DEVELOPMENT OF A MODULAR OPTICAL CHARACTER RECOGNITION AND LARGE LANGUAGE MODEL APPLICATION

LIEW ZI YUE

UNIVERSITI TUNKU ABDUL RAHMAN

DEVELOPMENT OF A MODULAR OPTICAL CHARACTER RECOGNITION AND LARGE LANGUAGE MODEL APPLICATION

LIEW ZI YUE

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Electrical and Electronic Engineering with Honours

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2026

DECLARATION STATEMENT

I hereby declare that this project report is my original work, with all sources duly acknowledged, and that it has not been submitted previously or concurrently for any degree or award at UTAR or elsewhere. I also declare the following *(check the boxes if applicable)*:

☒ Generative AI tools (e.g., ChatGPT, Gemini, Copilot, etc.) were used for language refinement, such as grammar correction and readability improvement.

☒ Vibe coding tools and/or frameworks (e.g., Replit Agent, Cursor Composer, Bolt, etc.) were used in project development.

☐ Other declaration(s), if any (*please consult your supervisor for guidance in completing this section*):

_____

_____

I take full responsibility for the originality, accuracy, and integrity of all content presented in this report. No Generative AI tools were used to manipulate project data and results in ways that could cause falsification or misrepresentation.

Name : Liew Zi Yue

ID No. : 2103108

Date : 29/4/2026

COPYRIGHT STATEMENT

This final year project report is submitted in partial fulfilment of the requirements for the degree of Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman (UTAR). This final year project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this final year project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Ir Ts Dr Tham Mau Luen as my research supervisor and Ir Ts Dr Chua Sing Yee as my research co-supervisor for their invaluable advice, guidance and enormous patience throughout the development of the research.

I would also like to thank my dearest family and friends for their help, friendship, and support throughout this research, which have been instrumental to the success of this research project.

ABSTRACT

Optical Character Recognition (OCR) and Large Language Models (LLMs) can be combined to improve document digitization, especially when traditional OCR struggles with complex layouts, multilingual content, and limited contextual understanding. This project addresses these limitations by developing a modular OCR-LLM application designed for English, Malay, and Chinese document processing. The methodology involves a flexible pipeline that evaluates five free and open-source OCR engines (Tesseract, EasyOCR, PaddleOCR, DocTR, and DeepSeek-OCR) and two locally deployable 8-billion-parameter LLMs (DeepSeek-R1 and Llama 3.1), which are then deployed via a Streamlit web interface. Performance evaluation on single-column and unstructured layouts demonstrated that DeepSeek-OCR 2 achieved the highest robustness, maintaining 90.17% accuracy on simple layouts and 83.78% on complex structures. For contextual refinement, DeepSeek-R1 outperformed Llama 3.1 with a 94.77% BERT-based Semantic Similarity Score (BERTScore) and 81.38% Bilingual Evaluation Understudy with Representations from Transformers (BLEURT) score. While post-processing by the Artificial Intelligence (AI) agent improved the average F1-score to 91.34%, it introduced a trade-off by increasing the Character Error Rate (CER) and Word Error Rate (WER) due to semantic reformatting. Overall, this project demonstrates that combining free, open-source, and locally deployable OCR engines with LLMs in a modular architecture can produce a flexible, and user-friendly document processing system without Application Programming Interfaces (APIs) charges. Moving forward, future work should focus on GPU-based optimization, improved support for handwritten and highly degraded documents, stronger multilingual handling, and further enhancement of agentic AI capabilities for more functions.

Keywords: Optical Character Recognition, Large Language Model, DeepSeek-OCR 2, Multilingual Document Processing, AI Agent, Document Digitization, Streamlit Web Application

Subject Area: TA168 Systems engineering

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ALTO	Analyzed Layout and Text Object
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BERTScore	BERT-based Semantic Similarity Score
BLEURT	Bilingual Evaluation Understudy with Representations from Transformers
CER	Character Error Rate
CLS	Direction Classifier
CNN	Convolutional Neural Network
CRAFT	Character Region Awareness for Text Detection
CRNN	Convolutional Recurrent Neural Network
CSS	Cascading Style Sheets
CTC	Connectionist Temporal Classification
DB	Differentiable Binarization
FN	False Negative
FP	False Positive
GQA	Grouped-Query Attention
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
hOCR	HTML-based OCR Output Format
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
KIE	Key Information Extraction
LLM	Large Language Model
LSTM	Long Short-Term Memory
MLA	Multi-head Latent Attention

MoE	Mixture-of-Experts
OCR	Optical Character Recognition
PDF	Portable Document Format
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network
RoPE	Rotary Positional Embedding
SVTR	Scene Text Recognition with a Single Visual Model
TN	True Negative
TP	True Positive
UI	User Interface
VGG	Visual Geometry Group
VLM	Vision-Language Model
WER	Word Error Rate
XML	Extensible Markup Language

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Converting physical documents into digital form has become a crucial task for many industries, as managing large amounts of information from paper-based sources is now a common need. As stated by TechPrate (2023), organizations in sectors such as banking, healthcare, education and government process large volumes of information contained in printed reports, handwritten notes and scanned invoices. Manually converting this information into digital formats is not only time-consuming but also prone to human error. As a result, Optical Character Recognition (OCR) has become an important tool for automating this process. OCR systems can recognize printed or handwritten text from images and convert the text into editable or searchable content, improving operational efficiency and reducing the workload of staff (TechPrate, 2023).

However, most conventional OCR tools were developed to extract text from clean, structured documents that are printed in a standard font on plain backgrounds. These OCR systems reliably produce accurate results under those conditions, but their accuracy drops significantly when dealing with real world documents that feature multiple columns, tables, unusual layouts, a mix of printed and handwritten text as well as multilingual content (Sharma, 2023). In such cases, conventional OCR engines struggle to maintain accuracy, thus producing incomplete or incorrect results which is a serious limitation when the extracted data is used for critical purposes such as patient records, legal agreements, or financial reporting.

Another drawback of OCR systems is lack of ability to interpret the meaning or context of the extracted text. They can recognize characters and words but cannot interpret the content, the relationships between different elements of the document and the structure of the content itself. This lack of contextual understanding means that extracted text often requires additional manual correction, annotation or restructuring before it can be meaningfully used.

The rise of Artificial Intelligence (AI) has introduced new approaches to address these limitations. Large Language Models (LLMs) have shown outstanding ability in processing and producing text that resembles natural human language (Ferraris et al., 2025). LLMs can help bridge the gap by understanding context, handling domain-specific language and summarizing or reorganizing content based on what the user needs. When integrating LLMs with OCR, LLMs can add a layer of intelligence that turns raw text into useful information. For example, in a scanned academic paper, the OCR might detect all the words, but the LLM can recognize which parts are the abstract, introduction and conclusion. This kind of capability is valuable in many practical use cases. To support this, models like Llama (Large Language Model Meta AI) and DeepSeek provide open-source solutions (Browne, 2025) with strong text understanding capabilities, making them suitable for integration into research and development projects.

This project proposes a modular application that combines the strengths of OCR and LLM technologies to create a more adaptable and intelligent document or image processing system. A modular approach allows each part such as image pre-processing, the OCR engine, and the language model to function independently, making it easier to test, update or replace individual components without rebuilding the entire system. This OCR-LLM integration is useful in research and industry settings where continuous improvement is essential. The system will be deployed as a web-based application, offering users an accessible interface for uploading documents, viewing extracted text and interacting with the analysis tools. To ensure comprehensive text extraction performance, this project employs five open-source OCR engines which are Tesseract, EasyOCR, PaddleOCR, DocTR, and DeepSeek-OCR. These engines were chosen based on their active development, broad usage in both academic and industrial contexts and their support for multilingual text processing in Chinese, Malay and English.

Even with recent improvements, there are still major challenges in making OCR work well for different types of documents, keeping it accurate in important fields, and enabling the system not only to extract text, but also to interpret and refine the extracted content. This project aims to tackle these

problems by creating a system that is flexible, easy to improve and smart in handling documents.

1.2 Importance of the Study

The significance of this study lies in its focus on addressing the persistent limitations that existing OCR systems have yet to resolve. Many industries still struggle with documents that include complex layouts, multiple languages or handwritten elements (Sharma, 2023). Errors in these contexts can slow down workflows and increase costs.

By developing a modular OCR system enhanced with LLMs, this project offers a practical way to improve not only extraction accuracy but also contextual understanding. The modular architecture ensures long-term viability where it can be updated or replaced independently, allowing organizations to easily adopt better OCR engines or more advanced LLMs without a complete system redesign.

Furthermore, the integration of LLMs represents a critical advancement beyond mere text extraction. The system aims to understand the content, allowing tasks like summarizing, translating and organizing information. This makes the extracted text more useful and reduces the need for people to fix or process documents manually.

Finally, by developing the system as a user-friendly, web-based application, this project ensures the solution is accessible to a wide range of users, including small businesses and institutions lacking advanced technical resources. This study, therefore, not only contributes a functional tool but also provides a reference framework for building intelligent and adaptable document processing systems.

1.3 Problem Statement

1.3.1 Limitations of Traditional OCR in Handling Complex Inputs

Traditional OCR tools have been widely used for digitizing printed or handwritten text. Although they work reasonably well for clean, printed documents, they often struggle with complex formats such as tables, multi-column layouts, unusual fonts, handwriting and multilingual content (Sharma, 2023). Additionally, when documents contain more than one language or

handwritten notes, the accuracy of these tools drops significantly. This limits their effectiveness for use in practical situations where documents are messy, varied or non-standardized.

1.3.2 Cost and Data Privacy Concerns in Cloud-Based OCR and LLM Services

Many high performance OCR and LLM services are provided through commercial cloud-based Application Programming Interfaces (APIs). Although such services can achieve impressive performance, they pose two major challenges. First, their usage-based pricing structure can become expensive when large volumes of documents are processed regularly. For instance, Google Cloud Vision requires payment for every processed document, making the operation costly (Google Cloud, 2026). The second challenge is that uploading sensitive documents to external cloud servers introduces privacy risks. Studies reveal that passing user information to a cloud-based LLM model leads to security risks, such as data leaks and breaches (Li et al., 2025). Therefore, there is a clear need for a locally deployed OCR-LLM solution that can provide strong performance without the recurring cost and privacy risks of external cloud dependencies.

1.3.3 Lack of Contextual Understanding in Extracted Text

Most OCR tools can recognize individual characters or words, but they do not understand the context or meaning behind the text. This limitation becomes a problem when working with documents that contain complex formatting, abstract language or a mix of content types. OCR systems struggle not only to detect headings and interpret figures from tables but also to understand unfamiliar layouts, integrate visual elements like charts or diagrams and summarize key points accurately (Tanasă and Oprea, 2025). Without the ability to understand context, the extracted text may not be useful for deeper processing or analysis. To address this issue, the integration of LLMs with OCR offers a more robust solution by enhancing text understanding, summarization and generation.

1.4 Aim and Objectives

The aim of this project is to build a flexible and interactive system that combines OCR with LLMs to support accurate text extraction and processing from images or scanned documents. The project focuses on three key objectives.

- To develop a modular application that integrates open-source OCR engines for versatile text extraction.
- To integrate LLMs for text summarization, generation, and translation, allowing users to synthesize, interpret, and evaluate extracted content more effectively.
- To design a user-friendly interface to facilitate easy interaction and visualization of results, making the system practical and accessible for end users.

1.5 Scope and Limitation of the Study

The scope of this project is centered on developing a modular pipeline that integrates OCR engines with LLMs for enhanced document processing. Specifically, the system supports three languages (English, Malay, and Chinese) and is designed to handle both simple text-based documents and more complex layouts, including tables and diagrams. In addition to text extraction, the LLM extends the capabilities of the system by enabling higher-level functions such as summarization, translation, and structured data generation, making it useful for diverse academic and industrial applications. The solution is implemented as a web-based application to ensure accessibility, scalability, and interactive use.

However, the study is subject to several limitations. First, although the system was designed to support local deployment, some experiments were conducted using Google Colab with a T4 GPU to accelerate model testing. Full local GPU deployment was limited by hardware and cost constraints, which may affect processing speed and efficiency when handling very large datasets on local CPU-based machines. Second, while the modular design allows for flexibility in replacing or upgrading components, the performance of the system is still influenced by the accuracy of the OCR engines, particularly in cases of extreme text degradation, handwritten notes, or highly complex scripts. Lastly, the resource-intensive nature of LLMs poses constraints for real-time or high-

volume processing, although cloud or GPU-based solutions could be explored in future extensions of this work.

1.6 Contribution of the Study

This study contributes to the development of a modular OCR-LLM pipeline for multilingual document processing. The proposed system combines open-source OCR engines, local LLMs, AI agent post-processing, and a web-based interface into one engineering framework. This allows scanned documents and images to be processed from raw input into editable, refined digital text.

This study also provides a comparative evaluation of multiple OCR engines, including Tesseract, EasyOCR, PaddleOCR, DocTR, and DeepSeek-OCR. Their performance is evaluated using accuracy, precision, recall, F1-score, CER, and WER to identify the most suitable OCR engine for handling different document layouts and multilingual content. This provides a clearer understanding of how different OCR engines perform under the same evaluation pipeline.

In addition, this project demonstrates how locally deployed open-source LLMs can be integrated with OCR output to support contextual understanding, summarisation, translation, text refinement, and chat-based interaction. By using local deployment instead of relying on paid external APIs, the system supports better privacy, lower long-term cost, and greater flexibility for future modification.

Lastly, this study contributes a functional Streamlit web application that allows users to upload documents, view extracted text, refine the output, and interact with the processed content. Therefore, the project serves not only as an OCR application, but also as an engineering reference for building flexible, privacy-aware, open-source, and multilingual document processing systems.

1.7 Outline of the Report

This report is structured into five chapters. Each chapter covers a key part of the study, from the introduction and literature review to methodology, results and discussion, and finally the conclusion and recommendations.

Chapter 1 introduces the background and importance of the study, presents the problem statement, and outlines the aim and objectives. It also

discusses the scope, limitations, contributions, and the overall structure of the report.

Chapter 2 reviews a large number of relevant studies related to OCR engines, LLMs, and their integration strategies, while also examining evaluation metrics and identifying research gaps.

Chapter 3 outlines the methodology adopted for this project, including the overall research design, implementation approach, and the detailed work plan.

Chapter 4 presents the results and discussion, where the performance of OCR engines, LLMs, and the integrated OCR + AI-Agent system is evaluated using various metrics.

Chapter 5 concludes the project by summarising the major achievements based on the stated aims and objectives. It also provides recommendations for future work, such as GPU-based optimization and agentic AI enhancement.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter provides a review of the literature and technologies that support this research. It begins by describing the principles and development of OCR, including both traditional rule-based methods and modern deep learning-based techniques. Five OCR engines, which are Tesseract, EasyOCR, PaddleOCR, DocTR, and DeepSeek-OCR, are examined to understand their mechanisms and performance. The chapter also discusses LLMs, explaining their architecture, capabilities and potential to enhance OCR output through contextual understanding and advanced text processing. Existing studies that integrate OCR with LLMs are then analyzed to identify their methods, advantages, and limitations. The findings from this review form the basis for identifying research gaps and defining the modular approach proposed in this work.

2.2 Optical Character Recognition (OCR)

OCR is used to convert text by recognizing the printed text or handwriting in images, to make it searchable and editable as digital text. Christian and Kusuma (2023) explain that the process of OCR begins with the scanning of image followed by pre-processing of the image to deskew or remove background noise so that the text can be uniform. The OCR then extracts potential text from the scanned documents, analyzing each individual letter, numeral, and symbol, converting the information from an image file to digital text. This digital format allows users to store paper documents such as receipts, forms, and invoices more efficiently and retrieve them easily through search functions.

