Impact Factor (SJIF): 6.092 E-ISSN: 2583-1615

International Journal of Research in Academic World

Received: 09/May/2024 IJRAW: 2024; 3(6):67-75 Accepted: 11/June/2024

Automated Form Filling Using OCR and Machine Learning for Enhanced Data Accuracy and Efficiency

[*1]Greshma P Sebastian and [2]Thasneem Musthafa

[*1]Assistant Professor, Department of Computer Science and Engineering, SCMS School of Engineering and Technology, APJ Abdul Kalam Technical University, Kerala, India.

[2]Student, Department of Computer Science and Engineering, SCMS School of Engineering and Technology, APJ Abdul Kalam Technical University, Kerala, India.

Abstract

The system starts with optical character recognition (OCR) picture processing and text extraction, with an emphasis on improving image quality and obtaining pertinent textual data. Then, to deal with missing or noisy text input and convert it into an organized format appropriate for model training, a data preparation step is utilized. The construction and training of a machine learning model especially made for filling out forms using the preprocessed text input forms the basis of the system. This entails investigating different machine learning algorithms, maybe involving classification models and Natural Language Processing (NLP) methods. The trained model is integrated by the form filling engine, which then transfers the structured text data to the appropriate form fields. Among the advantages of the system are its ability to streamline the form-filling process and reduce human data entering. This helps in improving precision through language interpretation powered by machine learning. Adapting to changes in form layouts and making sure that the text and picture formats are robust are challenges.

Keywords: Tesseract, OCR, YOLO, OpenCV

1. Introduction

1.1. General Background

In recent years, advancements in machine learning and computer vision technologies have paved the way for innovative solutions in automating labor-intensive tasks. Among these, the extraction and utilization of information from images have gained prominence. Image-based text extraction, facilitated by Optical Character Recognition (OCR) [1] tools like Tesseract [2], has become a valuable tool in transforming unstructured visual data into machine-readable text. This capability has found applications in various domains, including document analysis, data entry, and information retrieval.

The integration of machine learning models into these processes has further enhanced their adaptability and efficiency. Specifically, in the context of form filling, the combination of image processing, text extraction, and machine learning enables the development of systems capable of automating the completion of forms by intelligently interpreting and mapping textual content from images. This technology not only reduces the burden of manual data entry but also opens up possibilities for increased accuracy and scalability across diverse form structures and layouts.

As these technologies continue to evolve, the potential for further advancements in image-based information extraction remains a dynamic area of research and development.

1.2. Objective

The objective of implementing the proposed machine learning-based form-filling system is to automate and streamline the process of completing forms by leveraging text extracted from input images. The primary goals include reducing manual data entry efforts, enhancing accuracy in interpreting and mapping textual content, and providing adaptability to diverse form layouts and structures. By integrating image processing, OCR, and machine learning techniques, the system aims to optimize the efficiency of form-filling tasks, making them more time-efficient and less prone to errors. This project seeks to contribute to the advancement of automation technologies, particularly in the domain of document processing, and aims to create a robust and scalable solution for handling various types of forms with minimal human intervention. The long-term objective is to establish a versatile and intelligent system that can adapt to evolving requirements and contribute to increased productivity.

***Corresponding Author:** Greshma P Sebastian

1.3. Scope

The scope of this project is to develop a robust machine learning-based form-filling system that automates the extraction of textual information from input images and intelligently populates corresponding form fields. The project encompasses the integration of image processing techniques and Optical Character Recognition (OCR) tools, with a focus on enhancing image quality and extracting accurate text data. Data preprocessing steps will be implemented to handle missing or noisy text data, transforming it into a structured format suitable for machine learning tasks. The project involves the design and training of a machine learning model, potentially leveraging Natural Language Processing (NLP) [3] techniques, to comprehend and map extracted text to specific form fields. A form-filling engine will be developed, ensuring adaptability to various form layouts and structures. Performance evaluation will assess the accuracy and efficiency of the system, guiding further refinements. Optionally, a user-friendly interface may be implemented for user interaction. The project's scope also includes the identification of opportunities for future enhancements, such as integrating deep learning techniques and extending the system's capabilities to handle dynamic forms in real-time. Overall, the project aims to create an effective, adaptable, and efficient solution for automating form-filling tasks based on image-derived textual data.

1.4. Organization of Report

There are five chapters in the report. The first report deals with the general background, objective, and scope of the project. The second chapter contains various literature reviews related to the project. The detailed architecture and working of the proposed system are discussed in the third chapter. The fourth chapter contains the experimental results and discussions. The conclusions are summarized in the final chapter. The future scope of the given project is also added in the last chapter. Finally, the references are given on the last pages.

