COMPARATIVE ANALYSIS OF OCRflow WITH STANDALONE OPTIC CHARACTER RECOGNITION SYSTEMS

Onu Sunday[1], Ugah J.O.[2], Ituma Chinagorom[2] & Ogbu Henry N.[2]
[1]David Umahi Federal University of Health Sciences, Uburu
[2] Department of Computer Science, Ebonyi State University, Abakaliki

ORCID ID: 0009-0005-1216-3135

Abstract
This research work compares standalone OCR with Hybrid model. Optical Character Recognition (OCR) system has been widely adopted to convert handwritten documents to a digitized, machine-readable format. This OCR system often produces error-prone outputs when handling cursive or irregular handwriting. At the same time, the emergence of Large language models, such as OpenAI's GPT family, has enabled automated correction of textual errors while preserving meaning. This article presents OCRFlow, a novel hybrid system that combines Google Cloud Vision OCR and OpenAI's GPT with a user-friendly graphical interface built using Python's Tkinter. It also analyses its accuracy, latency, efficiency and usability, with the aim to design a robust system that performs more efficiently than any standalone model.

Keywords: OCR, Convolutional Neural Network, GPT, Deep Learning, Connectionist Temporal Classification, Machine Learning, Long Short Term Memory, Language Model, Transformer Model

Introduction
Handwriting is one of the oldest and most enduring forms of human communication. Even in the digital age, handwritten documents continue to play a vital role in academia, healthcare, legal practices, and government administration. Examination scripts, medical prescriptions, meeting notes, and archival manuscripts are still predominantly handwritten across much of the world.
Recognizing handwriting on paper has been a big problem for people over time. Because it is such a complex skill and there are many children including adults who have difficulty mastering it. This may cause frustration and distress and affects a child's desire to write. There are some common problems with handwriting. Perhaps the most obvious problem is poor quality or illegible handwriting. We all know the old stereotype about doctors' handwriting, so trying to understand what others are writing can be challenging.
Traditional OCR systems were initially designed for machine-printed text and struggled with handwritten inputs. Over time, the integration of computer vision and machine learning improved recognition capabilities. Nevertheless, even state-of-the-art OCR engines still produce errors, especially when confronted with cursive writing, low-quality images, or non-standard writing styles. Errors in OCR outputs limit their usability in formal contexts, necessitating manual correction, which reintroduces inefficiency.
Recent developments in Artificial Intelligence (AI), particularly Natural Language Processing (NLP) and Large Language Models (LLMs), present an opportunity to bridge this gap. LLMs like GPT-4 can correct spelling and grammar errors, ensure contextual accuracy, and preserve semantic meaning. Their integration with OCR systems offers a hybrid solution that automates not only recognition but also correction.

Aim and objectives

The aim of the research work is to make a comparative analysis of OCRFlow, a hybrid system that integrates OCR and generative AI, and other standalone systems.

The objective of the project is:

1. To compare the OCRFlow with other Standalone OCR systems
2. To evaluate OCRFlow in terms of accuracy, efficiency, and usability.

Significance of the study

Handwritten Document to Digital Document Converter, OCRFlow, is significant because it enables the digitization of handwritten data, which has numerous applications across various fields. HCR facilitates the conversion of handwritten text into digital format, making it easier to store, search, and analyze vast amounts of handwritten information. This technology improves efficiency, accuracy, and cost-effectiveness in tasks like document processing, data entry, and even historical document preservation. It is significant in the following ways:

i. Document Digitization: OCR plays a vital role in digitizing historical documents, personal records, and other handwritten materials, making them accessible for research and analysis.
ii. Automated Data Entry: It automates the process of entering handwritten data into computer systems, reducing manual effort and increasing efficiency.
iii. Postal Automation: OCR is used in automated postal sorting systems to recognize addresses on envelopes, streamlining mail processing.
iv. Forensic Analysis: OCR can be used in forensic analysis to verify signatures, analyze handwriting patterns, and potentially identify authors.
v. Enhanced Accessibility: By converting handwritten text to digital format, OCR makes information more accessible to individuals with disabilities and those who may not be able to read handwritten text.
vi. Cost Savings and Increased Efficiency: Automating data entry and other tasks reduces the time and labor required, leading to cost savings and increased operational efficiency.
vii. Improved Accuracy: Advanced machine learning algorithms can minimize errors associated with manual data entry, enhancing the accuracy and reliability of the digitized information.
viii. OCR is used to process bank checks, recognize signatures, and verify financial transactions.
ix. Academia: Automated correction of student submissions, lecture notes, and archival manuscripts. It can also be used to grade handwritten exams, analyze student writing, and create digital learning materials
x. Healthcare: Digitization of handwritten prescriptions and patient notes. It is also is used in electronic health records (EHR) to transcribe handwritten notes and medical records, reducing errors and improving patient care
xi. Government and Business: Improved management of handwritten records, reducing reliance on manual clerical work.
xii. Technology Research: Demonstrates the value of hybrid approaches combining vision and language models.

xiii. Legal Field: OCR can be used to analyze handwritten documents, transcripts, and court records, facilitating legal research and analysis.

OCRFlow contributes not just as a proof-of-concept application but also as a framework for future work exploring deeper AI integration in document processing

Hypothesis

H_0 : There is no significant difference in the mean SUS score between OCRFlow and Standalone OCR system

H_1 : There is a significant difference in mean SUS scores between OCRFlow and Standalone OCR system

LITERATURE REVIEW

Analysis of Standalone OCR

OCR software streamlines the process of converting handwritten text into digital format, enhancing efficiency and accuracy in data entry tasks. When comparing OCR software options, it is essential to consider factors like accuracy rates, language support, and compatibility with different file formats. Some OCR tools excel in recognizing specific handwriting styles, while others offer more advanced features such as automatic language detection and intelligent text formatting. [1]_By utilizing OCR software, you can significantly reduce the time and effort required for manual data entry tasks, ultimately improving productivity and minimizing errors. These digital handwriting solutions not only save you valuable time but also ensure that your data is accurately captured and easily searchable.

According to [2] below are the ten best open source OCR softwares with their advantages and disadvantages:

1. Tesseract: This is a powerful open-source OCR, maintained by Google, compatible with Linux, Windows, and OS X. It supports numerous languages, with the ability to add additional extensions. While it is flexible, its use can be complex for beginners, requiring coding knowledge and setup. Once mastered, it provides accurate results and is highly versatile.