2.2.1 Traditional OCR

OCR systems can generally be divided into two main types which are traditional rule based methods and deep learning based approaches. Traditional OCR systems, particularly rule-based methods, rely on a fixed sequence of operations such as binarization, text detection using connected component analysis, and character recognition via pattern matching. These systems typically process

each text region independently and often fail to capture the spatial relationships between elements, leading to errors in reconstructing the logical structure (Singh, 2025). While effective in clean and simple documents, rule-based methods tend to break down in the presence of noisy background or non-rectangular fonts (Datta et al., 2025).

Traditional OCR systems work in a predetermined pipeline: first, the pre-processing phase; then, the segmentation phase; and finally, the pattern recognition phase. During the pre-processing phase, the input image will be binarized, so that there is a simplification of the further processing. Additionally, noise removal can be applied to eliminate any extra marks or smudges and deskewing to straighten tilted text lines (Patil et al., 2022). In the segmentation phase, the system identifies the text lines, words and individual characters using connected component analysis. Finally, in the recognition phase, the characters are identified using pattern matching. However, these are a deterministic or fixed rule-based pipeline, which does not adapt well to low quality, handwritten or other unconventional format image (Olson and Berry, 2021).

Early versions of Tesseract used traditional OCR techniques such as pattern matching and rule-based methods, while version 4.0 and later introduced Long Short-Term Memory networks, making it a deep learning–based OCR engine (Patience et al., 2024). ABBYY FineReader is another widely used OCR engine which also relied heavily on rule-based approaches in its earlier versions.

2.2.2 Modern OCR

In contrast, modern OCR engines utilize deep learning models such as convolutional and recurrent neural networks (Sharma, 2023). Like the earlier versions of OCR, modern day OCR use machine learning-based components that can learn from the available data and allows for a much more robust tolerance for differences in font styles, font orientations and complex layouts. Deep learning-based OCR offers greater flexibility and accuracy, particularly when handling multiple languages or processing text from real-world images (Biró et al., 2023). Thus, modern OCR tend to be more reliable than the traditional approaches for dealing with handwriting and visually complicated documents.

Unlike traditional OCR systems that solely operated as a pipeline or a sequence of steps, more recent deep learning-based OCR systems, have combined text detection and recognition into one framework (Singh, 2025). For example, convolutional recurrent neural network (CRNN) combines feature extraction, sequence modeling, and transcription all in the same network (Sai et al., 2020). Similarly, transformer based architectures such as single visual model for scene text recognition (SVTR) are able to capture global dependencies in image features, allowing for more flexibility in the recognition of irregular layouts, patterns and non-uniformities (Biró et al., 2023). The advantage of this end-to-end framework is that it mostly eliminates the need for handcrafted pre-processing and segmentation steps, thus making them more scalable in real-world applications.

2.3 Working Mechanisms of OCR Engines

This section explains the working mechanisms of the modern OCR engines discussed in Section 2.2, with focus on the engines relevant to this study which are Tesseract (Version 4++), EasyOCR, PaddleOCR and DocTR. Their processing pipelines consist of three main stages: text detection, text recognition, and output formatting. Each engine follows a distinct approach to locate text in an image, identify its content, and return the result in a readable format.

2.3.1 Tesseract OCR

Tesseract OCR is a widely used OCR engine originally developed by HP and later maintained by Google. It follows a pipeline-based strategy for rendering scanned or imaged text into machine-readable characters. The first step of the Tesseract process is image pre-processing. The input image is initially converted to grayscale to simplify it by removing color information and reducing computational load. It is then transformed into a binary image using Otsu's thresholding. According to Akhil (2016), this method applies a single global threshold value across the entire image to separate dark (text) pixels from light (background) pixels. In contrast, adaptive thresholding calculates different thresholds for different regions, which can help with uneven lighting or shadows. This also automatically determines a threshold value to separate dark pixels which assumed to be text and from light pixels which assumed to be background

(*Thresholding*, 2024). Following binarization of the image, Tesseract performs connected-component analysis, where it analyzes the image pixel by pixel to group connected black pixels as blobs. These bloblets are identified as individual characters or fragments of characters, and bounding boxes are labeled around them to show the possible areas where text can reside (Smith, 2007).

According to Gadade (2019), Tesseract 2.0, the engine performed a two-pass recognition process, initially recognizing words and adapting via a classifier, followed by a second pass over uncertain words to improve accuracy. Tesseract 4.0 then replaces this approach with a modern LSTM recognizer, enhancing text recognition performance significantly. LSTM is a type of Recurrent Neural Networks (RNN) that are particularly suited for sequence data. LSTMs process text left to right one character at a time, while maintaining context of previous characters during text recognition to improve accuracy (Ingole and Jain, 2025). For example, when the model sees "Th" as the first two letters it is more likely to give the next character an "e" to fill out the word "The." Then, Tesseract used a two-pass recognition approach, where the first pass attempted word recognition using a dictionary and the second pass reprocessed uncertain or rejected words for better accuracy.

Finally, the recognized characters are turned back to an ordinary text string. Normally, Tesseract will just return the raw text, but the spatial location of each word can be included in output formats such as hOCR or ALTO XML which includes the x, y location, width, height, and confidence of each word or line found (Wajer, 2023). This type of output format helps as a means for the layout awareness needed to accurately get out words in the right location in one page layouts. From the sum of the parts, Tesseract is a rule-based, sequence sensitive OCR engine that works well on relatively clean documents, but does not work well when the background is noisy or the text layout is bent. The overall workflow of Tesseract OCR is illustrated in Figure 2.1.

Figure 2.1: Architecture of Tesseract OCR Framework

2.3.2 EasyOCR

EasyOCR is a modern OCR engine that uses deep learning. It is known for its ease of use, support for multiple languages, and ability to read text even when presented in different fonts and layouts. EasyOCR uses neural networks for both text detection and text recognition (Tammana, 2023). The first step in EasyOCR is text detection which uses the deep learning model called CRAFT (Character Region Awareness for Text Detection). As described by Aloyan (2025), CRAFT scans the input image and outputs a series of heatmaps of the input image that indicates where text is likely to appear in the input image. The heatmap is then used to create bounding boxes for words or lines of text.

After detecting the text regions, the process moves on to text recognition. As stated by Xiong and Liu (2024), EasyOCR has text recognition capabilities based on Convolutional Recurrent Neural Networks (CRNN) which takes advantage of both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) for recognition. In this case, the CNN part extracts visual features from the text image based on edges and strokes while the RNN component implemented with LSTM understands the sequence created by these features and can output entire words (MathWorks, 2023). EasyOCR uses ResNet as the backbone of the CNN for feature extraction, enabling robust and accurate representation of text patterns before passing them to the LSTM for sequence modeling. The output is further processed by using a technique known as a Connectionist Temporal Classification (CTC) loss (Hosseini, Shabaninia and Nezamabadi-pour, 2025). This is done in such a way that it enables EasyOCR to moderate the output of the model by simply removing the repeated characters and unnecessary white spaces. For example, if the raw model output is "HH-EE-LL-LOO" the CTC layer would convert it to "HELLO".

The final output from EasyOCR is a raw text, which can be accompanied by the confidence score, and the area of the bounding box where the text appears. The coordinates of the area are provided by four corner coordinates and allows the model to report text in cases where there is rotated or skewed text (Jaided Ai, n.d.). The framework of EasyOCR is presented in Figure 2.2.

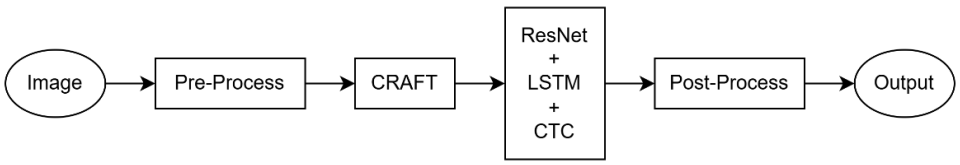

Figure 2.2: Architecture of EasyOCR Framework

2.3.3 PaddleOCR

PaddleOCR is an extended OCR system created as part of the PaddlePaddle deep learning framework that contains both deep learning and layout analysis, making it particularly effective for processing documents that incorporate complex layouts like tables, forms, or multi-column formats (Cui et al., 2025). Lim et al (2025) state that the initial step involves determining the textual areas in the document using a model called Differentiable Binarization (DB). DB produces a pixel-level heatmap that shows the probability of all the pixels being part of the textual area. Then DB uses adaptive thresholding to distinguish the text from the remaining background (also separating multi-layer layout images) and it produces accurate polygon-shaped bounding areas around the detected text (Liao et al., 2020). PaddleOCR uses an optional Direction Classifier (CLS) to correct the orientation of rotated or vertical text regions before recognition, meaning that PaddleOCR has the ability to accurately detect not only straight horizontal text but also curved, slanted and vertical text.

According to the PaddleOCR 3.0 technical report (Cui et al., 2025), after the text areas are determined, PaddleOCR proceeds to do the recognition with a CRNN model, which has in fact been optimized to recognize characters in varying fonts and sizes relative to one another. Furthermore, PaddleOCR incorporates layout analysis capability provided by an existing module called PP-Structure. The PP-Structure module can detect cell boundaries and layout content into rows and columns for table or spreadsheet identification or extraction. Furthermore, the PP-Structure module can perform key-value pair extraction useful in structured documents, like forms, where fields like "Name", or "Date", need to be matched with the associated data.

Cui et al. (2025) state that PaddleOCR produces output in two formats. The output for normal OCR includes the polygon coordinates of the text areas, the text and a confidence score while the output for structured documents is

provided as JSON or Markdown files. Overall, PaddleOCR has one of the most robust and complete OCR solutions available, with significant capabilities of both natural text recognition and layout analysis. The architecture of PaddleOCR is shown in Figure 2.3.

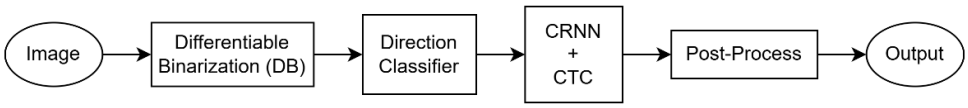

Figure 2.3: Architecture of PaddleOCR Framework

2.3.4 DocTR

DocTR, which stands for Document Text Recognition, is a relatively new OCR library designed for high-accuracy text extraction and document understanding. It uses a deep learning approach similar to PaddleOCR but is more focused on document-level interpretation rather than speed. Liao et al. (2023) outlines that DocTR uses the DB-ResNet50 model (similar to the DB model from PaddleOCR) which predicts the text areas and draws polygons around the detected text areas as a heatmap. However, it was designed with accuracy in mind, so it does not work very well for real-time application as it is slow.

Referring to Mindee (2021), DocTR utilizes the CRNN model based on the VGG16 backbone for text recognition. This model extracts features from the detected text regions and recognizes the content using sequential analysis. The main strength of DocTR lies in its document understanding features. DocTR allows grouping text blocks into paragraphs, titles, and sections using layout detection methods. DocTR supports key-value extraction capabilities, which is useful when processing documents that contain phrases such as "Invoice Number: 123", or the example "Total Amount: RM50.00". Each of the text blocks has been tagged with semantic types to represent their meaning in the document (Liao et al., 2023).

DocTR outputs its results in JSON format, containing a structured hierarchy of pages, blocks, lines, and words, along with the corresponding bounding box coordinates and recognized text content. In addition to this, when using the Key Information Extraction (KIE) module, the results are returned in a dictionary format, where each key represents a class label (such as "Name" or

"Date"), and the associated value contains the extracted predictions for that class on each page (Felix-T2K, 2023). However, DocTR does not appropriately include automatic detection and processing of tables. Instead, developers may use accredited third-party technology like OpenCV for table extraction. OpenCV is an open-source computer vision library that utilizes computer vision practices for detection of lines and contours that can be useful for identifying table structures from scanned images (Gaikwad, 2023). Overall, DocTR is a very powerful document-level OCR with understanding capabilities, particularly where layout understanding is important. Figure 2.4 illustrates the architecture of DocTR.

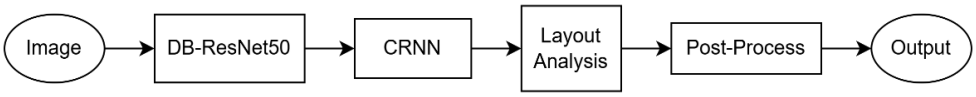

Figure 2.4: Architecture of DocTR Framework

2.3.5 DeepSeek-OCR

DeepSeek-OCR is a document oriented vision language model developed to improve OCR performance on visually complex documents. According to the official report (Wei, Sun and Li, 2026), the model is designed around a new encoder called DeepEncoder V2, which dynamically reorders visual tokens based on image semantics before they are passed to the language model decoder. This differs from conventional vision language models that usually process visual tokens in a fixed raster scan order from top-left to bottom-right. The authors argue that such fixed ordering does not reflect human visual reading behavior, especially when documents contain complex layouts, formulas, or tables.

As shown in Figure 2.5, the processing pipeline begins with dynamic tiling and pre-processing, where the image is prepared and divided into suitable visual regions. The image is then passed to the vision encoder, which is implemented using DeepEncoder V2 (Wei, Sun and Li, 2026). The report also states that visual tokens use bidirectional attention, while learnable query tokens use causal attention. This allows the model to progressively reorder visual information based on semantic relationships in the document.

A key contribution of DeepSeek-OCR is the reordering stage. Instead of preserving the original patch order, the encoder produces a new reading order guided by what the paper calls causal visual flow. Only the reordered causal flow tokens are then passed to the decoder. This design enables cascade causal aware visual understanding while keeping the model efficient.

The final stage is the LLM decoder, which interprets the reordered visual tokens and generates the textual output. According to the report, DeepSeek-OCR uses a DeepSeek-MoE decoder and preserves the image compression ratio and decoding efficiency of the earlier DeepSeek-OCR system (Wei, Sun and Li, 2026). Overall, this architecture shows that DeepSeek-OCR is particularly suitable for document OCR tasks where reading order and layout understanding are important.

The effectiveness of this visual causal flow is reflected in its benchmark performance, where DeepSeek-OCR achieves a CER of 2.30% and a WER of 5.60% (Skywork, 2026). Furthermore, it demonstrates high-level document understanding with an overall score of 91.09% on the OmniDocBench (Wei, Sun and Li, 2026).

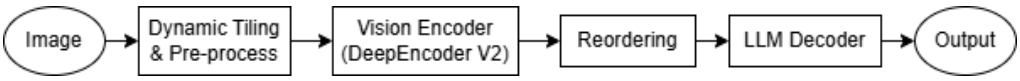

Figure 2.5: Architecture of DeepSeek-OCR Framework

2.3.6 Comparison of OCR Frameworks

Each OCR engine has its advantages and disadvantages, depending on the application requirements. In terms of accuracy, DeepSeek-OCR showed the strongest overall performance in this study, especially in CER and WER evaluation. PaddleOCR and DocTR are also deep learning-based OCR engines, but their performance was more affected by complex document layouts compared to DeepSeek-OCR. PaddleOCR uses the PP-OCRv5 pipeline which is highly optimized for Chinese and English, while DocTR applies a more document-focused recognition approach. EasyOCR also has capabilities across many languages and is fairly confident detecting text in natural images. Tesseract is still widely used, but it has limitations when it comes to distorted

text, low-resolution images, or uncommon fonts due to the fact that it does not utilize the full suite of modern deep learning based text detection.

When it comes to supported languages, EasyOCR is the most comprehensive with over 80 languages supported by default (Jaided, n.d.). Tesseract also supports a wide range of languages, however, Tesseract may require extra trained data files to be installed manually to support some languages. With regards to the currently lesser known and smaller application DocTR, it specifically supports far fewer languages, while currently it is specifically focused on supporting the English language and only a few others which makes it less suitable for multilingual processing.