2. Literature Survey

This chapter discusses some recent research papers on methods for Image to text conversion and pre-processing for OCR.

The paper “Text detection and localization in scene images: a broad review” [4] proposed approach of employing an unsupervised method leveraging wavelet transform for the automatic processing of text within intricate images. The wavelet transform, a mathematical technique enabling the decomposition of images into various frequency components, is harnessed to adeptly identify regions likely to contain text. Significantly, the method operates without the need for labeled training data, autonomously analyzing images to detect potential text areas through the distinctive features revealed by the wavelet transform. Following text detection, the subsequent stages involve text segmentation and binarization. Text segmentation separates the detected text regions from the rest of the image, while binarization converts these segmented regions into a binary format, typically black and white pixels.

In “Optical character recognition by open source OCR tool tesseract: A case study” [5] Patel, Chirag, Atul Patel, and Dharmendra Patel have conduct a study to compare between tesseract OCR and Transym OCR. The comparative analysis encompasses factors such as accuracy, ease of use, language support, customization, cost implications, and application-specific requirements. The choice between Tesseract and Transym OCR depends on the user's specific needs, budget constraints, and the unique characteristics of the OCR application at hand.

“Improving OCR performance with background image elimination” [6] forwards the idea of preprocessing image for OCR. In preprocessing stage document images are enhanced by leveraging the parameters of brightness and chromaticity to improve contrast. Brightness refers to the overall intensity of the image, and chromaticity relates to the color information. By considering these parameters, the contrast between text and background can be enhanced, leading to improved OCR accuracy. Additionally, color images are converted to grayscale during this process to simplify the analysis and reduce computational complexity. Grayscale conversion ensures that only intensity information is retained, eliminating the complexities associated with color variations.

The paper “Automated text extraction from images using OCR system” [7] aims to involves the conversion of colored images into a binary format, where each pixel is assigned either a black or white value based on a predefined threshold. Binarization simplifies the subsequent character recognition task by reducing the complexity of image data and emphasizing the contrast between text and background. Once the image is successfully binarized, character recognition is applied to interpret the binary image and convert it into ASCII text. Character recognition algorithms analyze patterns within the binary image, identifying distinct shapes and configurations that represent individual characters. The result is an ASCII text representation of the content within the image, providing a structured and machine-readable output that can be further processed or analyzed.

The “A different image content-based retrievals using OCR techniques” [8] proposes is employed to extract textual information from images or messages, allowing for the identification and retrieval of specific content. OCR algorithms analyze the visual patterns of characters, recognizing and translating them into text that can be processed by computers. Once the text is successfully recognized, the OCR system stores the extracted message in a designated file. This file could be a text document, spreadsheet, or any other format suitable for preserving the recognized text. By integrating OCR into image search processes, organizations and individuals can efficiently extract and store textual information from images, enabling better organization, searchability, and utilization of content contained within visual data.

The paper “OCR post-correction for detecting adversarial text images” [9] describes about the post correction methods needed for OCR. In scenarios where the quality of images is compromised due to distortions, noise, or other forms of perturbations, OCR systems may exhibit inaccuracies in text recognition. The post-correction algorithm acts as a remedial measure, systematically analyzing the OCR output and correcting errors or inaccuracies in the recognized text. It employs sophisticated techniques to identify and rectify misinterpreted characters, ensuring a more accurate and reliable transcription of embedded texts within perturbed images.

Lat, Ankit, and C. V. Jawahar proposed an unsupervised deep learning method for text summarization in “Enhancing OCR accuracy with super resolution” [10]. The utility of a system that validates the wide variety of document images without the need for pre-processing steps is characterized by its adaptability to diverse situations where fonts, styles, and

languages may vary. This capability is particularly advantageous in handling a broad spectrum of document types without requiring tailored pre-processing for each specific scenario. In essence, the system demonstrates versatility in recognizing and extracting information from documents with varying fonts, styles, and languages, eliminating the necessity for manual adjustments or preprocessing steps to accommodate different situations.

The paper "Selecting automatically pre-processing methods to improve OCR performances" [11] involves various document pre-processing methods, including noise reduction, image enhancement, and geometric correction, are applied to these distorted images. Multiple OCR engines are then employed to recognize text from both the pre-processed and original images. The evaluation includes metrics such as accuracy, precision, and recall to quantify the impact of different pre-processing techniques on OCR performance. By systematically varying the types and degrees of distortions, researchers can gain insights into the effectiveness of pre-processing methods in enhancing OCR engine performance across a spectrum of challenging document image.

