bodhisattwa P. Majumder, Rajesh Bhat, and Pawan Goyal. 2018. [Upcycle Your OCR: Reusing OCRs for Post-OCR Text Correction in Romanised Sanskrit](https://aclanthology.org/K18-1034). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 345–355, Brussels, Belgium. Association for Computational Linguistics.

24. Okan Kolak and Philip Resnik. 2005. [OCR Post-Processing for Low Density Languages](https://aclanthology.org/H05-1109). In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 867–874, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

25. Mika Hämäläinen and Simon Hengchen. 2019. [From the Paft to the Fiiture: a Fully Automatic NMT and Word Embeddings Method for OCR Post-Correction](https://aclanthology.org/R19-1051). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 431–436, Varna, Bulgaria. INCOMA Ltd..

26. Souibgui, M. A. and Kessentini, Y., “DE-GAN: A Conditional Generative Adversarial Network for Document Enhancement”, <i>arXiv e-prints</i>, 2020. doi:10.48550/arXiv.2010.08764.

27. Y. Baek, B. Lee, D. Han, S. Yun and H. Lee, "Character Region Awareness for Text Detection," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 9357-9366, doi: 10.1109/CVPR.2019.00959.

28. Ganji, T., Sekhar Velpuru, M., and Dugyala, R., “Multi Variant Handwritten Telugu Character Recognition Using Transfer Learning”, in <i>Materials Science and Engineering Conference Series</i>, 2021, vol. 1042, no. 1, p. 012026. doi:10.1088/1757-899X/1042/1/012026.

29. Bv, Dhandra, Gururaj Mukarambi, and Mallikarjun Hangarge. "A script independent approach for handwritten bilingual Kannada and Telugu digits recognition." International Journal of Machine Intelligence 3 (2011): 155-159.

30. Richard W Howell. 1951. The classification and de- scription of ainu folklore. *The Journal of American Folklore*, 64(254):361–369.

31. Rajbongshi, Aditya, et al. "Bangla optical character recognition and text-to-speech conversion using raspberry Pi." *International Journal of Advanced Computer Science and Applications* 11.6 (2020).

32. Shekar, K. Chandra, Maria Anisha Cross, and Vignesh Vasudevan. "Optical character recognition and neural machine translation using deep learning techniques." *Innovations in Computer Science and Engineering: Proceedings of 8th ICICSE*. Springer Singapore, 2021.

33. Kundaikar, Teja, and Jyoti D. Pawar. "Multi-font Devanagari text recognition using LSTM neural networks." *First International Conference on Sustainable Technologies for Computational Intelligence: Proceedings of ICTSCI 2019*. Springer Singapore, 2020.

34. Islam, Md Majedul, et al. "Towards building a bangla text recognition solution with a multi-headed cnn architecture." *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021.

35. Jayanthi, V., and S. Thenmalar. "Tamil OCR Conversion from Digital Writing Pad Recognition Accuracy Improves through Modified Deep Learning Architectures." *Journal of Sensors* 2023 (2023).

36. Dhanikonda, Srinivasa Rao, et al. "An efficient deep learning model with interrelated tagging prototype with segmentation for telugu optical character recognition." *Scientific Programming* 2022 (2022).

37. Kashyap, Abhishek. "OCR of Kannada Characters Using Deep Learning." *2022 Trends in Electrical, Electronics, Computer Engineering Conference (TEECCON)*. IEEE, 2022.

38. Pareek, Jyoti, et al. "Gujarati handwritten character recognition from text images." *Procedia Computer Science* 171 (2020): 514-523.

39. Khobragade, Ratnashil N., Nitin A. Koli, and Vrushali T. Lanjewar. "Challenges in recognition of online and off-line compound handwritten characters: a review." Smart Trends in Computing and Communications: Proceedings of SmartCom 2019 (2020): 375-383.