Layout handling is another important issue, particularly when dealing with scanned documents which may include layout elements such as tables or forms, or documents with mixed content. PaddleOCR has an advantage in this instance as its PP-Structure module allows layout analysis, including table structure recognition and key-value extraction (Cui et al., 2025). However, DeepSeek-OCR has an advantage in complex document understanding because it uses visual token reordering and reading order understanding. DocTR also provides layout understanding by grouping detected text into paragraphs or blocks, but it does not support table recognition. EasyOCR lacks layout analysis features and focuses mainly on text detection and recognition. Tesseract can output word positions using hOCR format but does not interpret tables or forms natively (Wajer, 2023).

Other important factors include community support, documentation, and customizability. Tesseract has been around for a long time and has active and useful community support. EasyOCR is newer, but has a good userbase and is easy enough to use for simple tasks. PaddleOCR has official documentation to help and also has a rather large collection of models that have actually been pre-trained for different types of use cases. Although DocTR is powerful, it has fewer users and resources available, so it might need more technical skills to use or modify.

In summary, the choice of OCR engines depends on the user needs. In a simple example, basic text extraction, Tesseract or EasyOCR may be sufficient. For a complete OCR document processing solution where structured data such as tables and key-value information is required, PaddleOCR remains

a suitable choice. However, when the main requirement is highly accurate text extraction from complex or unstructured documents, DeepSeek-OCR is the strongest option. If the main requirement is document-level understanding of text, where a high degree of accuracy is needed and outputs are highly detailed, then DocTR is a reasonable choice. As shown in Table 2.1, the comparison of the OCR engines are listed.

Table 2.1: Comparison Table of Five OCR Engines (Mangal, 2025)

Feature	Accuracy	Language Support	Layout Handling	Model Type
Tesseract	High	100+	Basic	LSTM
EasyOCR	High	80+	None	CRAFT + CRNN + CTC
PaddleOCR	Very High	80+	Advanced	DB + CRNN + CTC
DocTR	Very High	Few only	Moderate	Transformer-based
DeepSeek-OCR	Very High	100+	Advanced	VLM + DeepSeek-MoE

2.4 Large Language Models (LLMs)

LLMs are advanced natural language processing systems capable of processing and generating natural, human-style language. They have been trained on extensive text corpora and can perform tasks requiring a human ability, such as summarizing, sentiment analysis, translating, answering questions and generating content (Matarazzo and Torlone, 2025). LLMs make predictions of the next word sequence based on prior words and their contexts, as well as leverage that prediction ability in order to perform tasks that do not require extensive programming (Sartori and Orrú, 2023). Moreover, recent models contain billions of parameters and are often refined using techniques like Reinforcement Learning from Human Feedback (RLHF) to achieve higher accuracy and produce responses that better match human judgment (Wang et al., 2022).

According to Ferraris et al. (2025), LLMs are based on a transformer architecture and are able to process all the words at once and learn relationship within words, like other deep learning architectures using an attention mechanism to infer relationships within the data. The various LLMs they discussed learn from billions of sentences during their training phase and build statistical relationships between words, phrases, and patterns across languages.

LLMs are used for many process and services, such as automated customer service, summarization of documents, review of legal documents, and as a conversational agent (Singh, 2024a). While LLMs have numerous benefits, there are also limitations. LLMs can produce errors when the context is unclear or when the model is presented with data that varies considerably from the training data. LLMs can also produce responses that seem reasonable, but are factually inaccurate, this occurrence is sometimes referred to as "hallucination" (Cleti and Jano, 2024). In addition, LLM performance is dependent on the training and fine-tuning data used. Despite these challenges, LLMs have considerably advanced the natural language processing capabilities of computers to better understand and deal with human communication.

2.5 Architectural Principles of LLMs

LLMs are primarily built on the transformer architecture, which was a breakthrough that changed how machines process sequential data (Ferraris et al., 2025). While traditional methods to sequentially analyze data include recurrent neural networks (RNNs) and long short-term memory networks (LSTM), transformers analyze input words in a different way which is to consider entire sentences or paragraphs of text at once using what is known as self-attention. Through using self-attention, a transformer can determine how words relate to one another no matter their positions in the sequence, which results in a better understanding of context (Li, 2024).

According to Tay et al. (2022), LLMs are built with multiple layers of transformer blocks, with each layer of a transformer consisting of two primary features which are multi-head self-attention layers and feed-forward networks. The multi-head self-attention layers compute how important each word is relative to every other word in the input text, while the feed-forward networks take those relationships and compute meaningful relationships of the input.

Once relationships from the input to some output have been computed, the transformer layer can predict the next word or token in a sequential manner. Positional encoding is another key aspect of LLMs. Tay et al. (2022) also describe that, positional encoding is a way to incorporate an understanding of the order of text, which is important to maintain meaning in what could be sequential ordering.

2.5.1 DeepSeek

DeepSeek is an open-source LLM that starts from the transformer architecture and employs certain optimizations to improve scalability and computational efficiency. Liu et al. (2024) state that one of the main design features is using a Mixture-of-Experts (MoE) design whereby only a certain number of the total parameters are activated per input. While full model could be hundreds of billions of parameters, only some portion is activated at any one time and this results in faster processing and reduced supply chain hardware.

Besides, according to the DeepSeek technical report (Liu et al., 2024), the model employs sparse attention with Multi-head Latent Attention (MLA), a mechanism designed to reduce the need to compute relationships between every token pair in a sequence. This enables more efficient sequence processing and allows the model to handle longer documents with lower computational requirements.

DeepSeek shows its performance across various benchmarks. In summarization tasks, DeepSeek achieved a Final Score of 0.5794, outperforming competitors such as OpenAI o3mini (0.4869) and Poro (0.3333). It demonstrated particularly strong results in cosine similarity of 0.8943 and BERT-based Semantic Similarity Score (BERTScore) of 0.6820 (Ataei, 2025). Additionally, in zero-shot evaluations, DeepSeek showed exceptional capability on the ASE dataset, securing top scores in Bilingual Evaluation Understudy with Representations from Transformers (BLEURT) of 0.587 and BERTscore of 0.746. On the ArgKP benchmark, it maintained competitive performance with a BERTScore of 0.769 (Li et al., 2025).

This architecture makes DeepSeek particularly strong in code generation, mathematical reasoning and understanding multilingual documents with improved contextual accuracy as evident from extensive training across

datasets and languages such as both Chinese and English. By integrating sparse attention and Mixture-of-Experts (MoE) with conventional transformer layers, DeepSeek offers an efficient and scalable model.

2.5.2 Llama

Llama is another open-source LLM that uses a dense transformer architecture, where all parameters are active during inference (Dubey et al., 2024). Parameters here mean the values the model has learned during training, and being "active" means that all of these values are used whenever the model reads text and produces an output, instead of only some of them.

Llama introduces several optimizations to improve the efficiency of transformer-based language models. GeeksforGeeks (2024) state that Llama leverages the technique of Grouped-Query Attention (GQA). In this approach, queries are divided into smaller groups and each group will share the attention scores instead of computing them separately (Singh, 2024b). This method optimizes the attention mechanism by reducing memory consumption and maintaining accuracy, particularly for long sequences. In terms of computational and memory efficiency, Dubey et al. (2024) also mentioned that Llama uses rotary positional embeddings (RoPE) to better preserve contextual relationships when handling longer sequences.

This dense transformer architecture makes Llama well-suited for tasks requiring stable language understanding. By keeping all parameters active during inference, Llama delivers consistent performance, though at a higher computational cost, making it a reliable choice for fine-tuning and integration in modular OCR–LLM pipelines.

2.5.3 Comparison of LLMs

When comparing DeepSeek and Llama, both models demonstrate strong natural language processing capabilities, but each model focuses on different priorities in its design. DeepSeek adopts a Mixture-of-Experts (MoE) approach and sparse attention mechanisms, focusing on computational efficiency and scalability (Liu et al., 2024). This allows DeepSeek to choose only a subset of parameters to be activated during inference, which is a huge advantage in situations of limited computing capacity or where extended context lengths are required.

In contrast, Llama employs a dense transformer architecture, where all parameters are active during inference. This increases the overall compute load of the model, particularly when processing very long contexts, but on the other hand, it ensures consistent performance across all tasks. Llama integrates optimizations for performance such as Grouped-Query Attention (GQA) and rotary positional embeddings (RoPE) to balance performance (Dubey et al., 2024).

In terms of development, both models are open-source. However, their optimization strategies are different which, DeepSeek prioritizes selective activation and long-context reasoning, whereas Llama focuses on robust, stable performance. These differences influence their suitability for integration into modular systems that combine OCR with LLMs, depending on whether the application prioritizes efficiency or output consistency. Table 2.2 summarizes the fundamental architectural distinctions between DeepSeek and Llama.

Table 2.2: Architectural Differences between DeepSeek and Llama

Feature	DeepSeek	Llama
Architecture	Sparse attention + Mixture-of-Experts	Dense transformer
Parameter Activation	Partial (selective activation)	Full activation during inference
Context Length	Supports extended context (up to 128k)	Moderate context length (varies by version)
Optimization Focus	Computational efficiency & scalability	Stable performance & fine-tuning

2.6 Evaluation Metrics

To evaluate the effectiveness of OCR and LLM systems, different metrics are used depending on the nature of the task. In OCR evaluation, the metrics focus on exact character-level and word-level recognition accuracy, as well as the correctness of extracted output. In contrast, LLM evaluation places greater emphasis on semantic similarity and contextual quality, since generated text does not always need to match the reference exactly word for word.

2.6.1 OCR Evaluation Metrics

For OCR systems, evaluation is commonly conducted using accuracy, precision, recall, F1-score, Character Error Rate (CER), and Word Error Rate (WER). These metrics are useful because they measure different aspects of OCR performance.

According to Leung (2021), CER measures the proportion of characters that are incorrectly recognized compared to the ground truth. It is calculated based on the total number of substitutions, deletions, and insertions divided by the total number of characters in the reference text. Therefore, CER is useful for showing how well an OCR system performs at the character level. The formula for CER is shown in Equation (2.1).

$$CER = \frac{S+D+I}{N} \tag{2.1}$$

where

S = number of substitutions

D = number of deletions

I = number of insertions

N = number of characters in ground truth text

Leung (2021) also state that WER complements CER by assessing errors at the word level, accounting for insertions, deletions and substitutions. WER is calculated similarly, but at the word level. The formula for WER is shown in Equation (2.2).

$$WER = \frac{S+D+I}{N} \tag{2.2}$$

where S, D, I follow the same definitions as in CER, and N is the number of words in the ground truth text.

While CER and WER capture how often recognition errors occur, they do not show how well the system distinguishes correct predictions from incorrect ones. For this reason, accuracy, precision, recall and F1-score are also included. Precision measures the proportion of correctly predicted text among

all predictions, recall evaluates how well the OCR can capture all the text present in the ground truth while F1-score represents the mean of precision and recall which is useful for balancing the trade-off between correctness and completeness (Kashyap, 2024). Accuracy is also included to measure the overall proportion of correct predictions, while precision, recall, and F1-score are derived from confusion-matrix terms such as true positives, false positives, false negatives, and true negatives (Obi, 2023). Equations (2.3) to (2.6) show the formulas for accuracy, precision, recall, and F1-score.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.3}$$

$$Precision = \frac{TP}{TP+FP} \tag{2.4}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.5}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{2.6}$$

where

TP = true positives

TN = true negatives

FP = false positives

FN = false negatives

Together, these metrics provide a more complete view of OCR performance by measuring both exact recognition quality and correctness of extraction. This combination supports a more robust and fair comparison of the five OCR engines used in the study.

2.6.2 LLM Evaluation Metrics

The evaluation of LLMs focuses on the quality of generated text using metrics that emphasize semantic similarity and contextual closeness. In this study, the LLM evaluation focuses on BERTScore, BLEURT, and cosine similarity, as

these metrics are more suitable for assessing semantic quality than exact lexical overlap.

BERTScore evaluates semantic similarity at the contextual embedding level (Zhang et al., 2019). It matches each token in the candidate text with the most similar token in the reference text using cosine similarity, then computes precision, recall, and F1-score based on those similarities (Zhang et al., 2020). This makes it suitable for measuring semantic preservation even when the wording differs.Because of this, BERTScore is useful for measuring semantic preservation in generated output. The F1 measure of BERTScore is shown in Equation (2.7) (Zhang et al., 2020).

$$F_{BERT} = 2 \times \frac{P_{BERT} \times R_{BERT}}{P_{BERT} + R_{BERT}} \quad (2.7)$$

where

F_{BERT} = F1 BERTScore

P_{BERT} = Precision Bert

R_{BERT} = Recall Bert

BLEURT serves as a reference based metric that goes beyond word level similarity by assessing fluency, adequacy, and contextual relevance. Unlike traditional lexical metrics, it does not rely on a simple overlap formula, instead, it uses a pre-trained and fine-tuned BERT-based model to produce a scalar quality score for a candidate sentence relative to a reference sentence (Sellam, Das, and Parikh, 2020). Therefore, BLEURT is more aligned with human judgment and is useful for evaluating the quality of generated responses.

In addition, cosine similarity is used to measure how close the semantic representation of generated text is to the reference text. In Natural Language Processing, it is commonly applied to text embeddings, including contextual embeddings produced by BERT-based models (Reimers and Gurevych, 2019), thus it is also known as Sentence-Bert. A higher cosine similarity indicates that the generated output is more similar in meaning to the expected result.

By combining BERTScore, BLEURT, and cosine similarity, the evaluation provides a suitable basis for assessing LLM output in terms of semantic quality, contextual preservation, and overall similarity of meaning.

2.7 Integration of LLM with OCR Engines

The integration of OCR with LLM appears to be a highly effective approach to improve text extraction from complex documents. Traditional OCR systems are effective at detecting characters but often fail when faced with challenging layouts, multilingual content or handwritten text. LLMs enhance this process by providing contextual understanding, correcting errors, and enabling advanced tasks such as summarization, translation and information structuring. Together, OCR and LLMs create a pipeline that transforms raw extracted text into more accurate and meaningful content.

2.7.1 Existing Solutions

One of the existing solutions is the LLM-centric pipeline proposed by Loukil et al. (2024), which was developed for automated invoice processing. In this approach, raw text is first extracted from invoice images using OCR engines such as Tesseract or the deep-learning-based DocTR, after which a pre-trained LLM refines the noisy output and structures it into a standardized JSON format. This process allows for accurate identification of key details including supplier information, total amounts and product tables across multipage invoices, with Pydantic employed to validate the structured results. The system was implemented using the OLlama API to run open-source models such as Llama3, Llama3-8B and Mistral-7B. The experimental findings highlighted that OCR quality had a greater impact on performance than model size, with smaller LLMs still achieving substantial improvements at lower computational cost. Despite these advantages, the paper acknowledges limitations of inaccuracies in information generation and its monomodal dependency on OCR text, which prevents it from leveraging visual layout information. To address these challenges, the authors suggested to include incorporating multi-modal LLMs capable of directly interpreting invoice images and enhancing prompt design with layout information for instruction-tuned models to improve performance.

Another relevant solution is the OCR-augmented GPT pipeline introduced by Park et al. (2025), which focuses on addressing the difficulties of text recognition in industrial environments where handwritten notes, engravings or environmental noise often hinder accuracy. In this system, OCR engine such as PP-OCR is first used to extract raw text from image, which are then passed to a GPT model that applies contextual reasoning and prompt guidance to refine errors and determine the most reliable transcription. An optional feedback loop process further enhances adaptability by allowing human input to be incorporated without retraining, which in turn improves the contextual guidance applied within the GPT stage. The primary metric used for evaluation was the CER, where a lower score indicates better performance. Despite its effectiveness, the study acknowledges two key limitations which are the system struggles in cases of extreme text degradation where OCR fails to provide meaningful input, and its relatively high computational cost that could be a barrier for high-volume, high-speed industrial applications.