The framework proposed by "Document image analysis using ImageMagick and tesseract-ocr" [12] uses combination of both ImageMagick and Tesseract-OCR. ImageMagick provides a powerful set of tools for image processing, enabling operations such as resizing, cropping, and color adjustments. Integrating ImageMagick with Tesseract OCR, an optical character recognition engine, creates a comprehensive solution for document analysis.

In "Ocr using computer vision and machine learning" [13] introduces an efficient and highly scalable parallel architecture designed for the segmentation of input images that encompass tabular data, both with and without borders. The primary objective is to dissect these images into individual cells while maintaining the inherent tabular structure. Through a parallelized approach, the architecture can process large volumes of data swiftly, making it highly scalable and efficient. This parallel architecture not only facilitates the segmentation of tabular content but also focuses on reconstructing the tabular data while preserving its original format. The notable performance improvement achieved by this methodology serves to alleviate the laborious task of digitizing tabular data in bulk.

Proposed System

3.1. Introduction: The proposed system is designed to streamline the extraction of text from images, facilitate the conversion of this textual information into a structured CSV file, and subsequently employ the CSV data for automating form filling processes. Leveraging advanced Optical Character Recognition (OCR) technology, the system ensures accurate extraction of text from images, accommodating diverse fonts, styles, and languages.

The extracted text is organized into a CSV format, considering variations in data presentation, and stored securely for future use. A user-friendly interface allows users to upload images, trigger text extraction, and manage the generated CSV file. The system further includes a module for automating the population of forms with CSV data, enhancing efficiency in data entry tasks. Robust error handling mechanisms and security measures are integrated to address OCR inaccuracies, CSV generation errors, and form autofill failures, ensuring reliability and data integrity. The proposed system thus presents a comprehensive solution to expedite the process of extracting text from images and utilizing the information for seamless form filling, contributing to increased productivity and reduced manual data entry efforts.

At the heart of Pixel Pinnacle lies its integration of state-of-the-art OCR algorithms, with a particular emphasis on deep learning-based approaches. By employing convolutional neural networks (CNNs) [14] and recurrent neural networks (RNNs) [15], Pixel Pinnacle achieves remarkable accuracy and robustness in character recognition, even in the presence of complex backgrounds, varying fonts, and skewed perspectives.

Moreover, Pixel Pinnacle boasts a user-friendly interface that facilitates seamless interaction with the OCR system, allowing users to effortlessly upload images, configure settings, and visualize recognition results. Through meticulous attention to usability and accessibility, Pixel Pinnacle aims to democratize access to OCR technology, empowering users from diverse backgrounds and skill levels to leverage its capabilities for their specific needs.

3.2. Architecture

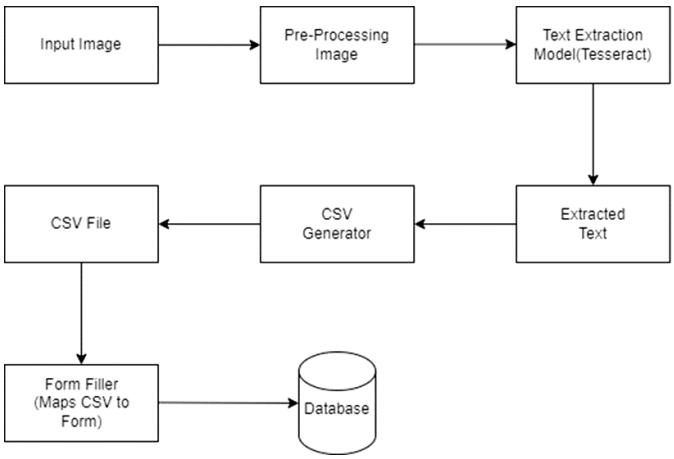

Fig 1: Proposed System Architecture

3.3. Pytesseract Library

Pytesseract [16] is a Python library that serves as a convenient wrapper for Google's Tesseract-OCR Engine, designed to extract text from images. Offering seamless integration within Python applications, Pytesseract simplifies the utilization of Tesseract's robust optical character recognition capabilities. Through a straightforward interface, developers can effortlessly process images in various formats, including JPEG, PNG, and TIFF, extracting text for subsequent analysis or manipulation. Pytesseract's strength lies in its ease of use, requiring minimal code to implement, making it accessible to developers of all skill levels. Furthermore, the library provides a range of configuration options, empowering users to fine-tune OCR settings to suit specific use cases, such as defining language models or adjusting segmentation modes. Supported across multiple platforms, including Windows, macOS, and Linux, Pytesseract offers versatility and flexibility in building OCR solutions tailored to diverse environments. Its active community fosters ongoing development, support, and knowledge sharing, ensuring Pytesseract remains a reliable and up-to-date tool for text extraction tasks in Python applications.