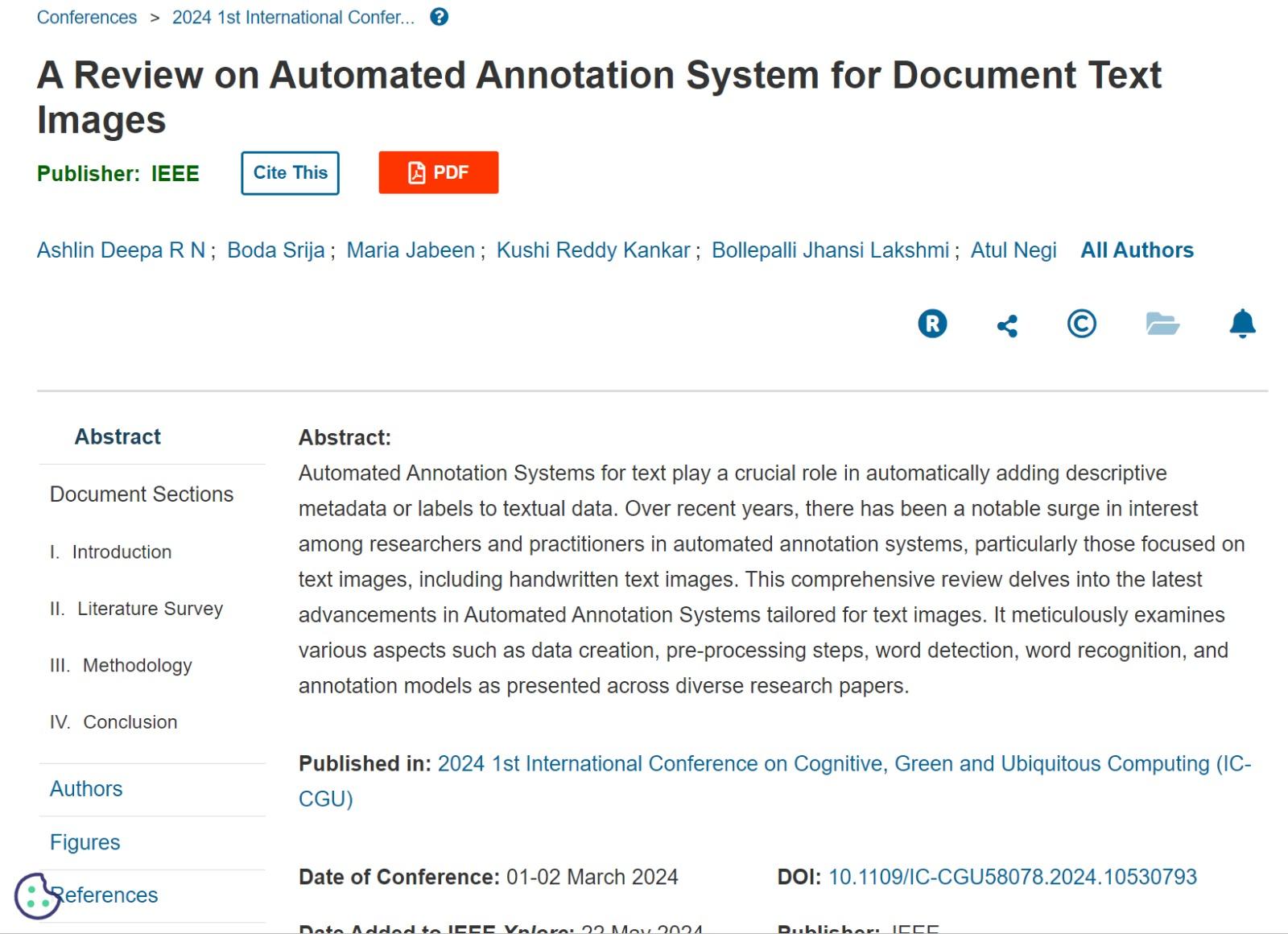
40 Ignat, Oana, et al. "OCR improves machine translation for low-resource languages." *arXiv preprint arXiv:2202.13274* (2022).

41. Huang, Jing, et al. "A multiplexed network for end-to-end, multilingual OCR." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

42. Deepa, Ashlin, et al. "Synthetic Data Generation for Document Text Recognition." *2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU)*. IEEE, 2024.