A further study by Gupta et al. (2024) examines the integration of OCR and LLMs for handwritten document processing. In this work, EasyOCR is used to detect and extract handwritten text, which is then refined using an LLM (Google's Gemini 1.5 Pro). The evaluation showed that the LLM-based approach consistently outperformed OCR in both accuracy and speed, producing more coherent text outputs and completing tasks significantly faster, even as the number of input images increased. However, despite these advantages, the study highlights important limitations which LLMs are resource-intensive, requiring high computational power and memory for inference. Moreover, their processing times can scale poorly with larger datasets, raising concerns for real-time or high-volume applications and their effectiveness declines when dealing with rare or complex scripts outside their training scope.

2.7.2 Gap of study

The current studies on OCR and LLM integration show clear improvements in text recognition and post-processing. However, several technical gaps remain unresolved. First, most existing solutions rely on a single OCR engine, which limits their adaptability when processing documents with different languages,

layouts, or handwriting styles. Second, these systems often follow a fixed pipeline design, making it difficult to replace or upgrade individual components such as the OCR module or the language model. Third, many of the reviewed approaches do not support real-time interaction or user feedback, reducing their ability to adapt to changing requirements. Lastly, the use of large LLMs increases computational demands, creating challenges for high-volume or time-sensitive applications.

To address these limitations, this study proposes a modular integration strategy that evaluates multiple open-source OCR engines, including Tesseract, EasyOCR, PaddleOCR and DocTR, combined with open-source LLMs such as DeepSeek and Llama. The system is designed to allow each module to be upgraded or replaced independently while maintaining overall functionality. It also supports advanced tasks including summarization, translation and structured information extraction, and will be deployed as a web-based application supporting Chinese, Malay and English documents.

2.8 Summary

This chapter explored the key technologies and related studies that support this research. It examined both traditional and modern OCR approaches, highlighting how deep learning-based engines such as PaddleOCR and DocTR outperform older rule-based systems in handling complex layouts and multilingual documents. This review also covered the architecture and capabilities of LLMs and their role in refining OCR outputs through contextual interpretation and advanced text processing. In addition, evaluation metrics such as accuracy, precision, recall, F1-score, CER, WER, BERTScore, BLEURT, and cosine similarity were reviewed to establish a suitable basis for assessing OCR and LLM performance. Analysis of existing OCR–LLM integrations revealed that while they improve accuracy and automate information extraction, most remain limited by fixed pipelines, single-engine dependence and high computational requirements. These insights guided the identification of research gaps and shaped the modular integration strategy adopted in this project.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

This chapter presents the design and implementation of the proposed OCR-LLM system, detailing the architectural framework, integration of key components and development of the web-based interface. The system adopts a modular architecture to ensure flexibility, scalability and maintainability, allowing each layer, ranging from image input and pre-processing to OCR, AI post-processing and language model refinement, to operate independently while contributing to the overall workflow. Five OCR engines (Tesseract, EasyOCR, PaddleOCR, DocTR and DeepSeek-OCR) are integrated for comparative evaluation across different document types in order to select the best performing OCR engine for the proposed system, while two LLMs (DeepSeek and Llama) are integrated for comparative evaluation in order to select the most suitable model for contextual refinement and chat-based interaction. To ensure usability, the entire pipeline is deployed through a Streamlit web application that supports user interaction and document management.

3.2 System Architecture

The proposed system was designed with a modular architecture to maximize flexibility and maintainability. At a high level, the architecture consisted of several interconnected layers that worked together to process physical or scanned documents and convert them into meaningful digital text.

The workflow began with the input module, where users uploaded scanned images or PDF documents. These files were passed to the pre-processing module, which prepared the images for OCR by applying basic image enhancement techniques such as resizing, noise reduction, and binarization. These steps were crucial for improving the quality of text recognition, particularly when dealing with documents that were scanned in poor lighting or low resolution.

Next, the prepared images entered the OCR engine layer, where one of the five selected engines (Tesseract, EasyOCR, PaddleOCR, DocTR, or

DeepSeek-OCR) was applied to extract raw text. Each OCR engine operated independently, and the system allowed comparison between them to assess their performance on various document types and select the best-performing OCR engine for the proposed system. After text extraction, the results proceeded to the AI post-processing module, where the text was further refined to improve readability and contextual quality.

After OCR extraction, the output was passed to the LLM layer. In this study, two LLMs, DeepSeek and Llama (both 8-billion-parameter versions), were integrated using the Ollama framework for comparative evaluation. The LLM layer supported both chat interaction and AI agent post-processing. In the chat function, the model generated responses based on the extracted text, while in the AI agent stage, it refined the OCR output to improve readability and contextual quality. Based on the evaluation results, the most suitable LLM was selected for the final implementation of the system.

Finally, the processed text was presented to users through a web interface module, where they could view, download, or further interact with the extracted content. Figure 3.1 shows the overall system architecture.

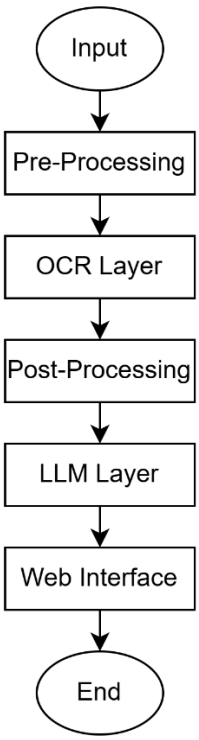

Figure 3.1: System Architecture Diagram

3.3 Dataset and Document Layout Categories

The dataset used in this study consisted of 500 images prepared for OCR evaluation. To assess OCR performance under different levels of layout complexity, the documents were grouped into two layout categories, which are single-column layout and unstructured layout.

The single-column layout contained text arranged in a regular top-to-bottom reading order with cleaner alignment and simpler structure. In contrast, the unstructured layout contained more complex arrangements, such as multiple text blocks, mixed alignment, embedded images of different sizes and less predictable reading order. The dataset used in this study was taken from a Chinese newspaper clipping sourced from the David Chen Electronic Library, which contains a large collection of newspaper clippings and related materials. Examples of both layout categories are shown in Figure 3.2.

These two layout categories were selected because they represented different levels of OCR difficulty. By evaluating OCR engines across both layout types, the study was able to assess not only extraction performance, but also the robustness of each OCR engine under more realistic and complex document conditions.

(a) (b)

Figure 3.2: Examples of (a) Single-column Layout and (b) Unstructured Layout

3.4 OCR Engine Integration

The system incorporated five OCR engines, which are Tesseract, EasyOCR, PaddleOCR, DocTR, and DeepSeek-OCR. These engines were chosen based on main criteria of open-source nature for academic use, and their active development communities that ensured continued updates and support.

The integration of these OCR engines was carried out through their respective Python libraries using custom scripts. Tesseract was accessed via the *pytesseract* wrapper, where an image was loaded using the library and processed with *image_to_string()*. EasyOCR was integrated through the *easyocr* package, with a reader initialized for multiple languages and applied to image files for extracting text along with confidence scores. PaddleOCR, built on the *PaddlePaddle* deep learning framework, was implemented using the *PaddleOCR* class with configurations to disable unnecessary orientation and document unwarping for streamlined processing. DocTR was employed using its *ocr_predictor* model, loading input images through the DocumentFile API and exporting results in structured dictionary format for text line iteration. DeepSeek-OCR was integrated through its OCR inference framework and was applied to uploaded document images to generate extracted text output for comparative evaluation against the other OCR engines.

Each OCR engine was implemented as a separate module within the system, and its performance was evaluated using standard metrics such as accuracy, precision, recall, F1-score, CER, and WER. Based on the evaluation results, the best performing OCR engine was selected for the proposed system. After the selected engine extracted the text, the output was passed to the AI post-processing module for further refinement before being sent to the LLM layer.

3.5 LLM Integration

To enhance the extracted text beyond simple recognition, two LLMs which are DeepSeek and Llama, were incorporated into the system and evaluated for performance. Both models were deployed through the Ollama platform, which enabled local hosting and execution of LLMs with manageable resource requirements. This project utilized the 8 billion parameter versions of these models to ensure an effective balance between computational efficiency.

The primary function of the LLM layer was to support chat interaction using the text produced by the selected OCR engine. The LLM was used to help users interact with the extracted document content in a more contextual and meaningful way. Through the chat interface, users could ask questions related to the uploaded page, and the LLM generated responses based on the OCR output. This allowed the extracted text to be used not only for viewing, but also for interactive understanding and user assistance.

Both models were evaluated using quantitative metrics, namely BLEURT, BERTScore, and cosine similarity. Based on these evaluation results, the model that delivered the best performance was selected for the final implementation of the system, ensuring that users benefited from the highest possible quality of text interpretation and enhancement. The integration involved sending the post-processed text as input to the selected LLM through Ollama, using prompts designed to guide the model in generating consistent and contextually relevant responses. Since both DeepSeek and Llama were used with the same parameter scale, their outputs could be compared fairly to analyze which model offered better contextual correction and text enhancement when working with OCR output.

3.6 AI Agent Post-Processing

After OCR extraction, an AI agent post-processing step was applied to refine the extracted text before it was used in the final system. This step was implemented to improve the readability and contextual quality of the OCR output, especially when the extracted text contained spacing issues, formatting inconsistencies, or OCR-related errors.

In the proposed system, the AI agent post-processing function was represented through the Autocorrect feature in the web application. After the OCR result was displayed, the extracted text could either be manually edited by the user or refined automatically through the AI-based post-processing step. This allowed the output to become cleaner and more suitable for further interaction.

In this project, the AI agent post-processing module was implemented using the Strands Agents framework. Strands Agents is an open-source Python framework that supports the development of model-driven AI agents (AWS,

2025). In this system, the Strands Agent was used specifically as an OCR token correction assistant rather than a general text rewriting tool.

A strict system prompt was defined to control the behaviour of the agent. The agent was instructed to correct only obvious one-token OCR mistakes caused by visual character confusion, such as replacing “qu1ts” with “quits”, “K1ta” with “Kita”, “NOVEM8ER” with “NOVEMBER”, “W0NG” with “WONG”, and “IMDB” with “1MDB”. The agent was not allowed to rewrite phrases or sentences, guess meaning, improve grammar, expand abbreviations or names, or change dates, metadata, and formatting.

To reduce uncontrolled modification, the extracted OCR text was divided into smaller chunks before being processed by the agent. For each chunk, the agent was required to return only a valid JSON object containing corrections in the form of incorrect-token and corrected-token pairs. If no obvious OCR mistakes were detected, the agent returned an empty JSON object. The working flow of the AI agent is described as follows:

1. The OCR engine extracts raw text from the uploaded document.
2. The extracted text is divided into smaller chunks.
3. Each chunk is passed to the Strands Agent for analysis.
4. The agent identifies obvious one-token OCR errors based on the predefined prompt rules.
5. The agent returns the detected corrections in JSON format.
6. The system reads the JSON object and applies the corrections to the original OCR text.

The AI agent post-processing stage was then evaluated by comparing the OCR-only pipeline with the OCR + AI agent pipeline. The comparison was carried out using the output generated from the selected OCR engine to observe whether the refinement step improved the quality of the extracted text under different language settings.

3.7 Web Application Development

To make the system accessible and practical, it was implemented as a web-based application using Streamlit, which was highly suited for rapid development of machine learning and OCR-driven applications. Streamlit was chosen because it enabled developers to create interactive web applications entirely in Python

(Sayed, 2024), eliminating the need for separate HTML, CSS and JavaScript coding. This made it ideal for quickly integrating OCR and LLM outputs, allowing users to upload documents, extract text, refine the OCR output, and interact with the extracted content through the chat function.

During the selection process, three options were considered which were Streamlit, Flask and Solara. Flask provided full flexibility and control over backend and frontend development but required manual implementation of web components, which increased complexity and development time (Bhadra, 2022). Solara was a Python library for building web applications that allowed developers to create apps entirely in Python, while also offering React-style state management with reactive variables that updated automatically across the app, reusable components for building applications, and support for standalone web apps (Kop, 2023). Streamlit, on the other hand, offered the simplest learning curve, seamless integration with Python workflows and rapid prototyping capabilities (Folder IT, 2025). Since this project prioritized OCR, AI post-processing, and LLM integration rather than complex UI customization, Streamlit was selected as the most practical solution. Table 3.1 shows the comparison of different web application frameworks.

Table 3.1: Comparison of Web Application Frameworks

Framework	Ease of Development	UI Complexity	Learning Curve	Best For
Streamlit	Easy	Built-in	Low	Rapid ML/OCR prototypes
Flask	Moderate	Manual HTML/CSS	Medium	Full-featured custom web apps
Solara	Easy	Component-based	Medium	Interactive dashboards, new projects

3.8 Software and Environment

The development and evaluation of the system require a consistent software environment to ensure reproducibility and compatibility. Table 3.2 summarizes the programming languages, libraries, frameworks and platforms used in the implementation and testing of the OCR and LLM application.

Table 3.2: Software and Environment Setup

Component	Tool / Library	Version / Platform
Programming Languages	Python	3.11
OCR Engine 1	Tesseract	5.3.4
OCR Engine 2	EasyOCR	1.7.2
OCR Engine 3	PaddleOCR	3.1.0
OCR Engine 4	DocTR	1.0.0
OCR Engine 5	DeepSeek-OCR	Version 2
Local LLM	Ollama (Llama, DeepSeek)	Ollama 0.5.3, models: Llama3.1:8b, DeepSeek-r1:8b
AI Agent	Strands Agents	Python SDK
Frameworks	PyTorch, OpenCV	2.8, 4.12
Visualization	Matplotlib, Tabulate	3.10.5, 0.9.0
Web Application	Streamlit	Latest release
Platform	Google Colab	T4 GPU

3.9 Performance Evaluation

Different metrics were used for OCR, LLM, and AI post-processing performance. OCR evaluation focused on both correctness of extraction and exact text matching, while LLM evaluation focused on semantic similarity between the generated output and the reference text.

For OCR evaluation, accuracy, precision, recall, F1-score, CER, and WER were used. These metrics were selected to provide a balanced assessment of OCR performance across different document layout categories. Accuracy,

precision, recall, and F1-score were used to measure the correctness of the extracted output under the selected evaluation setup, while CER and WER were used to measure how closely the OCR output matched the ground truth text at the character and word levels.

For LLM evaluation, BLEURT, BERTScore, and cosine similarity were used. These metrics were selected because they are suitable for measuring semantic similarity and contextual closeness between generated output and reference text. The selected metrics were used to compare DeepSeek and Llama in order to determine the more suitable model for chat interaction in the proposed system.

For AI post-processing evaluation, the performance of the OCR-only pipeline and the OCR + AI post-processing pipeline was compared using accuracy, precision, recall, F1-score, CER, and WER. These metrics were used to observe whether the refinement step improved the extracted output in terms of correctness, while also identifying its effect on exact text matching.

3.10 Gantt Chart

Figure 3.3 outlines the 28 weeks project timeline where Final Year Project 1 (FYP 1) focuses on initial research and preliminary evaluation, while Final Year Project 2 (FYP 2) covers system integration, development, and final testing.

Figure 3.3: Gantt Chart

3.11 Summary

This chapter presented the methodology used in the development and evaluation of the proposed OCR-LLM system. It described the system architecture, dataset and document layout categories, OCR and LLM integration, AI agent post-processing, web application development, software environment, and performance evaluation. Five OCR engines and two LLMs were integrated for comparative evaluation in order to select the most suitable models for the proposed system. In addition, an AI agent post-processing step was included to refine OCR output before it was used in the final system. This chapter also outlined the Streamlit web application and the evaluation metrics used for OCR, LLM, and AI post-processing analysis. Overall, the methodology supported the development and evaluation of the proposed OCR-LLM system.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the results and discussion of the proposed OCR-LLM system. The evaluation focuses on three main components, which are OCR performance, LLM performance, and AI agent post-processing performance. In addition, the chapter also presents the interface and functionalities of the developed web application.