ADVANTAGES

I. Highly configurable
II. Multilingual support for over 100 languages
III. Widely used and well-documented with a large support community

DISADVANTAGES

i. Can be complex to configure, requires technical skills
ii. Less efficient on handwritten or highly distorted documents.
iii. Longer processing time for complex documents or large volumes

2. EasyOCR: This is appreciated for its ease of integration and good performance, especially with medium-quality images. It supports over 80 languages and integrates easily into Python projects. Although it is less efficient than Tesseract for complex cases, its execution speed and ease of use make it an ideal choice for simpler needs.

ADVANTAGES

i. Good performance on blurry or medium-quality images
ii. Easy integration with Python

DISADVANTAGES

i. Lacks advanced customization compared to Tesseract
ii. Less efficient for very complex documents

3. Mistral: This is a powerful and fast OCR, known for its ability to handle a wide range of image formats. It is particularly valued for its high recognition accuracy, whether for simple documents or complex layouts.

ADVANTAGES

i. Optimized for speed
ii. Good recognition on non-standard documents
iii. Supports many languages, ideal for multilingual documents

DISADVANTAGES

i. Less suited for heavily stylized or unstructured documents
ii. Formatting inconsistencies in some extractions
iii. May misclassify certain PDFs as images

4. OCRopus is a modular solution that offers customization and flexibility to meet the specific needs of each project. It is particularly notable for its ability to process historical and handwritten documents, thanks to its adaptable structure. This makes it especially well-suited for advanced users with specialized requirements in text processing.

ADVANTAGES

i. Ability to add new modules
ii. Support for historical documents

DISADVANTAGES

i. Difficult to configure
ii. Less accessible for beginner users.

5. Doctr: This is a modern OCR tool focused on recognizing structured documents, such as forms or scanned files. Built on deep learning models, it performs well with documents featuring diverse layouts. It excels in recognizing well-structured text and offers a good level of flexibility, though it may have limitations when dealing with more complex documents.

ADVANTAGES

i. Uses deep learning models
ii. Excellent accuracy on well-structured documents

DISADVANTAGES

i. Documentation can sometimes be insufficient
ii. Limited for complex documents

6. Kraken: This is a sophisticated OCR engine that excels in recognizing old or historical documents. It is particularly well-suited for text recognition in complex formats, with remarkable accuracy in this area. While it may not be as widely known as other options, Kraken is an excellent choice for projects requiring precise and detailed processing of hard-to-read documents.

ADVANTAGES

i. Excellent recognition for historical documents

ii. Good customization capabilities

DISADVANTAGES

i. Slower than other OCRs for large volumes
ii. Lacks a GUI or simple integration tools, making it less accessible for non-technical users

7. Surya OCR: This stands out for its ability to handle complex documents, particularly those containing tables or mathematical elements. While its accuracy is high, its processing speed can be a drawback when dealing with large volumes of data.

ADVANTAGES

i. Suitable for complex documents, very good symbol detection accuracy (currencies, negative numbers, etc.)
ii. Can process low-quality images

DISADVANTAGES

i. Relatively slow processing due to the complexity of the algorithms used
ii. Limited documentation and difficulty finding online technical support

8. CuneiForm: This is an open-source OCR that, while not as powerful as other major solutions, remains useful for basic OCR tasks. This OCR engine is particularly suited for users seeking a straightforward solution without the need for advanced features. It supports multiple image formats and is easy to deploy.

ADVANTAGES

i. Easy to use with no technical skills required, and very fast for simple tasks.
ii. Supports numerous file formats

DISADVANTAGES

i. Less reliable accuracy on complex documents
ii. Outdated user interface, no active updates

9. OCRmyPDF: This is an ideal tool for automating OCR on PDF files, especially when you have a large volume of scanned documents to process. While it is limited to PDF use, it is highly effective for mass scanning tasks.

ADVANTAGES

i. Convenient for automating OCR on PDFs
ii. Easy integration with scripts and document processing tools

DISADVANTAGES

i. Limited to PDF files only
ii. Slow processing for large documents

10. OCR Space: This is a fast and efficient online OCR tool, perfect for users who don't want to install software. However, due to its online nature, it may raise data privacy concerns, and its performance is generally lower compared to local solutions.

ADVANTAGES

i. Easy access without installation
ii. Effective recognition even on medium-quality images

DISADVANTAGES

i. Limited to online services, privacy concerns
ii. Less efficient for complex documents

Steps in OCR

The OCR engine or OCR software works by using the following steps [3]:

1. Image acquisition

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

2. Preprocessing

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

i. De-skewing or tilting the scanned document slightly to fix alignment issues during the scan.
ii. De-speckling or removing any digital image spots, or smoothing the edges of text images.
iii. Cleaning up boxes and lines in the image.
iv. Script recognition for multi-language OCR technology

3. Text recognition

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

4. Pattern matching

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

5. Feature extraction

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

6. Postprocessing

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

OCR Scanning Tools

When selecting OCR software for your needs, be sure to evaluate your specific requirements and choose a tool that best fits your workflow and objectives. With the right OCR software in place, you can streamline your data entry processes and enhance overall efficiency in handling handwritten text.

The software tools, according to [4], include:

1. Adobe scan

Adobe Scan offers several modes for scanning, including books, documents, business cards, ID cards, and even whiteboards. is a solid option for anyone looking for a free OCR app. It comes with basic file management and editing capabilities, which makes it edge out Microsoft Lens for a free option. And for those looking for a few more features, like the ability to export to Word, compress and combine PDFs, protect PDFs with a password, or digitize up to 100 pages, there's a premium plan available. Its disadvantage is that it has some accuracy hiccups. [4]

2. Apple Notes

Apple Notes is intended to be a note-taking app, but Apple has added OCR features that are good enough to get it a place on this list. To scan and digitize text, click on the camera icon, four options, including *Scan Documents* and *Scan Text* will appear. The first option just scans the document and places it in a note, while the Scan Text option allows you to select a section of text and insert the text directly into a note. Its disadvantage is that scanning and digitizing a large section of text is difficult [1]