**Appendix**

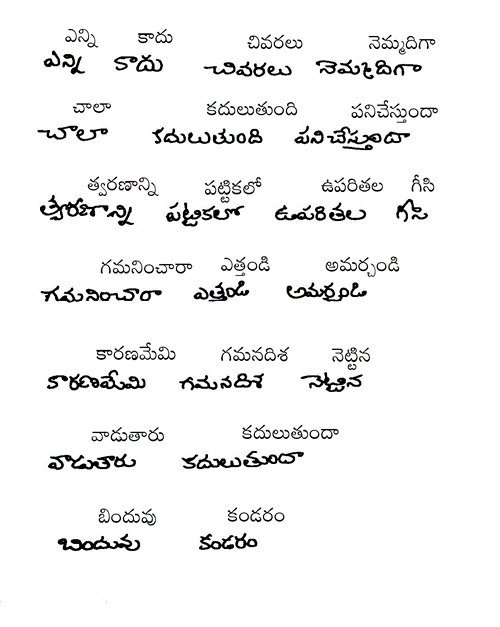
**Full Paper Publication Proof**

****

**Fig 21: Paper Acceptance for Review paper**

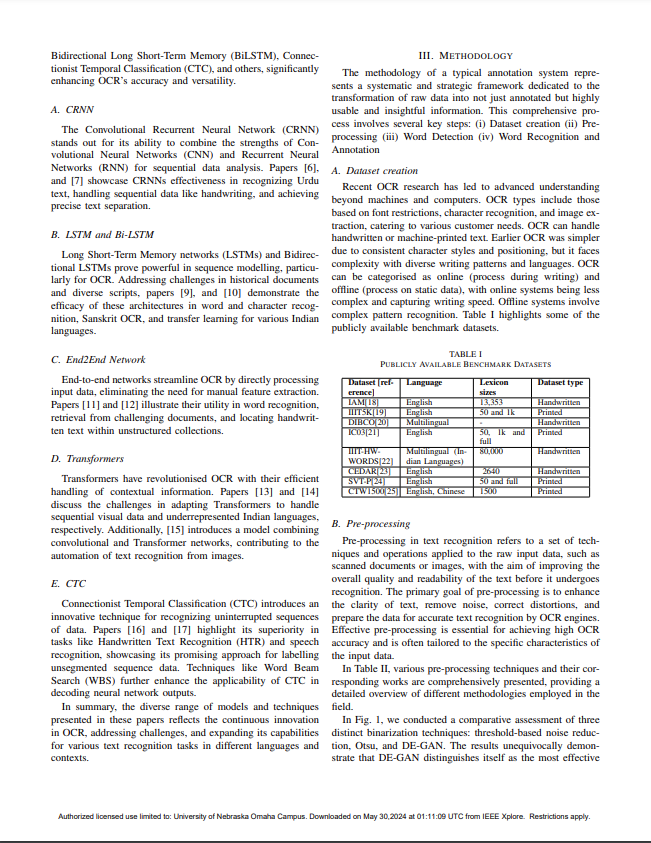
**Paper Link :** [**https://ieeexplore.ieee.org/document/10530793**](https://ieeexplore.ieee.org/document/10530793)