4.2 OCR Performance Analysis

This section presents the performance of the selected OCR engines used in this project. The evaluation was carried out on two types of layouts, which are single- column and unstructured layouts. The purpose of this evaluation was to compare how well each OCR engine performed when extracting text from documents with different levels of complexity. For each metric, the final value shown in the results represents the average performance across English, Malay, and Chinese. The detailed breakdown of performance by languages are provided in Table A-1 and Table A-2 in Appendix A.

4.2.1 CER and WER Analysis

The CER and WER results in Figures 4.1 and 4.2 show that DeepSeek-OCR achieved the best performance across both layout categories. It recorded the lowest CER and WER for both single-column and unstructured documents, indicating that its output was closest to the ground truth text at both the character and word levels. In contrast, DocTR recorded the highest CER, especially for unstructured documents, making it the weakest engine in terms of character-level transcription fidelity.

For single-column documents, PaddleOCR also produced relatively low CER and WER, showing that it performed well when the layout was simple and regular. However, its error rates increased much more clearly for unstructured documents, while DeepSeek-OCR remained comparatively stable. This is an important comparison because it shows that PaddleOCR was strong

in easier layouts, but DeepSeek-OCR was more robust when the document structure became more complex.

A major trend can also be observed in WER, where most OCR engines produced higher error rates for unstructured layouts than for single-column layouts. This suggests that word-level transcription became more difficult when the reading order was less regular. For WER, Chinese was excluded because word-level comparison is less fair when the language does not explicitly separate words using spaces. Even with this exclusion, the overall pattern remained clear.

Overall, the CER and WER results show that DeepSeek-OCR was the strongest OCR engine in exact text preservation, while DocTR was the weakest, particularly under more complex layouts. This is important because lower CER and WER indicate fewer spelling mistakes, substitutions, and word-sequence errors, making the OCR output more suitable for later stages such as LLM refinement and chat interaction for users.

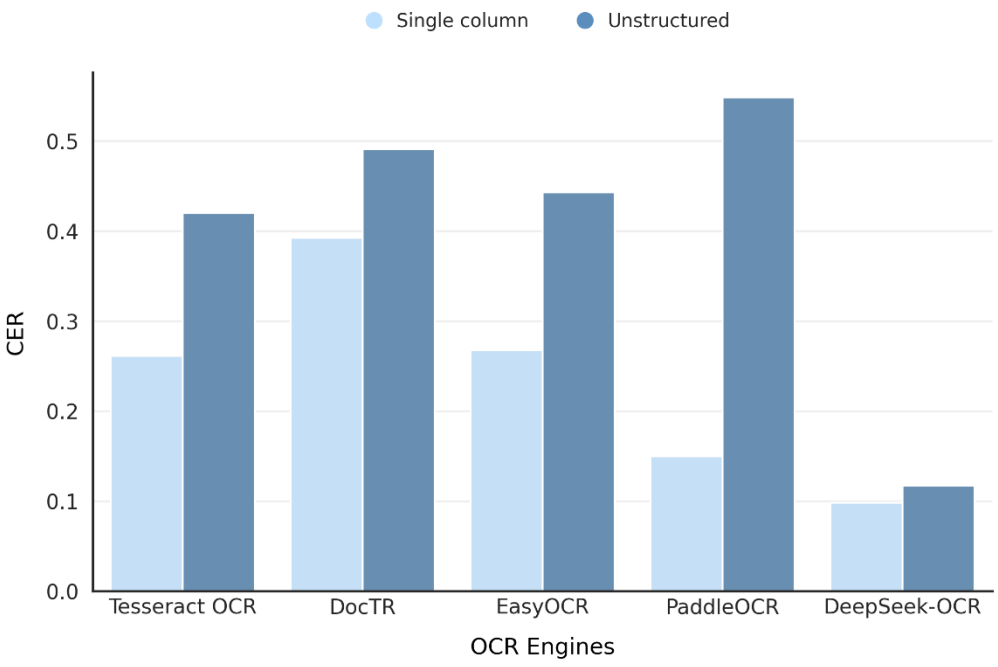

Figure 4.1: CER Comparison of OCR Engines

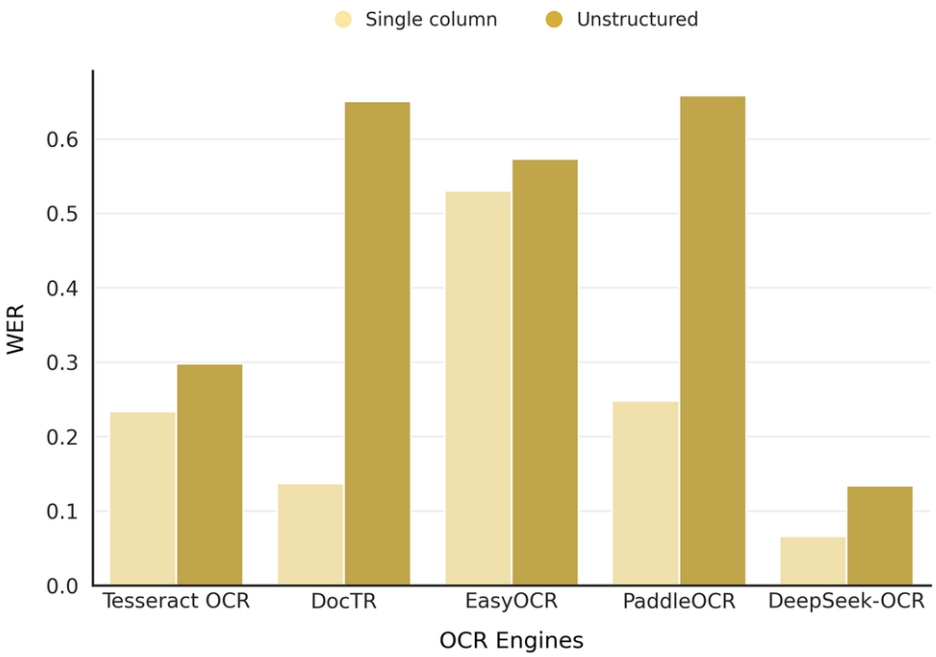

Figure 4.2: WER Comparison of OCR Engines

4.2.2 Accuracy, Precision, Recall, and F1-score Analysis

Based on Figures 4.3 to 4.6, the results show that DeepSeek-OCR and PaddleOCR were the strongest OCR engines overall, although their strengths differed across layout types. For single-column documents, DeepSeek-OCR achieved the highest accuracy at about 90%, while PaddleOCR obtained the highest precision, recall, and F1-score. This indicates that both engines performed strongly on simpler layouts, but in different ways.

For unstructured documents, DeepSeek-OCR clearly became the best performing engine, achieving the highest accuracy at around 84% and the strongest precision and F1-score, while also maintaining relatively high recall. By contrast, the accuracy of DocTR and PaddleOCR dropped, which shows a major gap between DeepSeek-OCR and the weaker engines when layout complexity increased. This is one of the most important comparisons in the results because it highlights the stronger robustness of DeepSeek-OCR across both layout types.

Among the other OCR engines, EasyOCR showed a smaller drop in recall, suggesting that it remained relatively capable of capturing text under different layouts. However, its overall scores were still lower than DeepSeek-OCR. Tesseract showed fair and balanced performance, but it did not achieve

the best score in any of the grouped metrics. DocTR generally remained the weakest, especially for unstructured documents, where both its accuracy and F1-score were low.

The results also indicate that recall behaves differently from accuracy and precision. An OCR engine may retrieve more text, but the output may still contain errors or incorrect grouping. Therefore, recall should be discussed together with precision and F1-score rather than being interpreted alone, since F1-score provides a balanced measure of completeness and correctness (scikit-learn, 2026).

Overall, the accuracy, precision, recall, and F1-score results indicate that DeepSeek-OCR offered the most consistent balance between correctness and robustness, while PaddleOCR was the strongest competitor in single-column layouts but showed a larger decline in more complex layouts.

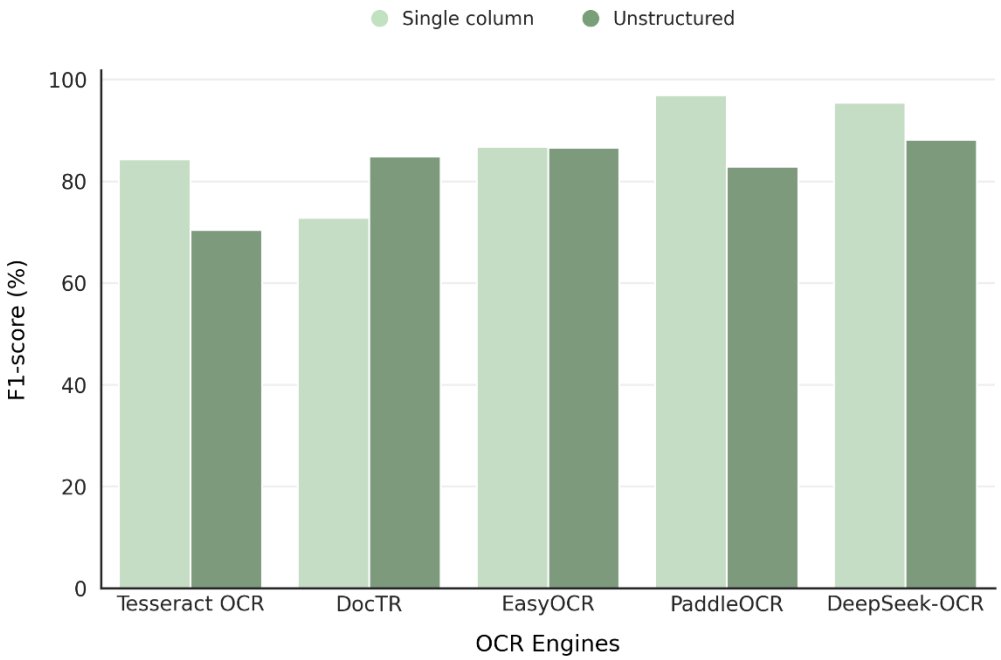

Figure 4.3: F1-score Comparison of OCR Engines

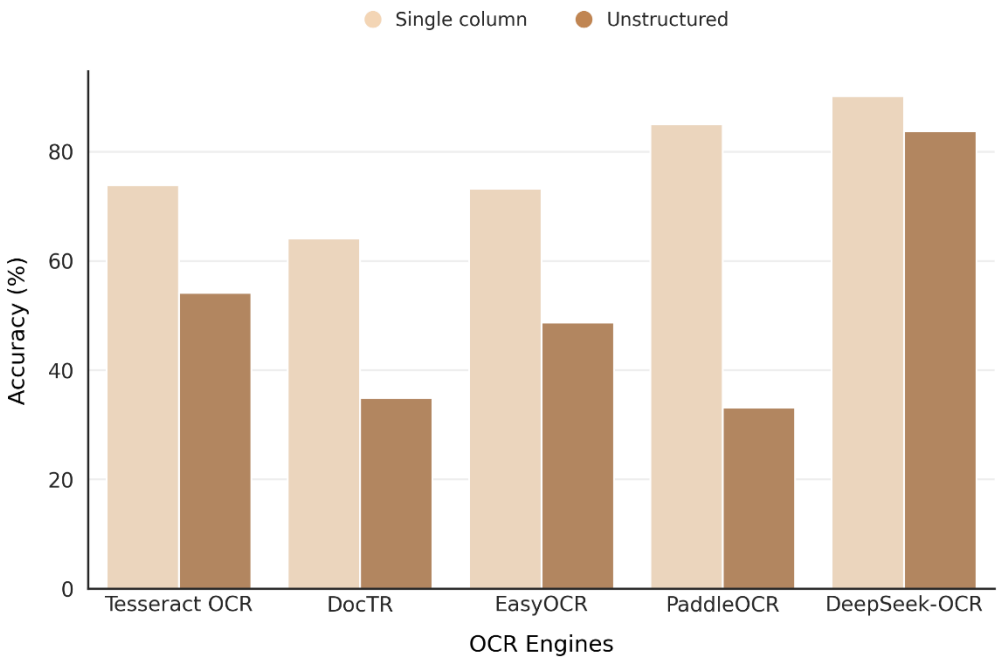

Figure 4.4: Accuracy Comparison of OCR Engines

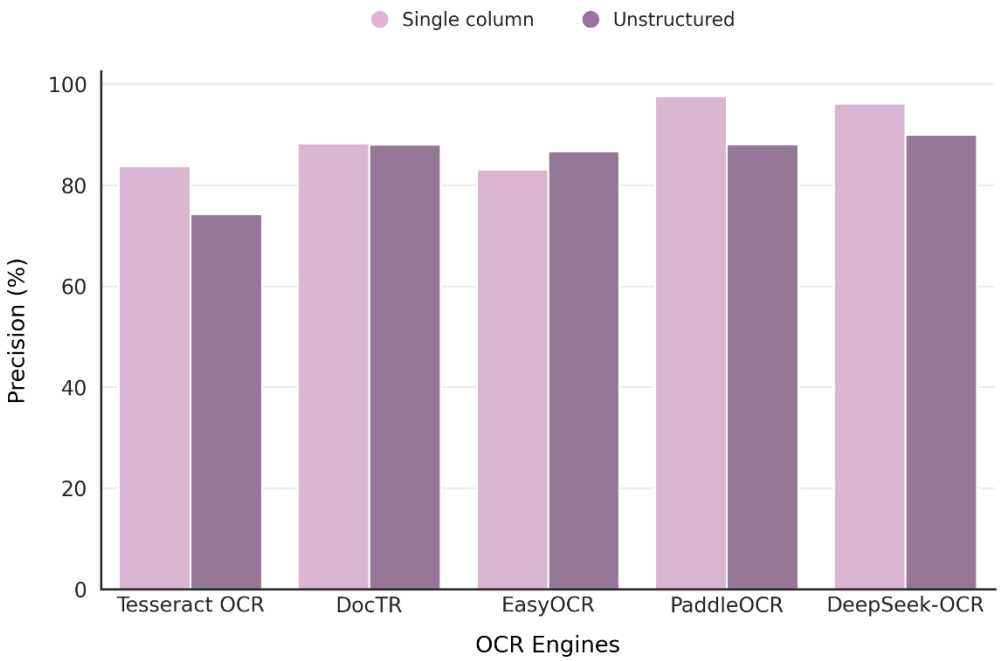

Figure 4.5: Precision Comparison of OCR Engines

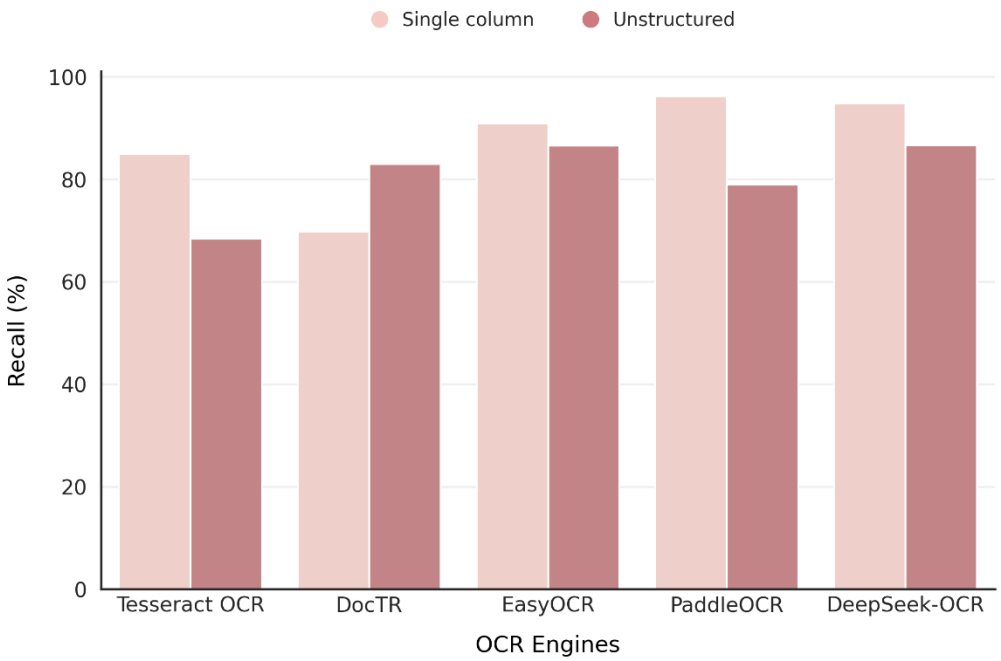

Figure 4.6: Recall Comparison of OCR Engines

4.2.3 Overall Comparison of OCR Engines

Based on the overall results presented in Figures 4.1 to 4.6, DeepSeek-OCR was the best performing OCR engine in this study. It achieved the highest scores in accuracy, precision, recall, and F1-score, while also producing the lowest CER and WER values across both single-column and unstructured document layouts In comparison, PaddleOCR also showed strong performance, especially for single-column documents. It achieved very high precision, recall, and F1-score in structured layouts. However, its performance dropped more obviously for unstructured documents, especially in terms of accuracy, CER, and WER. This means PaddleOCR is effective when the layout is simple, but less stable when document complexity increases.