3.4. You Only Look Once (YOLO)

YOLO (You Only Look Once) [17] is a family of object detection models known for their ability to identify objects in real-time. Unlike earlier detectors that required multiple scans

of an image, YOLO achieves this with a single pass through a convolutional neural network. YOLO models come in different versions, each offering a balance between speed and accuracy. While YOLO may not be the most accurate detector, its efficiency makes it a valuable tool for real-time object detection tasks. In YOLO's world of object detection, bounding boxes are like spotlights highlighting what the model sees. These boxes are rectangles drawn around objects the model identifies in an image. YOLO predicts not only the presence of an object but also its location and size. It achieves this by dividing the image into a grid. Each grid cell is responsible for detecting objects within its area. YOLO predicts bounding boxes and confidence scores for each cell. The bounding boxes represent the center coordinates, width, and height of the detected object relative to the grid cell. By analyzing these bounding boxes and their corresponding confidence scores, you can understand what objects YOLO has spotted in the image and their precise locations.

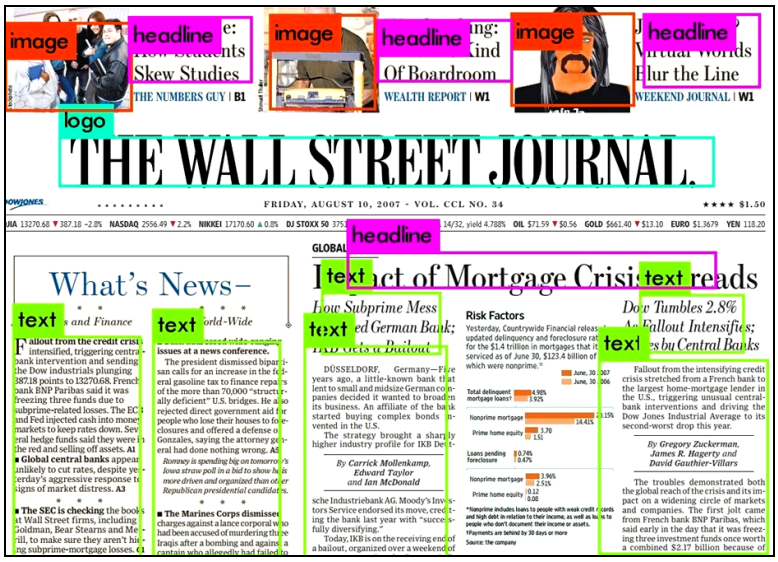

Fig 3: YOLO Example

3.5. Data Collection

I utilized the Indian drivers license dataset for this project. The dataset is provided by RoboFlow universe [18]. For training purposes, name, license number, date of birth, date of expiry, address have been labelled. Address which is superfluous for this assignment have been deleted. About 1700 license samples make up the collection. Almost every state license has been included in sample. Strategically expanding the dataset with images that showcase different lighting conditions, occlusions, or object scales can further enhance the model's ability to handle real-world variations. In essence, your dataset has equipped your YOLO model with the knowledge to not only identify objects but also precisely locate them within new images.

Fig 4: Dataset

3.6. Pre-processing

The Image preprocessing involved extraction of text from the image using the pytesseract library. The image underwent filtering to remove noise, resize, gray scale conversion using the opencv [19] library to improve the accuracy of the image-to-text conversion process. An image passing to tesseract

engine can cause reading all the characters residing inside in it. Hence the input image is first passed to trained yolo model to predict the fields to recognize text.

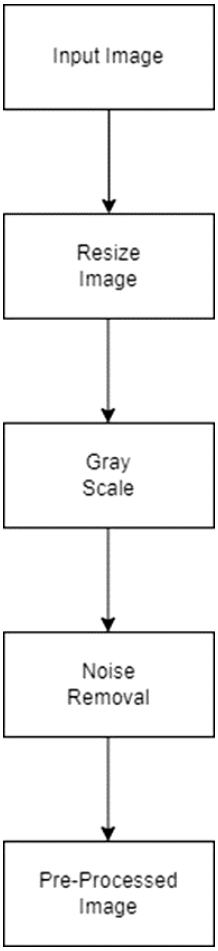

Fig 4: Pre-Processing.

3.6.1. Resize Image

Image resizing refers to the scaling of images. Scaling comes in handy in many image processing as well as machine learning applications. It helps in reducing the number of pixels from an image and that has several advantages e.g. It can reduce the time of training of a neural network as the more the number of pixels in an image more is the number of input nodes that in turn increases the complexity of the model. It also helps in zooming in on images. Many times we need to resize the image i.e. either shrink it or scale it up to meet the size requirements. OpenCV provides us several interpolation methods for resizing an image.