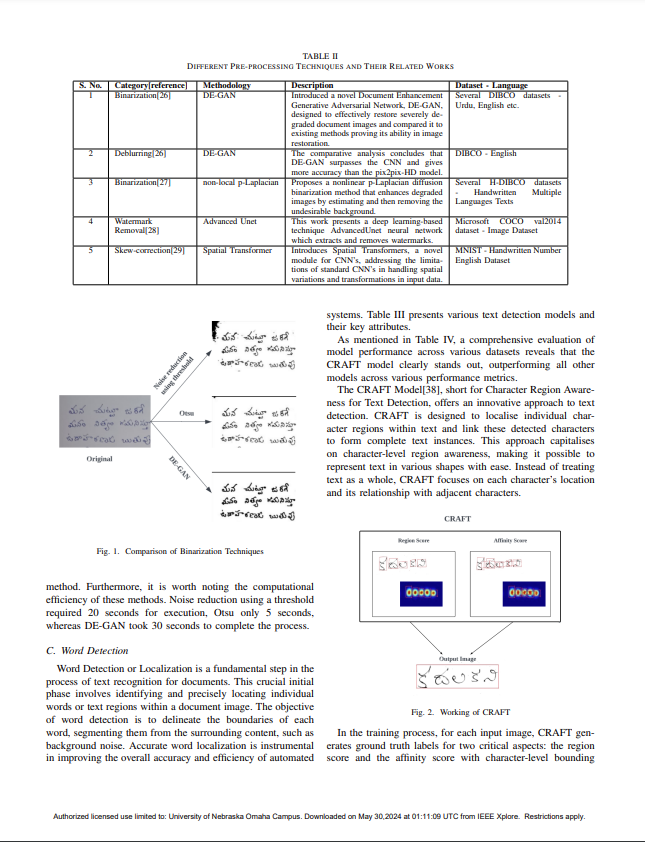
**Snapshot Of the Result**

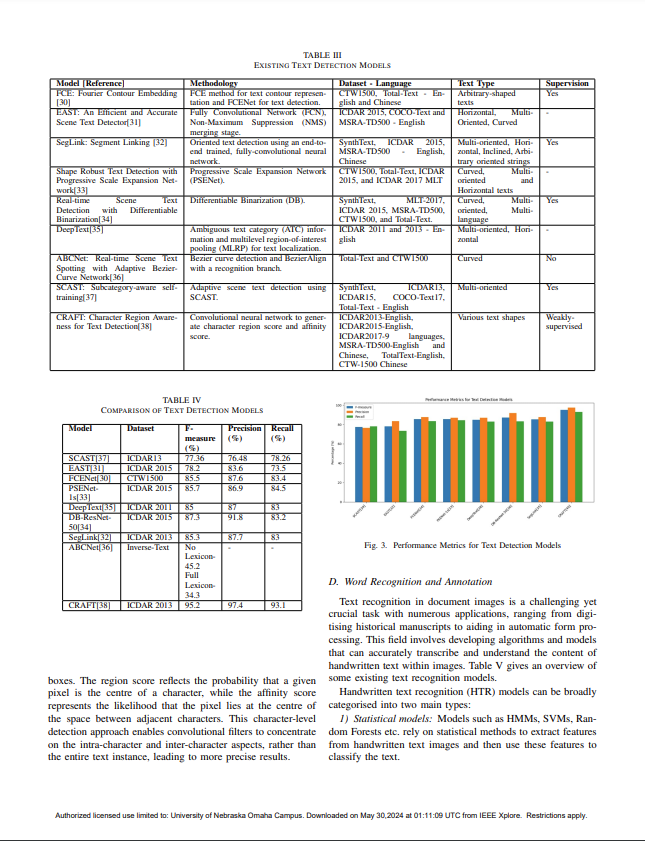
****

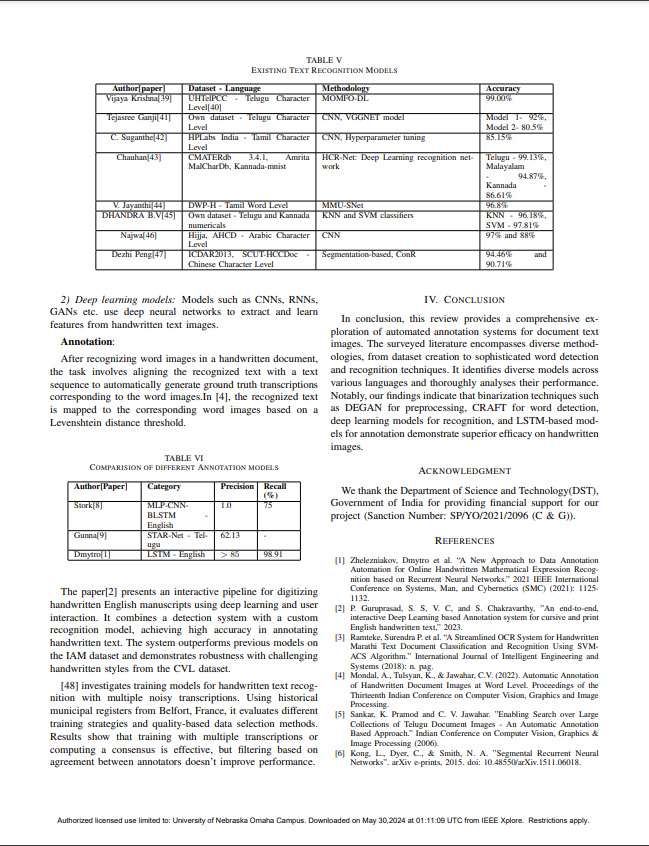