Meanwhile, Tesseract showed fair and balanced results. It did not achieve the best result in any metric, but it performed more consistently than some other engines. This makes it a useful baseline OCR engine, especially for simpler documents.

Regarding the remaining engines, EasyOCR produced moderate results. Its recall and F1-score on unstructured documents were relatively acceptable, but its error rates were still higher than DeepSeek-OCR. Similar to DocTR, which had mixed performance where it showed high precision and recall in

simpler layouts, but the low accuracy and high CER and WER when dealing with unstructured layouts suggest that the overall quality of the extracted text was less reliable.

From these findings, it can be concluded that DeepSeek-OCR is the most suitable OCR engine for the proposed system. The engine not only performed well on regular documents but also maintained strong results on more challenging unstructured layouts. Since this project aims to process different layout types in a web application, robustness across layouts is an important requirement.

This finding is also consistent with the benchmark discussion presented in Chapter 2. DeepSeek-OCR-2 has been reported to perform strongly in exact text recognition tasks, with benchmark values of 2.30% CER and 5.60% WER under general benchmark settings. In this study, DeepSeek-OCR also achieved the best overall OCR performance across the selected evaluation metrics. This further supports the suitability of DeepSeek-OCR-2 as the OCR engine for the proposed system.

4.3 LLM Performance Analysis

This section presents the performance of the LLMs used in this project. The main purpose of this evaluation is to compare two open source LLMs which are Llama 3.1 and DeepSeek r1, in order to determine which model is more suitable for the implementation of the chat function in the proposed web application. To ensure consistency in the evaluation, the LLM comparison was conducted using text produced by DeepSeek-OCR, as it was identified as the best performing OCR engine in the OCR evaluation.

4.3.1 Comparative Analysis of LLMs Performance

The comparison between Llama and DeepSeek is shown in Figure 4.7. Based on the figure, DeepSeek achieved higher scores than Llama in all three evaluation metrics, which are BLEURT, BERTScore, and cosine similarity.

For BLEURT, DeepSeek obtained a score of about 81%, while Llama obtained around 62%. This is the largest gap among the three metrics. The result suggests that DeepSeek generated responses that were more semantically appropriate and closer to the reference text. Since BLEURT focuses on text

quality and meaning, this shows that DeepSeek performed better in producing more natural and meaningful output.

For BERTScore, both models showed strong performance, but DeepSeek still achieved a slightly higher result, around 94%, compared to Llama at around 90%. This means both models were able to preserve important contextual meaning, but DeepSeek was still more accurate in matching the reference content.

For cosine similarity, DeepSeek again achieved a higher value, around 90%, while Llama achieved around 81%. This indicates that the semantic representation of DeepSeek was closer to the expected result. In simple terms, DeepSeek generated responses that were more similar in meaning to the target output.

The trend shown in Figure 4.7 indicated that DeepSeek consistently outperforms Llama across all evaluation metrics. This suggests that DeepSeek is more effective in handling text extracted by OCR and refining it into output that is more understandable and contextually correct. Llama still showed good performance, especially in BERTScore, but its lower BLEURT and cosine similarity indicate that it was less consistent in preserving the intended meaning.

This finding is also consistent with the benchmark discussion presented in Chapter 2, where in one benchmark setting, DeepSeek obtained an average cosine similarity of 0.8943, an average BERTScore of 0.6820, and a final score of 0.5794. In another benchmark setting, DeepSeek-R1 achieved a BLEURT value of 0.587 and a BERTScore value of 0.746, while Llama 3.1 obtained 0.524 and 0.721, respectively. Therefore, this suggests that the stronger performance of DeepSeek observed in this study is broadly consistent with benchmark findings, although the performance may still vary depending on the dataset, language setting, and evaluation method.

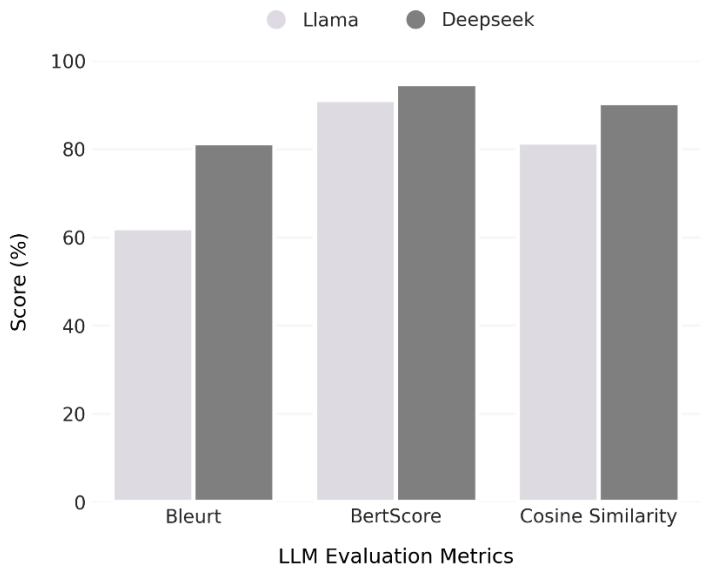

Figure 4.7: Comparative Evaluation of Llama and DeepSeek

4.4 AI Agent

This section presents the performance of the AI agent in the proposed pipeline. The evaluation was conducted to determine whether adding the AI agent after OCR could improve the overall output quality compared to using OCR alone. In this study, the AI agent was evaluated using text produced by DeepSeek-OCR, since DeepSeek-OCR was identified earlier as the best performing OCR engine. The comparison focused on unstructured document layouts in English, Malay, and Chinese to examine the performance of the AI agent.

4.4.1 AI Agent Performance Analysis

Based on Table 4.1, the AI agent improved the performance for English and Malay in most of the metrics. For English, the accuracy increased from 88.70% to 91.83%, while precision, recall, and F1-score also improved. A similar pattern can be observed for Malay, where all four metrics recorded clear improvement after the AI agent was applied.

The results suggest that the AI agent was able to improve the usefulness of the OCR output for these two languages. The increase in recall and F1-score is especially meaningful, as it shows that the refined output became more complete while still maintaining strong correctness. This indicates that the AI agent was effective in improving the semantic quality of the extracted text. For

Chinese, the improvement was not observed in the same way. As shown in Table 4.1, the accuracy decreased from 78.20% to 73.60%, while precision and F1-score also showed a slight decrease. Recall increased slightly, but the overall result was still weaker than the OCR-only pipeline.

These findings show that the AI agent was beneficial for English and Malay, but its performance was less stable for Chinese. This suggests that the AI agent was less effective for Chinese under the current implementation. A possible reason is that Chinese remains a challenging setting in open-source models, especially across different capability and evaluation dimensions (Cai et al., 2024). Furthermore, Chinese text does not explicitly separate words using spaces like English and Malay, it remains an important challenge in Chinese text processing (Chen et al., 2025). Therefore, even small changes introduced during post processing may lead to larger differences in the final evaluation. As a result, the lower performance for Chinese may be caused by both model limitations and the structural characteristics of Chinese text.

Table 4.1: Comparison of OCR-only and OCR + AI Agent Performance

Language	Pipeline	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
English	OCR-only	88.70	94.72	90.65	92.58
	OCR + AI Agent	91.83	96.80	94.94	95.83
Malay	OCR-only	84.44	91.09	87.10	89.01
	OCR + AI Agent	90.70	96.29	95.19	95.72
Chinese	OCR-only	78.20	84.03	82.23	83.04
	OCR + AI Agent	73.60	82.16	83.41	82.47

4.4.2 AI Agent Trade-Offs

Although the AI agent improved the accuracy, precision, recall, and F1-score for English and Malay, Table 4.2 also shows that both CER and WER increased after the agent was applied. This means that the refined output became less similar to the original ground truth text at the exact character and word levels.

This difference can be explained by the fact that accuracy, precision, recall, and F1-score were evaluated based on token overlap, while CER and

WER were based on edit distance. Therefore, the refined output could contain more meaningful and correctly matched tokens, which improved precision, recall, and F1-score. However, because the AI agent may semantically rewrite, rephrase, restructure, add, or remove words from the OCR output, the refined text may no longer preserve the original wording exactly. As a result, the output became easier to understand and more contextually meaningful, but it differed more from the ground truth text at the character and word levels, leading to higher CER and WER.

Furthermore, a larger increase in WER compared to CER was observed, which can be explained by the sensitivity of word-level evaluation. CER measures changes at the character level, so small spelling or character differences may only cause a slight increase. In contrast, WER treats a whole word as incorrect if the word is inserted, deleted, substituted, or reordered. Therefore, when the AI agent rewrites a sentence by adding words, removing words, or changing the word order, the word-level difference becomes much larger even if many individual characters remain correct. This explains why CER only increased slightly, while WER increased more significantly after AI agent post-processing.

In summary, the AI agent introduced a trade-off between improved output quality and exact text matching. Although CER and WER increased, the improvement in accuracy, precision, recall, and F1-score suggests that the output became more useful, which supports the objective of making OCR output more contextual using LLM.

Table 4.2: Comparison of CER and WER for AI Agent Performance

Language	Pipeline	CER	WER
English	OCR-only	0.05	0.11
	OCR + AI Agent	0.08	0.37
Malay	OCR-only	0.06	0.16
	OCR + AI Agent	0.09	0.39
Chinese	OCR-only	0.25	0.22
	OCR + AI Agent	0.29	0.82

4.5 Web Application Interface and Functionalities

This section presents the interface design and main functionalities of the developed OCR-LLM web application. The system was built to allow users to upload document images or PDF files, extract text using OCR, refine the extracted output, and interact with the content through a chat function.

4.5.1 User Interface Overview

The overall interface of the developed OCR-LLM dashboard is shown in Figure 4.8. The web application was designed with a simple and user-friendly layout so that users can perform document upload, OCR processing, text refinement, and chat interaction in a single interface.

As shown in Figure 4.8, the dashboard is divided into several main sections. These include the file upload area, document preview panel, extracted result panel, action buttons, and chat interaction section. The arrangement of these components allows users to follow the workflow more easily, starting from document input and continuing to text extraction, refinement, and interaction with the extracted content. Figure 4.8 also shows that the interface integrates OCR extraction, text refinement, and chat interaction into a single dashboard.

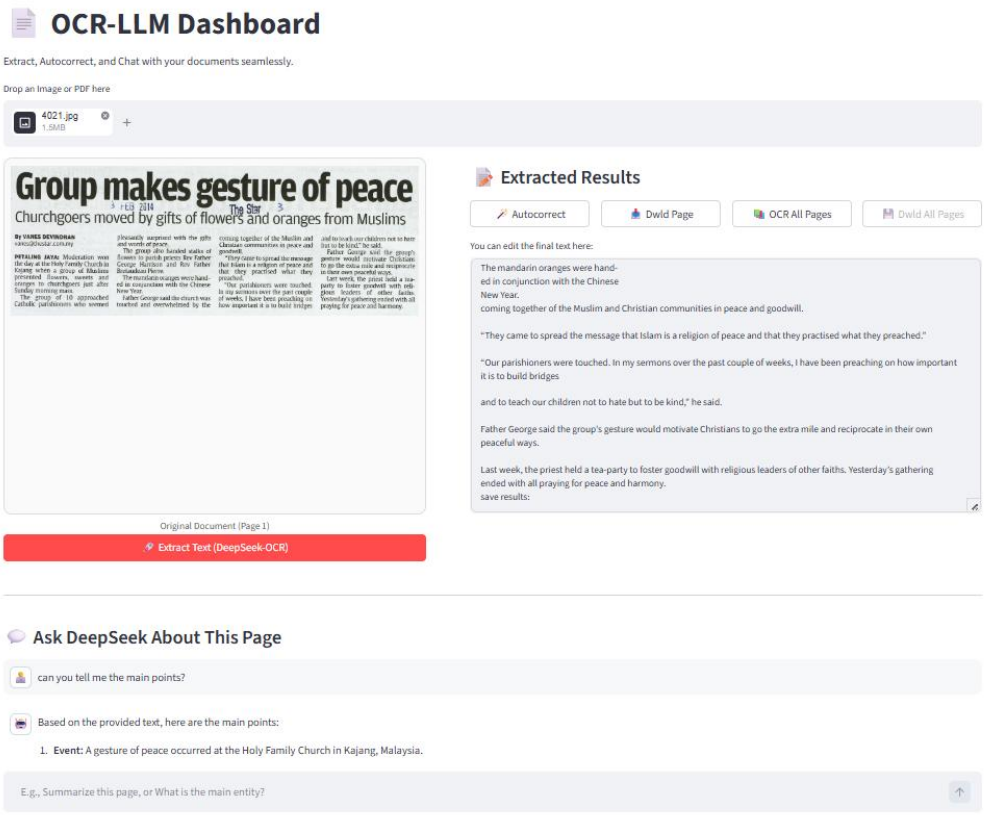

Figure 4.8: Overall Interface of the OCR-LLM Dashboard

4.5.2 File Upload and Document Preview Functions

The file upload and document preview functions allow users to provide input documents to the system in an easy and direct way. As shown in Figure 4.9, users can upload either an image or a PDF file through the upload area located near the top of the dashboard. After the file is uploaded successfully, the document is displayed in the preview panel on the left side of the screen.

This function is important because it gives users a clear view of the original document before OCR is performed. By showing the uploaded page visually, the system allows users to compare the extracted text with the original source. This is useful for checking text quality, layout structure, and possible OCR errors. In addition, for PDF input, the multipage preview also helps users understand which specific page is currently being processed.

Figure 4.9: File upload and Document Preview Functions

4.5.3 OCR Text Extraction Function

The OCR text extraction function is one of the core functions of the system. After a document is uploaded, the user can click the "Extract Text (DeepSeek-OCR)" button as shown in Figure 4.9, to begin the extraction process. Once the OCR process is completed, the extracted text is displayed in the result panel on the right side of the dashboard.

This function allows the system to convert the visual content of the uploaded document into machine readable text. The extracted result is shown in a text box as shown in Figure 4.10 so that users can immediately review the content without leaving the current page.

4.5.4 Text Editing and Autocorrect Functions

After the OCR result is displayed, the extracted text can be edited directly in the text area as shown in Figure 4.10 which allows users to manually correct the extracted content if needed. In addition to manual editing, the system also provides an Autocorrect button to further refine the extracted text.

In this system, the Autocorrect function represents the use of AI agent as a post-processing step. By providing both manual editing and AI refinement,

the system gives users more flexibility in handling OCR results. Users can either make their own corrections or use the autocorrect function to improve the extracted output automatically. Instead of limiting the system to raw text extraction only, the application allows the extracted text to become cleaner and more contextual before it is used for chat interaction or download.