3.6.2. Noise Removal

Noise in images can come from various sources, like camera sensor limitations or poor lighting conditions. It can make the image appear grainy or speckled, hindering your ability to analyze it clearly. OpenCV offers a toolbox of denoising algorithms, each tackling noise differently. Some, like averaging or Gaussian blurring, work by smoothing out the image. They essentially blur pixels together, reducing noise but potentially sacrificing some fine details. For a more sophisticated approach, OpenCV provides Non-Local Means Denoising. This technique goes beyond the immediate neighborhood of a pixel and searches for similar patches across the entire image. By leveraging these similarities, it effectively removes noise while preserving image details more effectively.

3.6.3. Gray Scale

Grayscale conversion using OpenCV involves transforming a color image into a single-channel grayscale image, where each pixel represents the intensity of light rather than its color. OpenCV provides a simple and efficient method to perform this conversion. The process typically follows a weighted average approach, where the intensity values of the Red (R), Green (G), and Blue (B) channels are combined to calculate the grayscale intensity. This method accounts for human perception of light by applying different weights to each channel. The formula used for grayscale conversion is usually something like:

$$Y=0.299\times R+0.587\times G+0.114\times B$$

Where Y represents the grayscale intensity, and R, G and B represent the intensities of the Red, Green, and Blue channels respectively. OpenCV provides a function for converting cv2.cvtColor() images between different color spaces. When converting a color image to grayscale, you typically usethe parameter cv2.cvtColor()_BGR2GRAY if the image is in BGR format cv2.Color_RGB2GRAY or if it's in RGB format. This function applies the aforementioned weighted average formula to each pixel, resulting in a grayscale image where each pixel value ranges from 0 (black) to 255 (white), representing the intensity of light.

3.7. Tesseract Model

Tesseract OCR, developed by Google, is one of the most popular open-source optical character recognition (OCR) engines available today. It's designed to recognize and extract text from images or scanned documents. Tesseract is known for its accuracy and language support, making it widely used for a variety of applications including document analysis, text extraction from images, and automated data entry.

Originally developed by Hewlett-Packard in the 1980s, Tesseract was later maintained by Google after its acquisition of HP's OCR research team in 2006. Since then, Google has significantly improved Tesseract, making it more accurate, efficient, and adaptable to a wide range of languages and scripts.

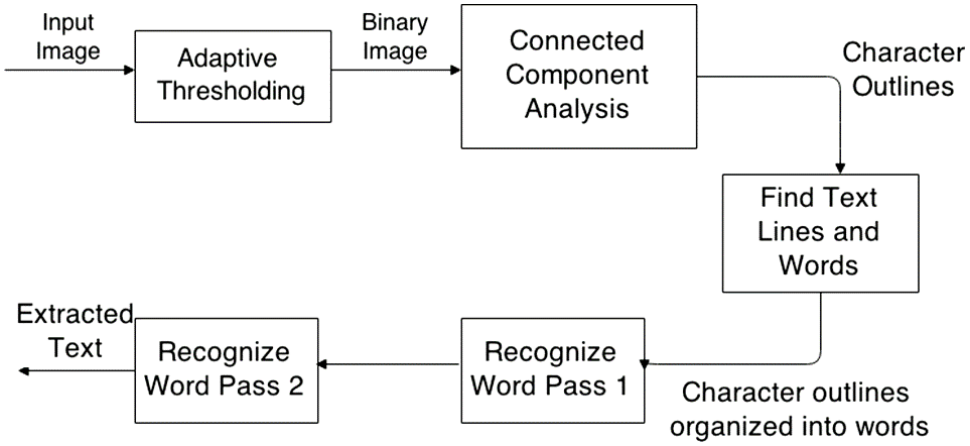

Fig 5: Tesseract model

The working of Tesseract OCR by Google involves a series of intricate steps that ultimately enable it to accurately recognize and extract text from images or scanned documents. At its core, Tesseract employs a combination of advanced image processing techniques and machine learning algorithms to decipher the textual content within an image. Initially, the input image undergoes preprocessing steps such as noise reduction, binarization, and desk wing to enhance its readability and ensure uniformity across different types of input.

Following preprocessing, Tesseract employs a convolutional neural network (CNN) architecture, which has been trained on vast amounts of annotated text data, to analyze the patterns of light and dark pixels within the image. This neural network dissects the image into smaller regions, known as text blocks or segments, and then proceeds to classify each segment into individual characters based on its visual features.

Tesseract's recognition process is further augmented by language models and dictionaries, which aid in resolving ambiguities and improving accuracy, particularly when dealing with languages with complex orthographies or ambiguous characters. Additionally, Tesseract's modular design allows for the integration of specialized techniques, such as character-level confidence scoring and contextual analysis, to further refine the recognition results. Once the characters have been identified, Tesseract reconstructs them into coherent words, sentences, and paragraphs.