3. CamScanner

As an OCR software, CamScanner has the following advantages [4]:

i. Text can be edited directly within documents
ii. It has multiple formatting options for digitized documents
iii. Has various export options, including Word, Excel, PowerPoint, or image
iv. Also has additional AI-based features, like solving math and translating text

Its major constraint is that OCR may miss spaces or punctuation

4. Microsoft Lens

Microsoft Office Lens offers a powerful solution for digitizing physical documents and whiteboards with ease. Through advanced image processing and visual recognition technology, Office Lens allows you to capture information from various sources effortlessly. The app excels in document digitization, enabling you to convert hard copies into editable digital files seamlessly. With its text extraction feature, Office Lens accurately recognizes printed text, making it searchable and editable. By simply snapping a picture using your smartphone or tablet, you can transform handwritten notes, receipts, or sketches into digital formats ready for further processing. [1]

It has the following disadvantages:

i. There is no way to create folders for file management
ii. You can't edit digitized text within the app

5. iScanner

As an OCR system iScanner has an Intuitive interface with helpful tutorials. It also has an abundance of extra features, such as the ability to do math or measure distances. Its disadvantages include that its most useful features, including OCR, are only available for paid users and formatting can be a little awkward for digitized text [1]

6. Notebloc

Notebloc is a versatile handwritten data entry tool that simplifies digitizing your notes and documents. The key features that make Notebloc a must-have tool for handwritten data entry are [1]:

i. Smart Cropping: Automatically detects and crops images of your notes or documents, ensuring a clean digital output.
ii. Cloud Integration: Seamlessly sync your digitized notes across devices using popular cloud services like Google Drive or Dropbox.
iii. OCR Technology: Convert handwritten text into editable digital text for easy searching and editing.
iv. Export Options: Share your digitized notes in various formats such as PDF or JPEG to suit your needs.
v. Annotation Tools: Add annotations, highlights, or comments to your digitized notes for better organization and clarity.

7. Pen to Print
Pen to Paper has the ability to scan and digitize handwritten text. The scanned text is accurate even with slightly messy handwriting. But its disadvantage is that it cannot create a folder [1]
8. Google lens
Google Lens becomes a mobile scanning and OCR app when used with other apps, like Google Photos. It has a built-in web search access using a scanned photo or digitized text and a built-in access to Google Translate. Its disadvantage is it has no way to store digitized files or text [1]

Analysis of OCRflow System
OCRFlow integrates Google Cloud Vision OCR, GPT correction. Google Cloud Vision OCR has superior handwriting recognition capabilities and support for multiple languages. It uses Deep Neural Network [like Convolutional Neural Networks (CNNs)] combined with Sequence Models. The model works as stated below:

1. CNNs are used for feature extraction. They detect strokes, shapes, and patterns in scanned documents or handwritten text. They convert raw pixels into high-level features (edges, characters, word shapes, etc.).
2. This is followed by Recurrent Neural Networks (e.g., Long Short-Term Memory (LSTM)), are used to model text sequences. This is important because text has order or sequence (e.g., "tab" and "bat", "yam" and "may", etc.). RNN helps to understand the flow of letters and words.
3. The Connectionist Temporal Classification (CTC) loss function aligns predicted sequences with actual text, handling cases where spacing is irregular (common in handwriting or noisy scans), since OCR does not know where one character ends and another begins.
4. Transformers and Attention Models help the model to focus on relevant parts of an image (like specific text regions) when recognizing characters.

OCRFlow also integrates OpenAI GPT, a Generative Pre-trained Transformer (GPT) with ability to preserve sentence structure while correcting grammar. OpenAI GPT has the following algorithms:

1. Transformer architecture introduced by [5]. It is a sequence-to-sequence model that looks at all words in a sentence (or tokens in text) simultaneously and capture long range dependencies. It processes text in parallel making it much faster and more scalable.
2. Self-Attention Mechanism captures how important each word is in relation to others. For instance, in the sentence "The boy fought two bullies when he was attacked", attention helps the model link "he" to the "boy" not "bullies". This gives GPT strong context understanding
3. In Feed-Forward Neural Network data is passed through fully connected neural networks for deeper feature extraction. The data is stacked into multiple layers or blocks to enable hierarchical language understanding
4. In Self-Supervised Learning, GPT is trained on massive text data using next-token prediction (causal language modelling). In the example above "The boy fought", the model learns to predict "two bullies". Here , no human label is needed but just raw text
5. Reinforcement Learning from Human Feedback (RLHF): Here, after pre-training, GPT is fine-tuned to align its responses with human intent, safety, and usefulness.

The system also integrates a Tkinter GUI with Word/PDF export features. The Tkinter displays live video preview, raw OCR output, and corrected output side by side, while python-docx and ReportLab saves corrected text as Word or PDF documents.

Strengths of OCRFlow

The evaluation highlights several strengths:

1. Hybrid Accuracy
2. Enhances user trust
3. Direct Word and PDF export.
4. Runs efficiently on mid-range hardware.
5. Code structured with .env and organized folders to enhance replicability.

Weaknesses and limitations

Despite these strengths, OCRFlow also presents weaknesses:

1. Internet Dependence
2. Billing Requirements.
3. Language Limitations
4. No Batch Processing
5. Potential GPT Over-Correction like subtle lexical substitutions.
6. No Scalability.
7. Not fine-tuned for specialized domains (e.g., medical or legal handwriting).

Comparing OCRFlow with Google Cloud Vision OCR system

Using Google Cloud Vision OCR alone gives a strong optical character recognition (extracting text from images, handwriting, scanned documents, etc.), but it often stops at raw text output. The output may still have issues like:

i. Misspelling from noisy scan or handwriting
ii. Formatting inconsistency
iii. Missing context like confusing "0" with "O" and "1" with "I"
iv. Lack of natural language understanding

When it is integrated with a GPT model, like OPenAI, perception (seeing text) and language intelligence (understanding and improving the text) are combined. This will lead to the following:

i. Improved Accuracy: OCR may misinterpret characters, especially in handwritten or poor-quality images, and GPT can auto-correct spelling, grammar, and word context (e.g., fixing "hte" to "the" or "decnet" to "decent"
ii. Contextual Understanding: OCR gives text literally as it sees it, OpenAI can infer meaning, detect entities, summarize, or clarify the extracted text (like identifying numbers, addresses, or medical terms).
iii. Noise Reduction: Standalone OCR may output broken sentences or irrelevant symbols. GPT can filter, clean, and reconstruct readable and meaningful sentences.
iv. Language Flexibility: OCR supports multiple languages but does not handle code-switching, idioms, or translation well. GPT can refine the text into a target language, making it more useful in multilingual settings.

v. Human-like Interaction: With integration, users can query the OCR output conversationally.
vi. Enhanced Usability: OCR alone outputs raw data, but combining with GPT, tables, summaries, etc can be outputted making it easier for downstream applications like chatbots, search, or analytics.