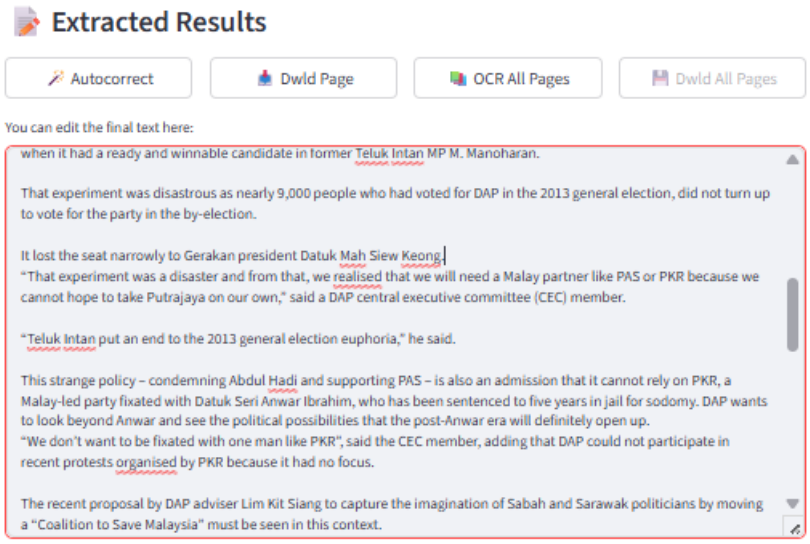

Figure 4.10: OCR Text Extraction Result in the Dashboard

4.5.5 Document Export and Multipage Processing

The system provides several buttons to support document output and multipage processing. These include Download Page, OCR All Pages, and Download All Pages as shown in Figure 4.11. The Download Page function allows the user to download the processed output for the current page only, while OCR All Pages allows OCR extraction to be performed across the entire document. The Download All Pages function supports the downloading of the full extracted result after multipage processing has been completed.

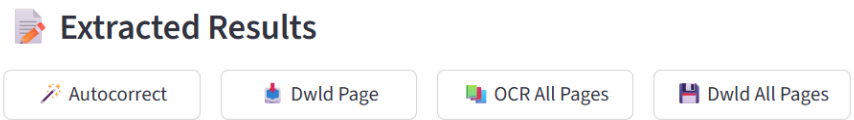

Figure 4.11: Action Buttons for Extracted Results

4.5.6 Chat Interaction Function

The web application also includes a chat function, as shown in Figure 4.12, which is at the lower section of the dashboard. This feature is labelled “Ask DeepSeek About This Page” and allows users to type questions related to the extracted content of the current page. The system then generates a response based on the OCR result shown in the extracted text area. This function allows users to interact with the document content in a more contextual way. For example, users may ask for the main points of the page, request a summary, or ask about specific information mentioned in the document. This supports the project objective of making OCR output more useful through LLM integration.

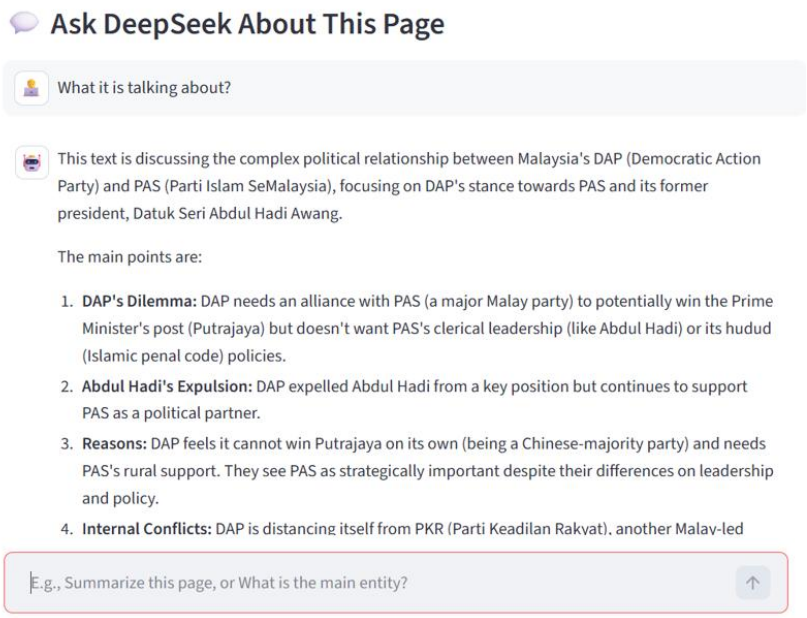

Figure 4.12: Chat Interaction Function

4.5.7 Overall Functional Flow

The overall functional flow of the web application is illustrated in Figure 4.13. The process begins when the user uploads an image or PDF file into the system. After the file is uploaded successfully, the document is displayed in the preview section so that the user can inspect the original content. The user then initiates the OCR process by clicking the extraction button, and the extracted text is displayed in the result panel.

As shown in Figure 4.13, once the OCR output is shown, the user may either edit the text manually or apply the autocorrect function to refine the

extracted result. If the uploaded file contains multiple pages, the user can choose to process all pages using the multipage OCR function. The resulting output can then be downloaded either for the current page only or for the entire document.

In addition, the user may interact with the extracted content through the chat function provided in the dashboard. This allows the system to support not only OCR extraction, but also text refinement and contextual interaction in a single interface. Therefore, the overall workflow shows how OCR and LLM are integrated to produce a more practical and user-friendly application.

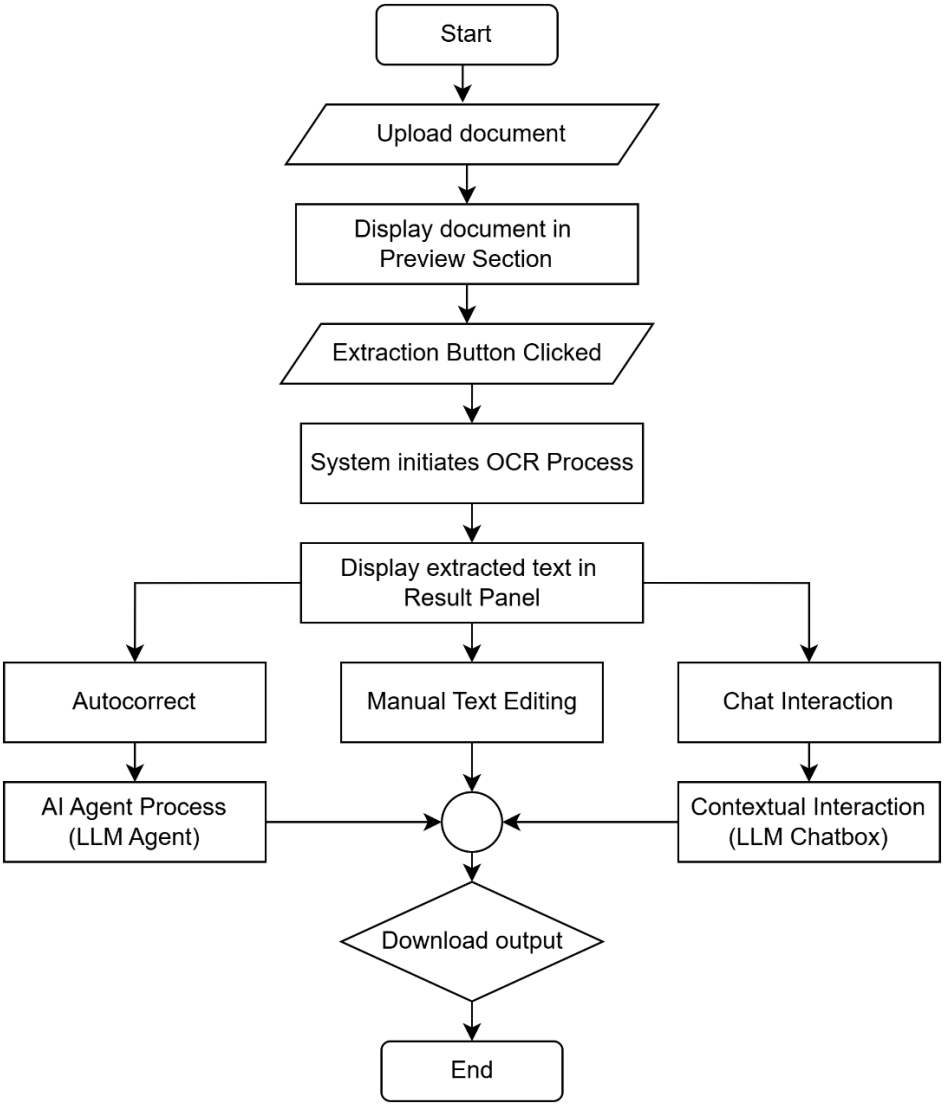

Figure 4.13: Functional Flow of the OCR-LLM Web Application

4.6 Summary

This chapter presented the results and discussion of the proposed OCR-LLM system. The findings showed that DeepSeek-OCR 2 achieved the best overall OCR performance, while DeepSeek-R1 outperformed Llama 3.1 in the LLM evaluation. The AI agent improved the output for English and Malay, although it also introduced a trade-off between improved output quality and exact text matching. In addition, the chapter presented the main interface and functionalities of the developed web application. Overall, the results support the suitability of the selected OCR engine, LLM, and AI refinement process for the proposed system.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project successfully developed a modular OCR-LLM application for multilingual document processing. The system integrates OCR extraction, AI agent post-processing, and local LLM interaction into a Streamlit web interface.

Based on the final evaluation, DeepSeek-OCR 2 was selected as the most suitable OCR engine, achieving an average F1-score of 88.21%, CER of 0.1175 and WER of 0.1622. In the LLM evaluation, DeepSeek-R1 outperformed Llama 3.1 with a BLEURT score of 81.38%, BERTScore of 94.77%, and cosine similarity of 90.46%, showing its ability to support contextual understanding and chat-based interaction. Furthermore, the OCR + AI agent pipeline achieved an average F1-score of 91.34%, showing improvement over the OCR-only baseline. While a trade-off was observed in exact word matching, the AI agent effectively produced cleaner, more useful text.

Overall, the implementation of this project demonstrates that open-source, locally deployable, privacy-aware, and multilingual technologies can be combined into a flexible OCR-LLM pipeline that can be upgraded and optimized in future work. Since the system can be deployed locally, it also reduces reliance on paid cloud services or external API platforms, making it more cost-effective and suitable for users or organisations with limited budgets.

5.2 Recommendations for Future Work

In future research, the proposed OCR-LLM pipeline can be further optimized by upgrading the system with GPU-based hardware. Since OCR extraction, AI agent post-processing, and local LLM inference require high computational resources, GPU acceleration presents a promising approach for improving the processing speed and efficiency of the system. This enhancement would allow the system to handle larger images, multipage PDF files, and higher-volume document processing more effectively.

Furthermore, by expanding the dataset size and increasing the number of supported languages, additional document samples can be collected from different layouts, image qualities, document types, and real-world conditions to provide a more comprehensive evaluation of the OCR-LLM pipeline.

In addition, the AI agent can be further developed into a multifunction agentic AI system. Instead of only refining extracted OCR text, the agent could be extended to interact with external platforms such as email, WhatsApp, or other communication tools. For example, the agent could summarise processed documents, draft responses, send extracted information to selected users, or assist in completing document-related workflows. With these improvements, the optimized system holds strong potential to support practical document automation tasks while maintaining the advantages of a free, open-source, locally deployable, and privacy-aware engineering pipeline.

REFERENCES

Akhil, S. (2016) *An overview of Tesseract OCR engine*. Seminar report. Department of Computer Science and Engineering, National Institute of Technology, Calicut Monsoon. Available at: https://www.researchgate.net/publication/315834331_Overview_of_Tesseract_OCR_engine (Accessed: 25 August 2025).

Aloyan, A. (2025) *Adaptation of CRNNs for low-resource Armenian handwriting recognition via fine-tuning and static detection*. Available at: https://cse.aua.am/files/2025/06/DS299_Capstone_AloyanA.pdf (Accessed: 8 August 2025).

Ataei, M. (2025) *Evaluating the accuracy of large language models for text summarization in Finnish*. Available at: https://lutpub.lut.fi/handle/10024/169636 (Accessed: 27 April 2026).

AWS (2025) *Introducing Strands Agents, an open source AI agents SDK*. Available at: https://aws.amazon.com/blogs/opensource/introducing-strands-agents-an-open-source-ai-agents-sdk/ (Accessed: 30 April 2026).

Bhadra, S. (2022) *The conclusive face-off: Flask vs Streamlit*. Available at: https://somabhadra.medium.com/the-conclusive-face-off-flask-vs-streamlit-40bdef6859a4 (Accessed: 10 September 2025).

Biró, A., Cuesta-Vargas, A.I., Martín-Martín, J., Szilágyi, L. and Szilágyi, S.M. (2023) 'Synthetized multilanguage OCR using CRNN and SVTR models for realtime collaborative tools', *Applied Sciences*, 13(7), pp. 4419-4419. Available at: https://doi.org/10.3390/app13074419 (Accessed: 25 August 2025).

Browne, R. (2025) 'DeepSeek's breakthrough emboldens open-source AI models like Meta's Llama', *CNBC*. Available at: https://www.cnbc.com/2025/02/04/DeepSeek-breakthrough-emboldens-open-source-ai-models-like-meta-Llama.html (Accessed: 29 August 2025).

Cai, Y., Sun, H., Huang, H.-Y. and Wu, Y. (2024) *Assessing the performance of Chinese open source large language models in information extraction tasks*. Available at: https://arxiv.org/abs/2406.02079 (Accessed: 26 April 2026).

Chen, Y., Li, Z., Zhang, C., Yang, C., Cady, A., Lee, A.K., Zeng, Z., Jo, E.L., Pan, H. and Park, J. (2025) *Parsing through boundaries in Chinese word segmentation*. Available at: https://arxiv.org/abs/2503.23091 (Accessed: 26 April 2026).

Christian, I. and Kusuma, G.P. (2023) 'Improving OCR performance on low-quality image using pre-processing and post-processing methods', *International Journal of Engineering Trends and Technology*, 71(6), pp. 396-405. Available at: https://doi.org/10.14445/22315381/ijett-v71i6p239 (Accessed: 29 August 2025).

Cleti, M. and Jano, P. (2024) *Hallucinations in LLMs: types, causes, and approaches for enhanced reliability*. Available at: https://www.researchgate.net/publication/385085962_Hallucinations_in_LLMs_Types_Causes_and_Approaches_for_Enhanced_Reliability (Accessed: 25 August 2025).

Cui, C., Sun, T., Lin, M., Gao, T., Zhang, Y., Liu, J., Wang, X., Zhang, Z., Zhou, C., Liu, H., Zhang, Y., Lv, W., Huang, K., Zhang, Y., Zhang, J., Zhang, J., Liu, Y., Yu, D. and Ma, Y. (2025) *PaddleOCR 3.0 technical report*. PaddlePaddle Team, Baidu Inc. Available at: https://arxiv.org/pdf/2507.05595 (Accessed: 31 July 2025).

Datta, J.K., Rehan, M., Raj, I., P, A.S. and Gowda V, K. (2025) 'Survey on OCR and CNN-based approaches for text extraction from images and documents', *International Advanced Research Journal in Science, Engineering and Technology*, 12(2), pp. 148-156. Available at: https://doi.org/10.17148/IARJSET.2025.12218 (Accessed: 29 August 2025).

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A. et al. (2024) *The Llama 3 herd of models*. Available at: https://arxiv.org/abs/2407.21783 (Accessed: 25 August 2025).

Felix-T2K (2023) *mindee/doctr*. GitHub. Available at: https://github.com/mindee/doctr (Accessed: 8 August 2025).

Ferraris, A.F. et al. (2025) 'The architecture of language: understanding the mechanics behind LLMs', *Cambridge Forum on AI: Law and Governance*, 1, p. e11. Available at: https://doi.org/10.1017/cfl.2024.16 (Accessed: 9 September 2025).