**Fig 22: Result**

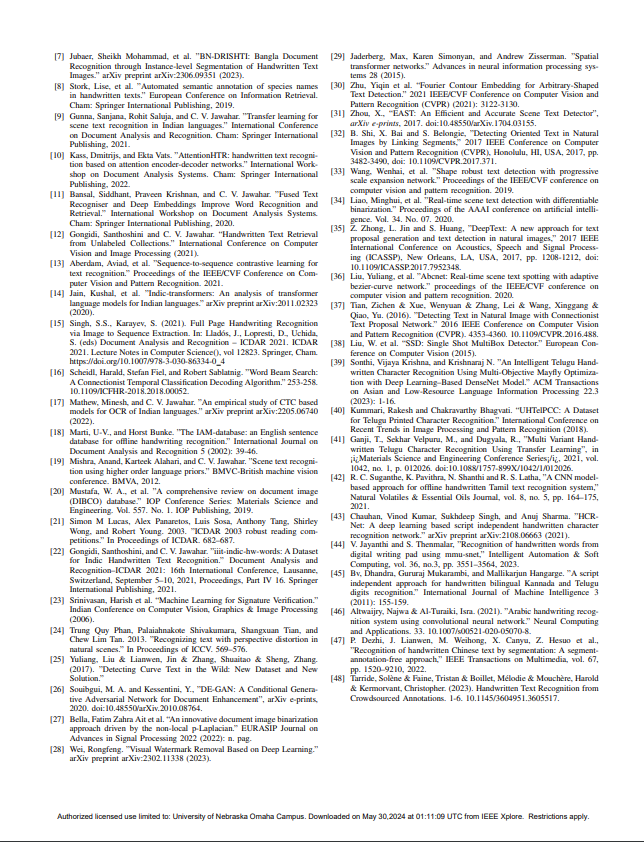
**Published paper:  
**

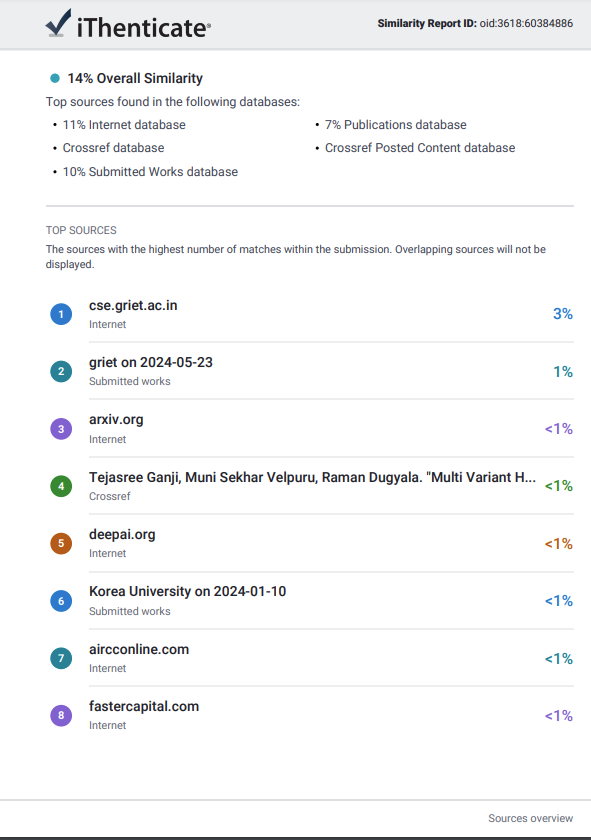
****

****

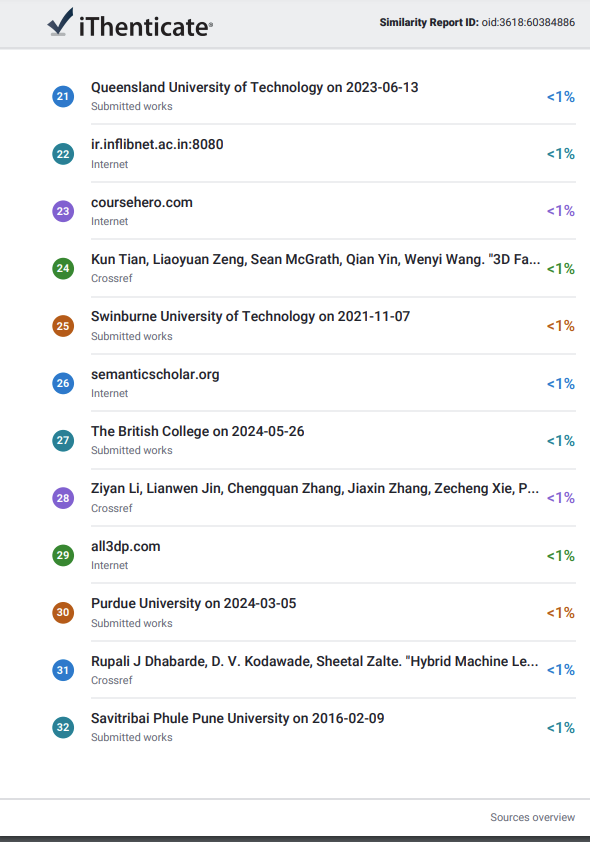