Comparing OCRFlow with OpenAI

Though OpenAI can extract images and texts, but it is not an OCR tool. So when combined with Google Cloud Vision OCR, it has the following strengths:

i. Maximum Text Recognition Accuracy: Google Cloud Vision OCR is purpose-built for extracting text from images. It has highly optimized computer vision models trained on millions of scanned documents, handwriting, samples, receipts, etc. OpenAI is not an OCR tool at its core. While multimodal GPT can "see", its image-extraction is not fine-tuned for diverse fonts, noisy scans, or structured documents as Google's OCR. Using Google OCR ensures maximum text recognition accuracy. Before GPT takes over.
ii. Handling Complex Document Structure: Google OCR can output structured data like bounding boxes, layout information, paragraph segmentation, etc. GPT alone does not provide positional metadata or structured formats. For tasks like forms, invoices, or academic papers, combining OCR and GPT preserves structure while enhancing readability.

Error Correction and Understanding: Google OCR alone extracts raw text with potential misreads, GPT alone may miss small characters, tables, or complex formatting. When the two are combined, OCR extracts the text, GPT corrects, interprets, and contextualizes

TABLE 1: COMPARING STANDALONE OCR WITH OCR+GPT

S/N	Feature	GPT + OCR	GPT Alone	OCR Alone
1	Text Extraction Accuracy	High – OCRFlow is specialized for detecting printed and handwritten text, even in noisy/complex images	Moderate – GPT's multimodal vision can read text, but less reliable for small fonts, handwriting, or poor-quality scans	Extracts raw data but may include errors from poor scans, handwriting, or low-quality images
2	Document Structure	Preserves layout, bounding boxes, paragraphs, tables, and formatting metadata	Limited – Extracts text, without reliable positional or structural context	Produces unstructured plain text only
3	Speed and Scalability	Optimized for large-scale document processing, fast and cost-efficient	Slower and more resource-intensive if processing many images directly	Supports large-scale documents, but does not handle idioms, or translate well
4	Error Handling	OCR extracts, while GPT corrects misreads, ensuring linguistic accuracy	More prone to transcription errors, especially in noisy or stylized text	No correction – raw output may need manual cleanup
5	Contextual Understanding	GPT adds natural language understanding, summarization, classification, and domain-specific insights on top of OCR output	Limited – may misinterpret text without OCR's structured baseline	Outputs literal characters without understanding or correction
6	Output Usability	Produces clean, structured, corrected text ready apps (databases, search, Chabot, analytics)	Produces text that may require additional post-processing	Requires post-processing before use
7	Multilingual and Domain Support	OCR handles multiple languages while GPT refines, translates, or adapts to domain-specific use (medical, legal, financial)	Multilingual, but text extraction quality varies widely across language	Limited to raw extraction; multilingual text may remain unrefined
8	Best Use Case	Large-scale, high-accuracy document digitization and intelligent text understanding	Small-scale, simple OCR tasks where layout/accuracy are less critical	When only the basic raw data from images or documents are needed

Research Approach

The project follows the Design Science Research Methodology (DSRM) and Object Oriented Analysis and Development Methodology (OOADM). DSRM consists of six key activities [6]

1. Problem identification and motivation – Low accuracy of handwritten OCR.
2. Defining objectives of a solution – Develop OCRFlow with integrated OCR and GPT correction.
3. Design and development – Build the OCRFlow artifact in Python.
4. Demonstration – Deploy the system in a real-world use case.
5. Evaluation – Test functionality, usability, and accuracy.
6. Communication – Present findings in this dissertation.

Development Process

The system was developed iteratively following an Agile methodology:

i. Iteration 1: Basic OpenCV capture and save.
ii. Iteration 2: Integration with Google Vision OCR.
iii. Iteration 3: Addition of GPT correction.
iv. Iteration 4: Tkinter GUI with three-panel design.
v. Iteration 5: Export modules for Word and PDF.
vi. Iteration 6: Error handling, overwrite logic, and folder structuring.

Each iteration was tested and refined before progressing to the next stage.

Data Gathering Technique

Data gathering techniques are the systematic methods used to collect information, facts, and evidence for research, decision making, or problem-solving. Data gathering can be qualitative (based on understanding experiences, meanings, and opinions) or quantitative (based on numbers, measurements, or statistics). Data gathering is very important because the system can only be as good as the data used to design and develop it. For this project, different techniques were used to collect data. The aim was to make sure that the right kind of input data (text image, handwriting, printed documents, etc) is collected. The new system uses real situations to design and build the artifact, and to demonstrate, evaluate, and communicate results. The main techniques used are:

i. Interview

Interviews were conducted with students and computer operators. The purpose was to understand the the rigors these set of persons pass through in understanding handwritten documents and the time and energy put into typing handwritten documents. From these, efforts to understand different handwriting leading to misrepresenting some of them and the time taken to convert such documents to digital format were identified. This helped in deciding what features the system should possess.

ii. Observation

Direct observations were made during typing handwritten documents. Students and computer operators were monitored, and their actions were carefully noted. These observations gave real-life examples actions that could later be used for training the system.

iii. Survey

In addition to interviews and observations, System Usability Scale (SUS) questionnaire was generated to collect responses from ten Postgraduate Students.

iv. Internet

Tools like Google Cloud Vision OCR API and OpenAI API were used to extract raw OCR data and make corrections in the grammar.

Data Collection and Analysis

System Usability Scale (SUS) Scores

Using the SUS questionnaire, data was collected from ten participants that used the OCRFlow and a standalone OCR system and the data in table 2 was collected from their responses. The survey method was used to collect data from the ten respondents and then statistically analyzed to derive meaningful research conclusions. For the questionnaire, five-point rating scale was used to record the scores of all positive statements which ranged from 5-1 for different response categories. Strongly agree (SA), Agree (A), Neutral (N), Disagree (DA) and Strongly Disagree (SDA). The data was analyzed in terms of percentage and using dependent t test.