Folder IT (2025) *Streamlit: build interactive web apps with just Python*. Available at: https://folderit.net/streamlit-build-interactive-web-apps-with-just-python/ (Accessed: 10 September 2025).

Gadade, Y.A. (2019) *Tesseract introduction, installation*. Available at: https://limitlessdatascience.wordpress.com/2019/07/01/tesseract-4-0-intro-installation/ (Accessed: 29 August 2025).

Gaikwad, V. (2023) 'OpenCV: a comprehensive review and applications in computer vision', *International Journal of Advanced Research in Science, Communication and Technology*, 3(2), pp. 291-297. Available at: https://doi.org/10.48175/568 (Accessed: 10 September 2025).

GeeksforGeeks (2024) *Llama 3: Meta's new AI model*. Available at: https://www.geeksforgeeks.org/artificial-intelligence/Llama-3-metas-new-ai-model/ (Accessed: 25 August 2025).

Google Cloud (2026) *Cloud Vision pricing*. Available at: https://cloud.google.com/vision/pricing (Accessed: 1 May 2026).

Gupta, U., Kumar, A., Gupta, A., Raj, G. and Agrawal, A.P. (2024) 'Advances in handwritten character recognition: a comparison of OCR and large language model-based approaches', *2024 International Conference on Emerging Technologies and Innovation for Sustainability (EmergIN)*, pp. 192-198. Available at: https://doi.org/10.1109/emergin63207.2024.10961487 (Accessed: 8 August 2025).

Hosseini, F.S., Shabaninia, E. and Nezamabadi-pour, H. (2025) 'Empirical evaluation of well-known Farsi OCR engines on the IDPL-PFOD dataset', *Power, Control, and Data Processing Systems*, 2(1), p. e722202. Available at: https://doi.org/10.30511/pcdp.2025.2052926.1018 (Accessed: 23 August 2025).

Ingole, S.A. and Jain, S.A. (2025) 'Text information extraction from images using deep CNN', *International Journal of Environmental Sciences*, 11, pp. 1442-1455. Available at: https://doi.org/10.64252/4mn15v76 (Accessed: 9 August 2025).
Jaided Ai (n.d.) *Jaided AI: EasyOCR documentation*. Available at: https://www.jaided.ai/easyocr/documentation/ (Accessed: 8 August 2025).

Junadhi, Agustin, Efrizoni, Okmayura, F., Habibie, D.R. and Muslim (2025) 'Improving evaluation metrics for text summarization: a comparative study and proposal of a novel metric', *Journal of Applied Data Sciences*, 6(2), pp. 885-896. Available at: https://doi.org/10.47738/jads.v6i2.547 (Accessed: 9 September 2025).

Kashyap, P. (2024) *Understanding precision, recall, and F1 score metrics*. Available at: https://medium.com/@piyushkashyap045/understanding-precision-recall-and-f1-score-metrics-ea219b908093 (Accessed: 9 September 2025).

Kop, A. (2023) *Building web apps very fast with Solara*. Available at: https://medium.com/@arlindkopliku/building-web-apps-very-fast-with-solara-6e0773de904f (Accessed: 9 September 2025).

Leung, K. (2021) *Evaluate OCR output quality with character error rate (CER) and word error rate (WER)*. Available at: https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510/ (Accessed: 9 September 2025).

Li, G., Zhang, Y., Wang, Y., Yan, S., Wang, L. and Wei, T. (2025) PRIV-QA: Privacy-Preserving Question Answering for Cloud Large Language Models. arXiv. Available at: https://doi.org/10.48550/arXiv.2502.13564 (Accessed: 1 May 2026).

Li, H., Sun, Y., Schlegel, V., Yang, K., Batista-Navarro, R. and Nenadic, G. (2025) *Arg-LLaDA: argument summarization via large language diffusion models and sufficiency-aware refinement*. Available at: https://arxiv.org/abs/2507.19081 (Accessed: 27 April 2026).

Li, X. (2024) ‘Comparative analysis and prospect of RNN and Transformer’, *Applied and Computational Engineering*, 75(1), pp. 178-184. Available at: https://doi.org/10.54254/2755-2721/75/20240535 (Accessed:2 8 August 2025).

Liao, H., RoyChowdhury, A., Li, W., Bansal, A., Zhang, Y., Tu, Z., Satzoda, R.K., Manmatha, R. and Mahadevan, V. (2023) ‘Doctr: document transformer for structured information extraction in documents’, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9584-19594. Available at: https://arxiv.org/pdf/2307.07929 (Accessed: 8 August 2025).

Liao, M., Wan, Z., Yao, C., Chen, K. and Bai, X. (2020) ‘Real-time scene text detection with differentiable binarization’, *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07), pp. 11474-11481. Available at: https://doi.org/10.1609/aaai.v34i07.6812 (Accessed: 8 August 2025).

Lim, Z.Y., Bong, Z.J., Pang, Y.H., Ooi, S.Y., Khoh, W.H. and Hiew, F.S. (2025) ‘Integration of YOLOv8 and PaddleOCR for car plate recognition system in diverse environments: Malaysia’, *Journal of Engineering Science and Technology*, 20(1), pp. 250-270. Available at: https://jestec.taylors.edu.my/Vol%2020%20Issue%201%20February%202025/20_1_16.pdf (Accessed: 8 August 2025).

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F. and Hao, G. (2024) *DeepSeek-V3 technical report*. Available at: https://doi.org/10.48550/arXiv.2412.19437 (Accessed: 25 August 2025).

Loukil, F., Cadereau, S., Verjus, H., Galfre, M., Salamatian, K., Telisson, D., Kembellec, Q. and Le Van, O. (2024) ‘LLM-centric pipeline for information extraction from invoices’, *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pp. 569-575. Available at: https://doi.org/10.1109/fllm63129.2024.10852504 (Accessed: 14 August 2025).

Mangal, A. (2025) *A researcher’s deep dive: comparing top OCR frameworks*. Available at: https://adityamangal98.medium.com/a-researchers-deep-dive-comparing-top-ocr-frameworks-ca6327b3cc86 (Accessed: 21 August 2025).

Matarazzo, A. and Torlone, R. (2025) ‘A survey on large language models with some insights on their capabilities and limitations’, *arXiv*. Available at: https://doi.org/10.48550/arxiv.2501.04040 (Accessed: 14 August 2025).

MathWorks (2023) *Text detection and recognition*. Available at: https://www.mathworks.com/help/vision/text-detection-and-recognition.html (Accessed: 29 August 2025).

Mindee (2023) *doctr.models – DocTR documentation*. Available at: https://mindee.github.io/doctr/latest/modules/models.html (Accessed: 8 August 2025).

Obi, J.C. (2023) ‘A comparative study of several classification metrics and their performances on data’, *World Journal of Advanced Engineering Technology and Sciences*, 8(1), pp. 308-314. Available at: https://doi.org/10.30574/wjaets.2023.8.1.0054 (Accessed: 28 August 2025).

Olson, L. and Berry, V. (2021) ‘Digitization decisions: comparing OCR software for librarian and archivist use’, *The Code4Lib Journal*, 52. Available at: https://journal.code4lib.org/articles/16132 (Accessed: 31 July 2025).

Park, J., Cho, J., Han, S. and Kim, K. (2025) ‘OCR-augmented GPT for accurate text extraction in industrial environments’, *IEEE Access*, 13, pp. 136226-136236. Available at: https://doi.org/10.1109/access.2025.3594682 (Accessed: 14 August 2025).

Patience, O.O., Amaechi, E.M., George, O. and Isaac, O.N. (2024) ‘Enhanced text recognition in images using Tesseract OCR within the Laravel framework’, *Asian Journal of Research in Computer Science*, 17(9), pp. 58-69. Available at: https://doi.org/10.9734/ajrcos/2024/v17i9499 (Accessed: 8 August 2025).

Patil, S., Varadarajan, V., Mahadevkar, S., Athawade, R., Maheshwari, L., Kumbhare, S., Garg, Y., Dharrao, D., Kamat, P. and Kotecha, K. (2022) ‘Enhancing optical character recognition on images with mixed text using semantic segmentation’, *Journal of Sensor and Actuator Networks*, 11(4), p. 63. Available at: https://doi.org/10.3390/jsan11040063 (Accessed: 9 August 2025).

Reimers, N. and Gurevych, I. (2019) *Sentence-BERT: sentence embeddings using Siamese BERT-networks*. Available at: https://arxiv.org/abs/1908.10084.
Sai, K.M., P, H.C., Bebe, K., Roja Pramila, G.S. and Sankara Rao, G. (2020) ‘Optical character recognition using CRNN’, *International Journal of Innovative Technology and Exploring Engineering*, 9(8), pp. 115-120. Available at: https://doi.org/10.35940/ijitee.h6264.069820 (Accessed: 10 September 2025).

Sartori, G. and Orrú, G. (2023) ‘Language models and psychological sciences’, *Frontiers in Psychology*, 14. Available at: https://doi.org/10.3389/fpsyg.2023.1279317 (Accessed: 10 September 2025).

Sayed, E. (2024) *How to build interactive web apps with Streamlit*. Available at: https://medium.com/@sayedebad.777/how-to-build-interactive-web-apps-with-streamlit-1c80cd748006 (Accessed: 9 September 2025).

scikit-learn (2026) *Precision-recall*. Available at: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html?utm_source= (Accessed: 26 April 2026).

Sellam, T., Das, D. and Parikh, A. (2020) *BLEURT: learning robust metrics for text generation*. Available at: https://arxiv.org/pdf/2004.04696 (Accessed: 11 September 2025).

Sharma, P. (2023) ‘Advancements in OCR: a deep learning algorithm for enhanced text recognition’, *International Journal of Inventive Engineering and Sciences*, 10(8), pp. 1-7. Available at: https://doi.org/10.35940/ijies.f4263.0810823 (Accessed: 18 August 2025).

Singh, P. (2025) ‘Deep learning-based text detection and recognition in document images: a comprehensive review’, *International Journal of Creative Research Thoughts*, 13(4), pp. 369-377. Available at: https://ijcrt.org/papers/IJCRT25A4800.pdf (Accessed: 31 July 2025).

Singh, R. (2024a) *Large language models and text summarization: a powerful combination*. Available at: https://medium.com/@singhrajni/large-language-models-and-text-summarization-a-powerful-combination-6400e7643b70 (Accessed: 25 August 2025).

Singh, Y. (2024b) *What is GQA (grouped query attention) in Llama 3*. Available at: https://medium.com/%40yashsingh.sep30/what-is-gqa-grouped-query-attention-in-Llama-3-c4569ec19b63 (Accessed: 29 August 2025).
Skywork (2026) *DeepSeek-OCR 2: a deep dive into the future of document AI*. Available at: https://skywork.ai/skypage/en/deepseek-ocr-document-ai/2016342792772972544 (Accessed: 27 April 2026).

Smith, R. (2007) *An overview of the Tesseract OCR engine*. Available at: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf (Accessed: 8 August 2025).

Tammana, S. (2023) *Text detection using EasyOCR Python*. Available at: https://medium.com/%40sreekartammana/text-detection-using-easyocr-python-5d900623f306 (Accessed: 29 August 2025).

Tanasă, A.-M. and Oprea, S.-V. (2025) ‘Rethinking chart understanding using multimodal large language models’, *Computers, Materials & Continua*, 84(2), pp. 2905-2933. Available at: https://doi.org/10.32604/cmc.2025.065421 (Accessed: 10 September 2025).

TechPrate (2023) *OCR solutions: transform data extraction and document processing*. Available at: https://www.techprate.com/tech/ocr-solutions-transform-data-extraction-document-processing/ (Accessed: 29 August 2025).

Thresholding (2024) *Thresholding*. Available at: https://datacarpentry.github.io/image-processing/07-thresholding.html (Accessed: 29 August 2025).

Wajer, M. (2023) *OCR at the Internet Archive with Tesseract and hOCR*. Available at: https://archive.org/developers/ocr.html (Accessed: 8 August 2025).

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N.A., Khashabi, D. and Hajishirzi, H. (2022) *Self-Instruct: aligning language models with self-*

generated instructions. Available at: https://arxiv.org/abs/2212.10560 (Accessed: 10 September 2025).

Wei, H., Sun, Y. and Li, Y. (2026) *DeepSeek-OCR 2: visual causal flow*. Available at: https://arxiv.org/abs/2601.20552 (Accessed: 27 April 2026).

Xiong, J. and Liu, C. (2024) 'Design and research of industrial high availability character recognition technology', *2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Chongqing, China, 2024, pp. 1472-1476. Available at: https://doi.org/10.1109/IMCEC59810.2024.10575809 (Accessed: 31 July 2025).

Zhang, T., Kishore, V., Wu, F., Weinberger, K. and Artzi, Y. (2019) *BERTScore: evaluating text generation with BERT*. Available at: https://arxiv.org/abs/1904.09675 (Accessed: 11 September 2025).

APPENDICES

Appendix A: Tables

Table A-1: OCR Performance Comparison (Single-Column Layout)

OCR Engine	Lang.	Accuracy (%)	CER	WER	Precision	Recall	F1-score
DeepSeek-OCR	EN	97.21	0.0279	0.0325	1.0000	1.0000	1.0000
	MAL	96.97	0.0303	0.1000	1.0000	1.0000	1.0000
	CHI	76.33	0.2367	0.6078	0.8825	0.8455	0.8636
PaddleOCR	EN	94.70	0.0530	0.1870	1.0000	0.9804	0.9901
	MAL	93.93	0.0607	0.3091	1.0000	0.9800	0.9899
	CHI	76.14	0.2386	1.3137	0.9525	0.9450	0.9488
Tesseract	EN	94.70	0.0530	0.1220	0.9434	0.9804	0.9615
	MAL	86.21	0.1379	0.3455	0.9038	0.9400	0.9216
	CHI	61.52	0.3848	2.4314	0.7713	0.7592	0.7652
EasyOCR	EN	89.68	0.1032	0.3252	0.9091	0.9804	0.9434
	MAL	72.41	0.2759	0.7364	0.8103	0.9400	0.8704
	CHI	57.59	0.4241	2.8235	0.7739	0.8063	0.7897
DocTR	EN	96.51	0.0349	0.0650	0.9804	0.9804	0.9804
	MAL	95.72	0.0428	0.2091	0.9800	0.9800	0.9800
	CHI	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*

**Note: DocTR data was not available for Chinese*

Table A-2: OCR Performance Comparison (Unstructure Layout)

OCR Engine	Lang.	Accuracy (%)	CER	WER	Precision	Recall	F1-score
DeepSeek-OCR	EN	88.70	0.0483	0.1130	0.9472	0.9065	0.9258
	MAL	84.44	0.0578	0.1556	0.9109	0.8710	0.8901
	CHI	78.20	0.2465	0.2180	0.8403	0.8223	0.8304
PaddleOCR	EN	31.05	0.5309	0.6895	0.9271	0.8078	0.8575
	MAL	37.25	0.4501	0.6275	0.8999	0.7755	0.8290
	CHI	31.27	0.6652	0.6873	0.8167	0.7853	0.8001
Tesseract	EN	72.18	0.2231	0.2782	0.8629	0.8257	0.8279
	MAL	68.15	0.2109	0.3185	0.8361	0.8449	0.8397
	CHI	40.20	0.6292	0.5980	0.6499	0.5234	0.5683
EasyOCR	EN	39.42	0.4619	0.6058	0.9046	0.9060	0.9050
	MAL	45.98	0.3825	0.5402	0.8709	0.8738	0.8719
	CHI	60.69	0.4853	0.3931	0.8261	0.8177	0.8214
DocTR	EN	30.42	0.5405	0.6958	0.8789	0.7955	0.8269
	MAL	39.44	0.4416	0.6056	0.8812	0.8640	0.8721
	CHI	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*

**Note: DocTR data was not available for Chinese*