4. Experiments and Results

4.1. Confusion Matrix

In this work, the image predicted by the model and trained image is evaluated using the Confusion Matrix. This matrix organizes predictions made by the model against ground truth annotations, offering insights into the model's performance across different classes of objects. True positives represent correct detections where both the model and ground truth agree on the presence of an object of a particular class. False positives occur when the model incorrectly identifies an object that is not present in the ground truth annotations. Conversely, false negatives signify instances where the model fails to detect objects that exist in the ground truth.

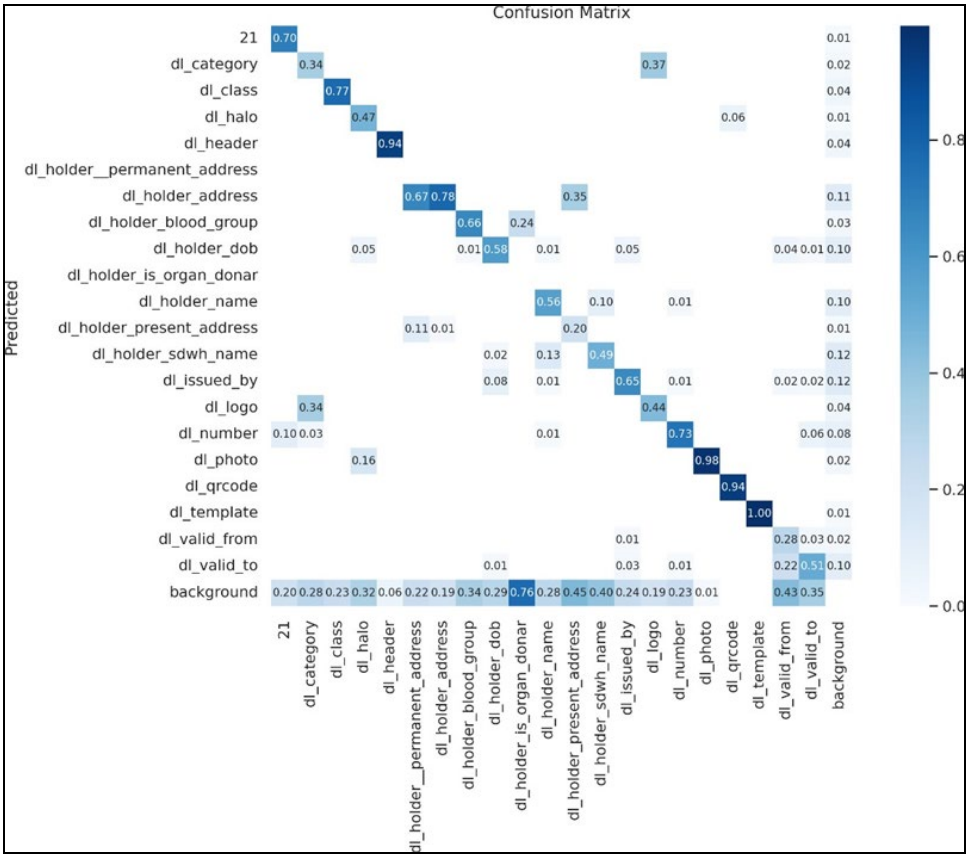

Fig 4: Confusion matrix

4.2. User Interface

The development of the web application was carried out with Python's Django web framework [18]. Django was preferred due to its simplicity and adaptability, which allowed for the creation of a dynamic and expandable web application. The application is hosted on a cloud server and can be accessed via any web browser. The required Python packages and libraries were installed through pip. HTML, CSS, JavaScript and Ajax were utilized in the creation of the front-end. The user details entered through form are then stored in database.