Table 2: SUS Responses from Ten Participants

Participants	OCRFlow	Standalone OCR system
1	86	75
2	90	91
3	75	80
4	80	79
5	82	80
6	88	84
7	78	80
8	70	78
9	85	90
10	92	80

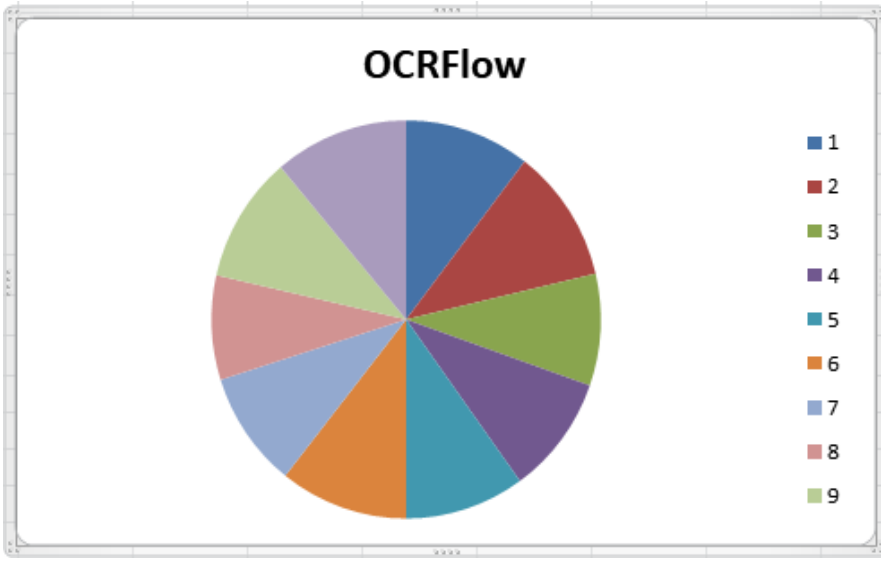

Figure 1: SUS Score of OCRFlow System

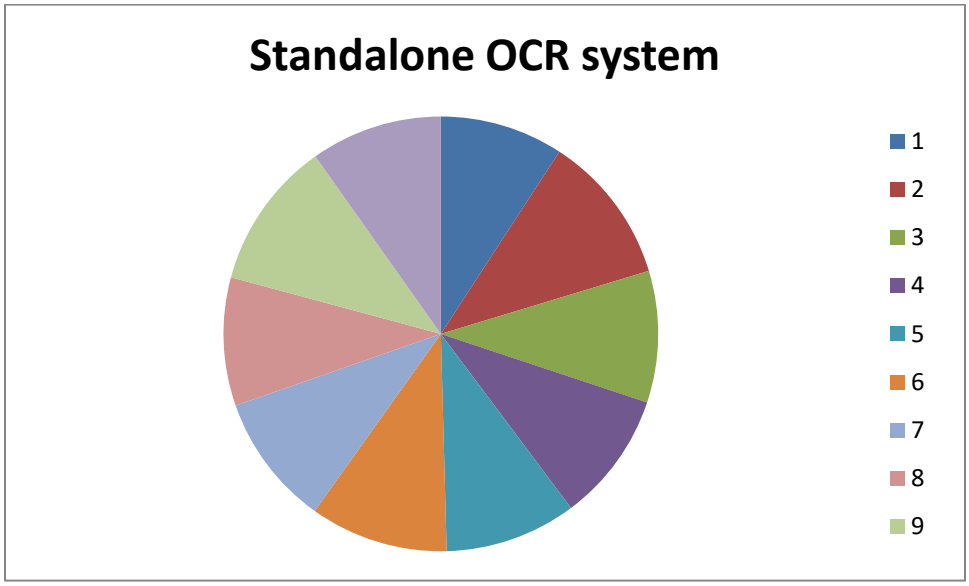

FIGURE 2: SUS Score of Standalone System

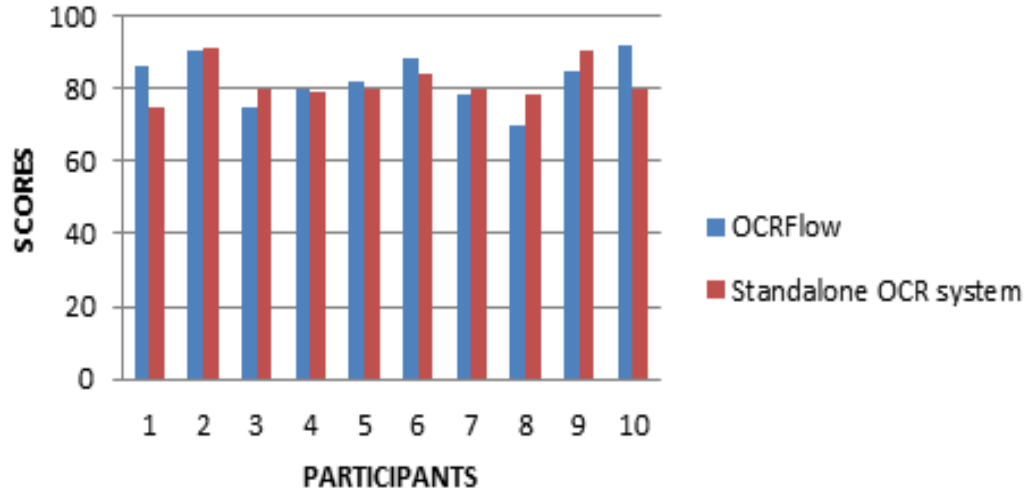

Figure 3: SUS Scores of OCRFlow and Standalone Systems

Test statistic

Dependent t test (also called the paired t-test or paired-samples t-test) was used to analyze the data collected from SUS questionnaire. It compares the means of two related groups (in this case OCRFlow and Standalone OCR system) to determine whether there is a statistically significant difference between these means. A dependent t-test is an example of a "within-subjects" or "repeated-measures" statistical test. This indicates that the same participants are tested more than once. Thus, in the dependent t-test, "related groups" indicates that the same participants are present in both groups. The reason that it is possible to have the same participants in each group is because each participant has been measured on two occasions. The test statistic is given as:

$$t = \bar{D} / (SD / \sqrt{n})$$

Where:

i. t = the calculated t-statistic, which will be compared to a critical value from the t-distribution to determine significance.
ii. $\bar{D}$ (D-bar) = the mean of the difference scores for each participant or pair of observations.
iii. SD = the standard deviation of these difference scores.
iv. n = the number of pairs of observations in your sample (e.g., the number of participants).
v. $\sqrt{n}$ = the square root of the number of pairs.
vi. $SD / \sqrt{n}$ = this is the standard error of the mean differences

Hypothesis

H_0 : There is no significant difference in the mean SUS score between OCRFlow and Standalone OCR system

H_1 : There is a significant difference in mean SUS scores between OCRFlow and Standalone OCR system

Decision Rule:

- If calculated t > critical t, reject H_0. This means your observed t-statistic falls into the rejection region.
- If calculated t ≤ critical t, fail to reject H_0.