****

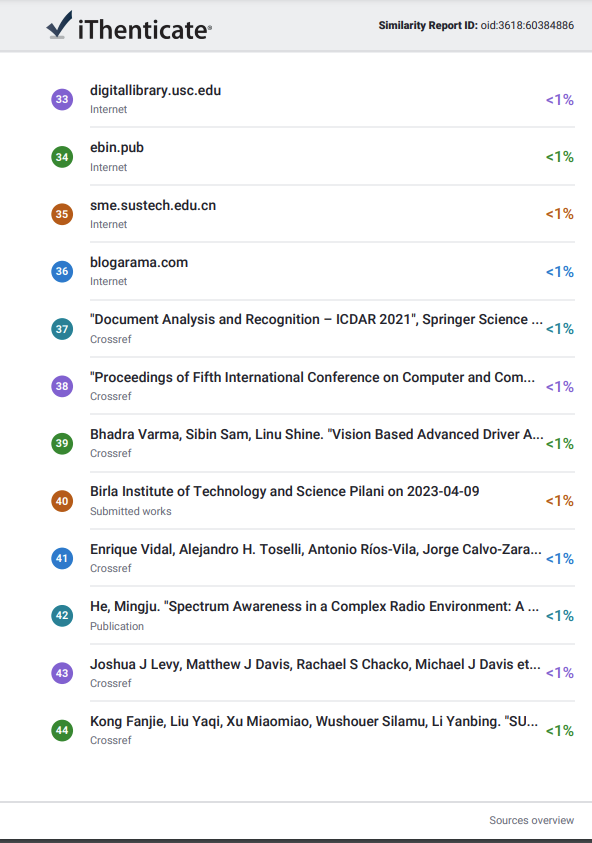
****

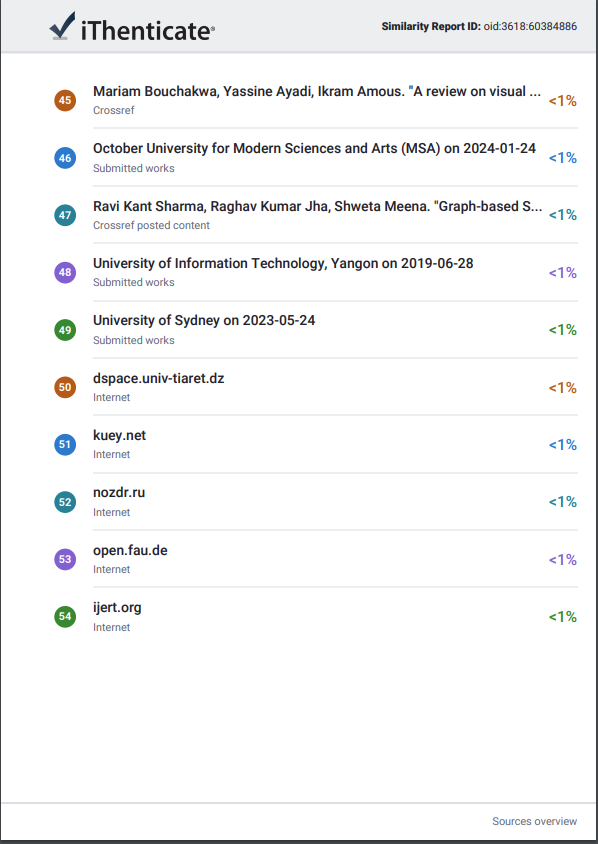
****

**Plagiarism report:  
**

****

****

****

****