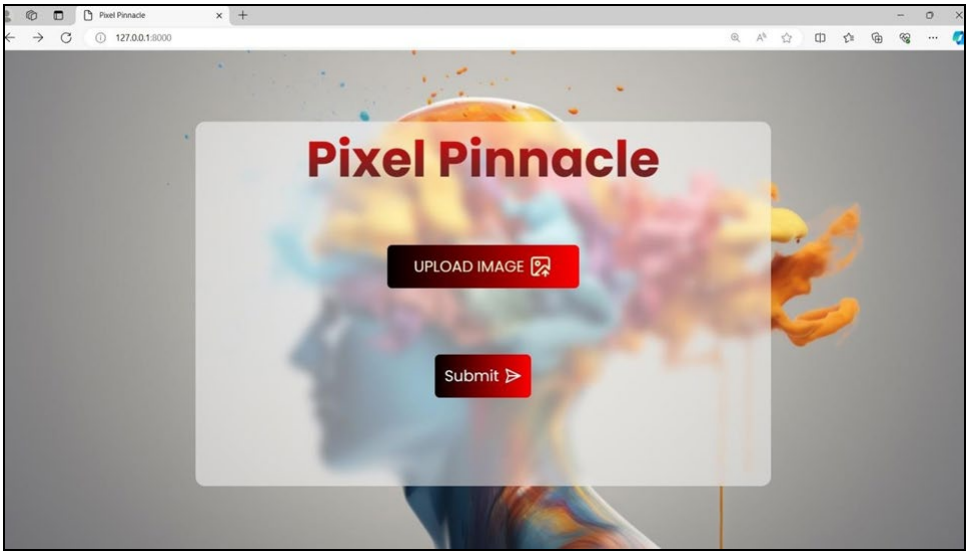

Fig 5: Home Page

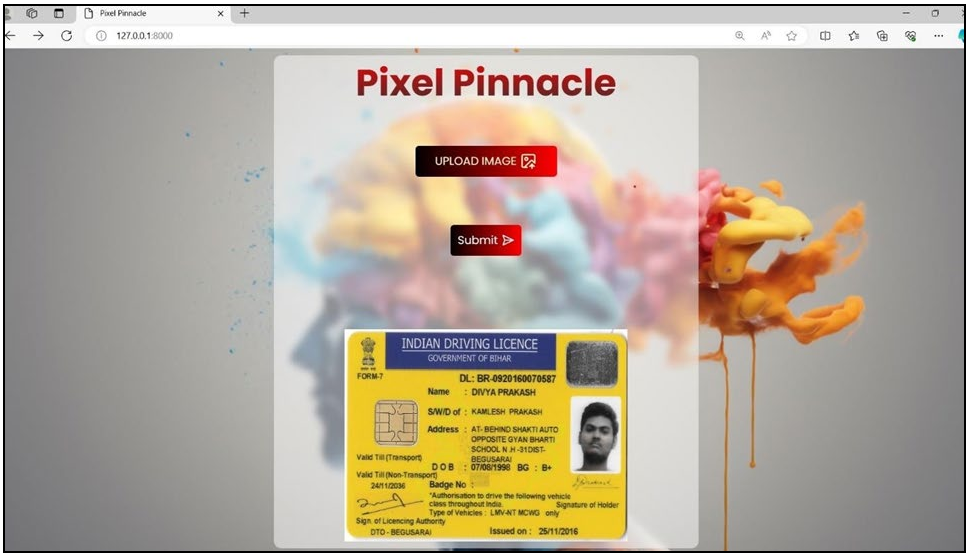

Fig 6: Preview of Input Image

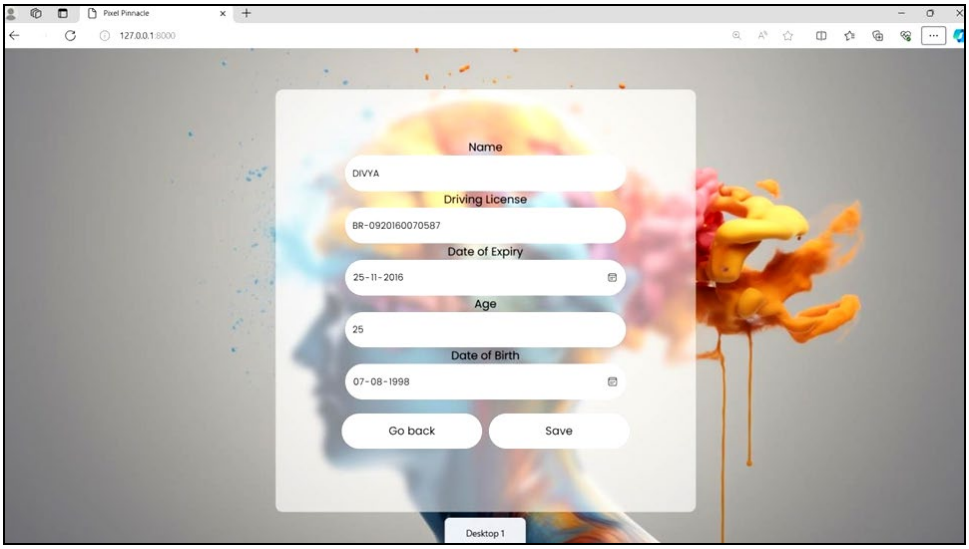

Fig 7: Result page

5. Conclusion & Future Scope

5.1. Conclusion

In this work, an image to text conversion model that leverages the PyTesseract library, OpenCV library, YOLO, Tesseract architecture and various pre-processing techniques is proposed to generates a necessary details of ID holder from the input image. Evaluated the model using confusion matrix and observed that while it is able to capture some of the important content from the input image, there is still room for improvement. However, the results show promise and suggest that the approach has the potential to be further developed into a useful tool for automated data entries.