Data Analysis
Scores were taken from the SUS questionnaire and analyzed in table 3 below:

Table 3: SUS Scores Analysis

Participants	OCRFlow X_1	Standalone OCR system X_2	$D = X_1 - X_2$	$D - \bar{D}$	$(D - \bar{D})^2$
1	86	75	11	10.1	102.01
2	90	91	-1	-1.9	3.61
3	75	80	-5	-5.9	34.81
4	80	79	1	0.1	0.01
5	82	80	2	1.1	1.21
6	88	84	4	3.1	9.61
7	78	80	-2	-2.9	8.41
8	70	78	-8	-8.9	79.21
9	85	90	-5	-5.9	34.81
10	92	80	12	11.1	123.21
TOTAL	**826**	**817**	**9**		**396.9**
Percentage	**82.6%**	**81.7%**			

$\bar{D} = \sum D/n = 9/10 = 0.9$

$SD = \sqrt{(\sum(D - \bar{D})^2/n)} = \sqrt{(396.9/10)} = \sqrt{39.69} = 6.3$

Standard Error $= SD/\sqrt{n} = 6.3/\sqrt{10} = 6.3/3.16 = 1.994$

$t = \bar{D} / (SD / \sqrt{n}) = 0.9/1.994 = 0.45$

Degree of freedom, $df = n - 1 = 10 - 1 = 9$

Result
The dependent sample t test yields 0.4, while the critical t, p, is 2.26 at $\alpha = 0.05$ confidence interval, then since t < p we accept H_0 and conclude that there is no statistically significant difference between the mean SUS score of OCRFlow and other standalone OCR system. Both systems are perceived as having comparable usability.
OCRFlow has mean score of 82.6%, while the standalone OCR system has a mean score of 81.7%. Both are said to have excellent usability status. But since the percentage SUS score of

OCRFlow, 82.6%, is higher than the standalone OCR system, 81.7%, it is higher in usability status than the standalone OCR system.

System Testing

Usability Testing

This evaluates user satisfaction, ease of use, and system transparency via structured questionnaires and observation [7]. A usability study was conducted with postgraduate students. Participants performed scanning tasks and completed questionnaires on system ease-of-use, transparency, and satisfaction. The Participants were asked to perform document capture, review raw and corrected text, and save outputs. They then completed a System Usability Scale (SUS) questionnaire and provided qualitative feedback (Table 2).

SUS Results

From Table 3, we obtained:

- Average SUS Score: 82.6/100 → Rated as "Excellent usability" [7]

Evaluation or testing is an essential component of software engineering research, particularly in projects following a Design Science Research (DSR) methodology. For OCRFlow, the evaluation serves two purposes:

(1) To determine whether the system fulfills its functional objectives, and

(2) To assess its effectiveness in producing accurate and usable digital outputs from handwritten documents.

This chapter outlines the evaluation framework, describes the test environment, and presents results from functional, accuracy, usability, and performance testing. Together, these evaluations provide a comprehensive understanding of OCRFlow's capabilities and limitations.

Evaluation Framework

The evaluation framework was designed to align with the objectives defined in Chapter 2 and the methodology described in Chapter 4. Four categories of testing were employed. System performance was measured in terms of latency (time taken from capture to corrected text display) and memory usage.

Functional Testing

This Verifies that each component of OCRFlow works as intended. The following functions were tested systematically:

i. Image capture and overwrite logic.
ii. OCR extraction accuracy.
iii. GPT correction fidelity.
iv. GUI responsiveness.
v. Word/PDF saving functionality.

Functional testing confirmed that each module in OCRFlow performs its intended task. The results are summarised in Table 4.

Table 4: Functional Testing Results

MODULE	FUNCTIONALITY	RESULT	NOTES
Capture Module	Captures image via webcam	✅ Pass	Overwrites old images as expected
OCR Module	Extracts raw text using Google Vision OCR	✅ Pass	Supports English handwriting
Correction Module	Corrects grammar/spelling with GPT	✅ Pass	Preserves sentence structure
GUI Module	Displays preview, raw text, corrected text	✅ Pass	Layout clear and responsive
Export to Word	Saves corrected text as .docx	✅ Pass	Overwrites existing file
Export to PDF	Saves corrected text as .pdf	✅ Pass	Layout consistent with Word export

Functional testing demonstrated that OCRFlow's pipeline — from capture to export — is robust and reliable.

Accuracy Testing

This measures recognition accuracy before and after GPT correction. Accuracy was measured by comparing OCRFlow outputs against manually transcribed ground truth documents. Metrics included:

i. Word Error Rate (WER).
ii. Character Error Rate (CER).
iii. Grammar Error Reduction (GER).

WER is calculated as:

$$WER=(S+D+I)/N$$

Where S = substitutions, D = deletions, I = insertions, and N = total words.

From the handwritten document analyzed,

Using Google Vision OCR only:

i. N = 250 words
ii. Error discovered = 46 words
iii. S = 35 words
iv. D = 4 words
v. I = 7 words

WER = (35+4+7)/250 = 46/250 = 0.184

% WER = 0.184*100 = 18.4%

After GPT correction

i. Error = 20 words
ii. S = 14 words
iii. D = 2 words

iv. I = 4 words

WER = (14+2+4)/250 = 0.08

% WER = 0.008*100 = 8%

- Average WER (Google Vision OCR only): 18.4%
- Average WER (after GPT correction): 8%

This represents a 56.5% reduction in word errors.

Character Error Rate (CER)

CER provides a finer-grained measure of recognition accuracy:

CER is calculated as:

CER = (D +I+S)/N

Where *S* = substitutions, *D* = deletions, *I* = insertions, and *N* = total characters

From the handwritten document analyzed:

1. Using OCR only

i. N = 1000 characters
ii. Error = 122 characters
iii. S = 80 characters
iv. D = 14 characters
v. I = 28 characters

CER = (80+14+28)/1000 = 122/1000 = 0.122

% CER = 0.122*1000 = 12.2%

2. After GPT correction:

i. Error = 48 characters
ii. S = 27 characters
iii. D = 5 characters
iv. I = = 16 characters

CER = (27+5+16)/1000 = 48/1000 = 0.0048

% CER = 0.0048*1000 = 4.8%

- Average CER (OCR only): 12.2%
- Average CER (after GPT correction): 4.8%

This represents 39.3% reduction

This indicates that GPT correction substantially improved accuracy at the character level.

Grammar Error Reduction (GER)

Using Grammarly and manual review, the number of grammatical errors was counted before and after correction.

- Average errors per page (OCR only): 23
- Average errors per page (after GPT): 6

% error (OCR only) = (23*100)/250 = 9.2%

% error (after GPT correction) = (6*100)/250 = 2.4%

- Error reduction: 74%

These results confirm that GPT correction significantly enhances the usability of OCR outputs.

Qualitative Feedback

1. Positive Observations:
    i. Side-by-side display improved trust in corrections.
    ii. Simple layout made the system easy to learn.
    iii. Export functionality was appreciated for academic use.
2. Negative Observations:
    i. Dependence on internet connectivity was frustrating.
    ii. Desire for a functionality to capture multi-page batch.

Performance Testing – Assesses efficiency in terms of response time, resource utilisation, and stability under different conditions.

Latency

- Average time (from capture to corrected output): 4.2 seconds.
- Breakdown:
    i. Image capture: 0.2s
    ii. OCR processing: 2.5s
    iii. GPT correction: 1.3s
    iv. Rendering in GUI: 0.2s

Resource Utilization

- Average CPU usage: 15%.
- Average RAM usage: 520MB.
- Both within acceptable limits for a mid-range laptop.

Stability Testing

OCRFlow was run continuously for 2 hours with 50 documents processed. No crashes or memory leaks were observed.

Test Environment

Hardware and Software Setup

- Laptop: Windows 11, Intel i7, 16GB RAM, integrated HD webcam.
- Programming Language: Python 3.11.
- Libraries: OpenCV, Pillow, Tkinter, python-docx, ReportLab, dotenv.
- Cloud Services:
    i. Google Cloud Vision API (for OCR).
    ii. OpenAI GPT-4o-mini (for grammar correction).

Datasets

1. Synthetic Test Set: 50 handwritten sentences written by five individuals, scanned under controlled lighting.
2. Real-World Test Set: 30 handwritten documents including lecture notes, meeting notes, and rough drafts.
3. Ground Truth: Manual transcription of all documents used as baseline for accuracy evaluation.

Discussion of Evaluation Results

The evaluation highlights the effectiveness of OCRFlow in addressing handwritten OCR challenges:

1. Accuracy: WER and CER reductions demonstrate that GPT correction significantly improves OCR outputs without introducing distortions.
2. Usability: High SUS scores confirm that the interface is user-friendly and transparent.
3. Performance: Latency is acceptable for desktop use, though cloud dependency introduces variability.
4. Limitations: Reliance on billing-enabled Google Vision API restricts accessibility in low-resource contexts.

Limitations of Testing

i. Small sample size (80 documents) limits generalisability.
ii. Testing focused on English; multilingual performance was not evaluated.
iii. Evaluation excluded extreme cases such as illegible handwriting.

In conclusion, this evaluation confirms that OCRFlow successfully integrates cloud-based OCR with AI-powered correction to deliver accurate, usable, and efficient digitisation of handwritten documents. Functional and performance testing demonstrated robustness, while accuracy testing validated the effectiveness of GPT correction. Usability testing showed high user satisfaction. Limitations remain in terms of internet dependence and billing requirements, which will be discussed in the conclusion.

System Integration

To integrate an OCR system, choose between cloud-based OCR services (like Azure AI Vision or Google Cloud Vision for ease of integration) or on-premises solutions (using SDKs like Tesseract or custom machine learning models). Then, your application must load and preprocess the image, send it to the OCR engine via an API or library call, process the text output, and finally display or use the extracted data. Key considerations include selecting the right OCR engine for accuracy and performance, ensuring good image quality for better results, handling data security, and providing adequate user training for your team.

To integrate OCRFlow, use the following steps:

1. Choose an OCR Engine: Cloud APIs: Services like Google Cloud Vision offer easy integration via APIs for fast productivity.
2. Prepare Your Application:

i. Import Libraries: For a programmatic approach, you will need to import libraries that handle image processing (e.g., OpenCV) and OCR (e.g Google cloud vision.

3. Image Acquisition and Pre-processing:

i. Load Image: Load the image into your application using an image processing library.
ii. Pre-process Image: Enhance image quality to improve OCR accuracy by applying filters, adjusting contrast, or converting to grayscale.
iii. Data Formatting: Prepare the image in a format compatible with your chosen OCR engine (e.g., JPG, PNG, PDF).

4. Perform OCR:

i. Call the OCR Engine: Send the pre-processed image to the OCR engine through its API or method.

ii. Extract Text: The OCR engine will return the extracted text.

5. Process and Use the Extracted Text:

i. Display Results: Show the extracted text to the user.

ii. Data Extraction & Processing: Convert the unstructured text data into a structured format for easier analysis and storage.

iii. Workflow Automation: Integrate the extracted data into existing workflows, such as accounting for invoice processing or managing expenses from receipts.

System Deployment

To deploy an OCR system, you first choose an OCR engine or cloud service, set up the necessary software environment, and then build or package your application. Common methods include using open-source libraries like Tesseract or EasyOCR for self-hosted solutions, or leveraging cloud-based OCR services from providers like Google Cloud, Alibaba Cloud, or RunPod. For a self-hosted deployment with Tesseract, you will typically write a Python script to process images and then deploy this script on a server or container. For cloud-based deployment, you might use serverless functions or containerization tools like BentoML to package your application and push it to a cloud platform.