Pixel Pinnacle's success lies not only in its technical prowess but also in its potential to drive transformative changes in data management practices across diverse domains. By offering a user-friendly interface, remarkable accuracy, and robust performance, Pixel Pinnacle empowers individuals and organizations to unlock the full potential of textual data, facilitating informed decision-making, enhancing productivity, and enabling new avenues for innovation.

As we reflect on our journey with Pixel Pinnacle, we are filled with a sense of pride and accomplishment. Our project stands as a testament to the power of collaboration, perseverance, and innovation in tackling real-world challenges. Moving forward, we remain committed to further refining and enhancing Pixel Pinnacle, ensuring that it continues to meet the evolving needs of users and remains at the forefront of OCR technology.

5.2. Future Scope

There are several areas in which this project could be extended in the future. First, we could explore the use of more advanced pre-processing techniques text data, such as using different filtering or normalization methods. Second, we could experiment with different model architectures or fine-tuning techniques to improve the quality of the extracted text. Finally, we could consider deploying the model in a real-world application, such as automated data entry application where happens a lot of manual data entry. Overall, there are many exciting possibilities for future work in this area, and I believe that this project has provided a solid foundation for further exploration.

References

1. Zdebska, Tetyana *et al*. "Optical Character Recognition." Computational linguistics and intelligent systems: proceedings of the 4nd International conference (2), 2020.
2. Dome, Saurabh, and Asha P. Sathe. "Optical charater recognition using tesseract and classification." 2021

International Conference on Emerging Smart Computing and Informatics (ESCI). IEEE, 2021.
3. Khurana, Diksha, *et al*. "Natural language processing: State of the art, current trends and challenges." Multimedia tools and applications. 2023; 82(3):3713-3744.
4. Mahajan, Shilpa, and Rajneesh Rani. "Text detection and localization in scene images: a broad review." Artificial Intelligence Review. 2021; 54(6):4317-4377.
5. Patel, Chirag, Atul Patel, and Dharmendra Patel. "Optical character recognition by open source OCR tool tesseract: A case study." *International Journal of Computer Applications*. 2012; 55(10):50-56.
6. Shen, Mande, and Hansheng Lei. "Improving OCR performance with background image elimination." 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). IEEE, 2015.
7. Kaundilya, Chandni, Diksha Chawla, and Yatin Chopra. "Automated text extraction from images using OCR system." 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2019.
8. Wankhede, Poonam A., and Sudhir W. Mohod. "A different image content-based retrievals using OCR techniques." 2017 international conference of electronics, communication and aerospace technology (ICECA). Vol. 2. IEEE, 2017.
9. Imam, Niddal H., Vassilios G. Vassilakis, and Dimitris Kolovos. "OCR post-correction for detecting adversarial text images." *Journal of Information Security and Applications* 66 (2022): 103170.
10. Lat, Ankit, and C. V. Jawahar. "Enhancing OCR accuracy with super resolution." 2018 24th International Conference on Pattern Recognition (ICPR). IEEE, 2018.
11. Bui, Quang Anh, David Mollard, and Salvatore Tabbone. "Selecting automatically pre-processing methods to improve OCR performances." 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). Vol. 1. IEEE, 2017.
12. Smitha ML, Antony PJ, Sachin DN. "Document image analysis using imagemagick and tesseract-ocr." *International Advanced Research Journal in Science, Engineering and Technology (IARJSET)*. 2016; 3:108-112.
13. Ranjan, Ashish, Varun Nagesh Jolly Behera, and Motahar Reza. "Ocr using computer vision and machine learning." Machine Learning Algorithms for Industrial Applications, 2021, 83-105.
14. Ketkar, Nikhil *et al.* "Convolutional neural networks." Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, 2021, 197-242.
15. Cheng, Samuel. "Recurrent neural networks." (2023).
16. Saoji, Saurabh, *et al*. "Text recognition and detection from images using pytesseract." *J Interdiscip Cycle Res*. 2021; 13:1674-1679.
17. Diwan, Tausif, G. Anirudh, and Jitendra V. Tembhurne. "Object detection using YOLO: Challenges, architectural successors, datasets and applications." multimedia Tools and Applications. 2023; 82(6):9243-9275.
18. www.universe.roboflow.com
19. Sharma, Ayushi, *et al*. "Object detection using OpenCV and python." 2021 3rd international conference on advances in computing, communication control and networking (ICAC3N). IEEE, 2021.