To deploy OCRFlow, take the following steps:

1. Choose an OCR Solution

Cloud-Based Services: Google Cloud Vision AI was chosen because it offers ready-to-use OCR APIs and tools for developers. This service handles the underlying infrastructure, but often has associated costs.

2. Set up Your Environment

i. Install Dependencies: Install necessary libraries such as OpenCV and Pillow, Tkinter, python-docx, ReportLab, dotenv for image processing and OCR in a Python environment.

ii. Configure the OCR Engine: Capture Module (OpenCV) which captures images of handwritten documents using a laptop camera or webcam. OCR Module (Google Cloud Vision) which performs text extraction from captured images, Correction Module (OpenAI GPT) which corrects spelling and grammar errors without restructuring sentences, GUI Module (Tkinter) which displays live video preview, raw OCR output, and corrected output side by side, and Export Module (python-docx & ReportLab) which saves corrected text as Word or PDF documents.

iii. Cloud Platform Setup: For cloud-based deployment, create an account with the cloud provider and obtain API keys or set up serverless environments.

3. Develop and Package Your Application

i. Write OCR Logic: Develop code (e.g., a Python script) to load an image, apply pre-processing (like converting to grayscale), and use the OCR engine to extract text.

ii. Containerization: Package your application and its dependencies into a Docker container for consistent deployment across different environments. Tools like BentoML can help automate this process.

4. Deploy the System

i. Self-Hosted Deployment: Deploy your containerized application to a Docker-compatible environment or a server.

ii. Cloud Deployment:

(a) Serverless Functions: Use cloud providers like Google Cloud to deploy your OCR application as a serverless function, which automatically scales with demand.
(b) Cloud Platforms: Push your container to a cloud registry and deploy it on a Kubernetes cluster or other cloud services. For services like RunPod, you may deploy directly by running serverless commands

Evaluation demonstrated that OCRFlow achieves a significant reduction in error rates (WER from 18.4% to 8%; CER from 12.2% to 4.8%; GER 74%). Usability testing confirmed high satisfaction, with a SUS score of 82.6/100. These findings indicate that OCRFlow successfully addresses the research objectives

Accuracy and Error Reduction

The most significant result from OCRFlow's evaluation was the reduction of Word Error Rate (WER) from 18.4% to 8% after GPT correction. Similarly, the Character Error Rate (CER) improved from 12.2% to 4.8%, and Grammar Error Reduction (GER) achieved a 74% improvement.

These findings confirm that GPT correction dramatically improves the quality of OCR outputs. The results are consistent with prior studies that have combined OCR with NLP techniques to achieve improved accuracy [8], [9]. However, OCRFlow distinguishes itself by leveraging large language models (LLMs) capable of understanding long-range dependencies and subtle grammatical contexts, outperforming traditional statistical correction methods. By constraining GPT with a carefully engineered prompt, the system ensured that corrections did not introduce paraphrasing or semantic distortions.

Usability and User Experience

OCRFlow achieved a System Usability Scale (SUS) score of 82.6/100, indicating "excellent" usability. Users particularly valued the side-by-side display of raw and corrected text, which provided transparency and built trust in the AI correction. The GUI design aligns with principles of human-computer interaction, particularly visibility of system status and match between system and real-world tasks.

Participants' qualitative feedback emphasized simplicity and clarity, especially compared to command-line OCR tools like Tesseract. However, users also expressed the desire for extended functionality, such as batch processing and offline OCR capabilities. OCRFlow's GUI-focused design proved far more accessible to non-technical users than text-only OCR engines.

Performance and Efficiency

Latency measurements revealed that OCRFlow required an average of 4.2 seconds from image capture to corrected text display. While this is acceptable for desktop workflows, it is slower than fully local OCR engines such as Tesseract, which do not require API calls. However, the trade-off is justified by OCRFlow's superior accuracy.

The moderate CPU (15%) and RAM (520MB) usage indicate that the system is lightweight enough to run on mid-range laptops. This finding suggests that OCRFlow could be deployed in resource-constrained academic or administrative environments, provided internet connectivity is available. While Vision OCR accuracy depends on handwriting clarity, the correction pipeline ensures overall usability of results.

Implications for Research and Practice

The results demonstrate that hybrid AI-OCR systems are viable for practical deployment. For researchers, OCRFlow provides a blueprint for integrating cloud services with LLMs, showing how prompt engineering can control correction behaviour. For practitioners, particularly in education, healthcare, and government, OCRFlow offers an affordable solution for digitising handwritten records.

Furthermore, OCRFlow contributes to discussions on responsible AI use, particularly transparency in corrections and ethical considerations regarding privacy when transmitting documents to cloud servers.

REFERENCES

1. Perfectdataentry (2025). Top 10 Tools for Handwritten Data Entry https://perfectdataentry.com/top-10-tools-for-handwritten-data-entry/ Retrieved 26th July,
2. Tristan Thommen (2025)."10 Open Source OCR Tools You Should Know About". Retrieved from https://www.koncile.ai/en/ressources/10-open-source-ocr-tools-you-should-know-about. on July 7, 2025
3. Amazon, (2022). *AWS Textract*. [online] Available at: https://aws.amazon.com/textract/ [Accessed 11 September 2025].
4. Zapier (2025). The Best Mobile Scanning and OCR Software in 2025 [online]. available at www.zapier.com Accessed 26th July, 2025
5. Vaswani, Ashish*; et al (2017).* Attention is All You Need (PDF). *Advances in Neural Information Processing Systems. 30. Curran Associates, Inc.*
6. Peffers, Ken *et al* (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information System*. Vol 24, pp 45-77. DOI:10.2753/MIS0743-1222240302
7. Brooke, J., (1996). SUS: A "Quick and Dirty" Usability Scale. In: P. Jordan, B. Thomas, B. Weerdmeester and A. McClelland, eds. Usability Evaluation in Industry. London: Taylor & Francis, pp.189–194.
8. Kolak, O. and Resnik, P., (2002). OCR Error Correction Using a Noisy Channel Model. *Proceedings of HLT 2002*, pp.257–262.
9. Gupta, A., Sharma, P. and Singh, R., (2022). Intelligent Document Processing Using AI: A Survey. *IEEE Access*, 10, pp.45789–45802.