OULUN AMMATTIKORKEAKOULU

# IMPLEMENTING IMAGE RECOGNITION TECHNIQUE TO FIND AND READ OBJECTS OR DATA FROM IMAGES

Moeez Khan
Bachelor's Thesis
Spring 2025
Bachelor of Information Technology
Oulu University of Applied Sciences

# ABSTRACT

Oulu University of Applied Sciences
Degree Programme in: Information Technology

Author: Moeez Khan
Title of thesis: Image Recognition System to Find and Read Objects or Data from Images
Thesis supervisor: Juha-Matti Huusko (OAMK), Panu Vuokila (Organisation)
Term and year of completion: Spring 2025
Pages: 31

Data extraction from images and Image recognition are important tasks in many fields, enabling automatic visual information processing such as searching

This project implements a simple web application that allows a user to upload an image, which is then processed using an AI-based method to identify objects and extract textual data. The methodology involved building a web page in HTML/CSS for the image upload interface, using a Flask Python backend to receive the image and send it to the ChatGPT vision-capable API. The returned objects and text are again displayed to the user by the frontend.

The project demonstrates the feasibility of using large language models for image analysis, highlights practical challenges (such as image size limits and cost), and suggests future improvements.

## ACKNOWLEGDEMENTS

This thesis, "Image Recognition System to Find and Read Objects or Data from Images," was completed at Oulu University of Applied Sciences in Spring 2025. The topic was provided by Matti Uusitalo at Ouda Oy, and I am thankful for the opportunity to work on a subject that closely reflects my professional interests and future goals.

I would like to express my sincere gratitude to my supervisors, Juha-Matti Huusko from Oulu University of Applied Sciences and Panu Vuokila from Ouda Oy, for their continuous support, guidance, and encouragement throughout this project. I am also grateful to the team I worked with for offering a supportive environment and for sharing their knowledge during the process. Their help was invaluable in turning this work into a meaningful experience.

Finally, I want to thank everyone who supported me personally along the way. Your encouragement made a real difference during this journey.

---

# CONTENTS

# 1 INTRODUCTION

Computers analyze and interpret visual information through a technique called image recognition. To put it simply, image recognition software can recognize words, scenes, or objects in an image. For instance, it can identify whether a picture has a person, a vehicle, or a sign, or it can extract text from a document image. This enables machines to carry out tasks that would otherwise require human vision in a practical sense. Given that we have several printed books and newspapers covering a variety of topics, there is currently a rising need for software systems that can identify characters in computer systems when information is scanned from paper materials.

Keechul Jung et al (2004) in “Text information extraction in images and video: a survey” stated that, text in images and videos contains essential information that makes it easier for visual material to be automatically annotated, indexed, and organized. This information is extracted by a multifaceted procedure that includes numerous crucial processes, such as: 1. Detection: Locating regions that might contain text. 2. Localization: specifying the text's precise borders. 3. Tracking: keeping an eye on text throughout video frames. 4. Extracting the text so it may be examined. 5. Enhancement—making the text more readable. 6. Recognition—the process of turning visual text into text encoded by a machine. However, there are many difficulties in this process. Variability in text properties, like alignment, size, font style, and orientation, might make extraction more difficult. Furthermore, the written content is frequently obscured by intricate backgrounds and low visual contrast, making it difficult to see and identify.

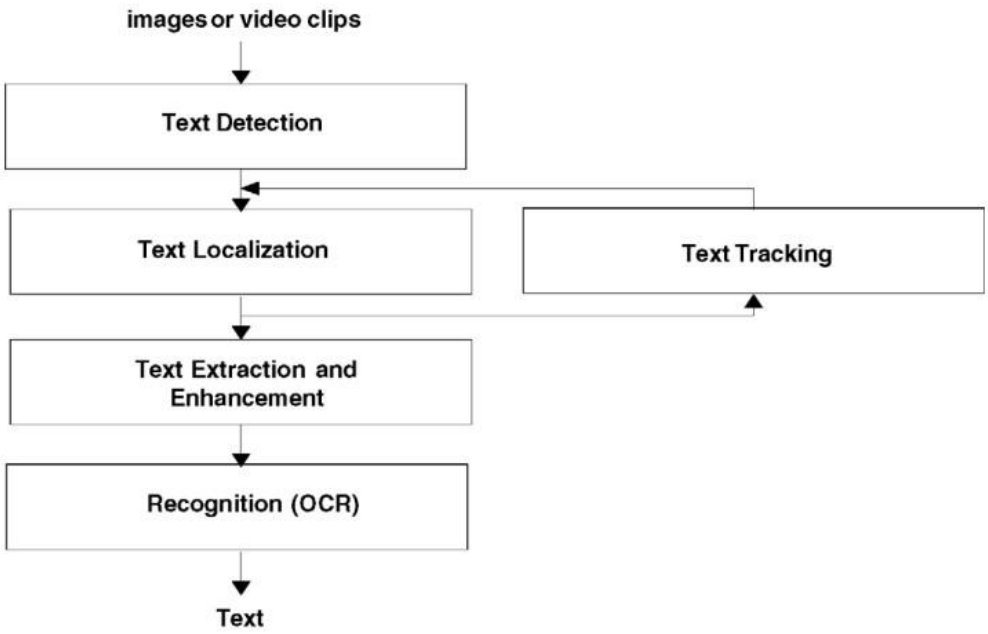



*FIGURE 1. Architecture of TIE System According to (Keechul Jung 2004)*

These complexities render the task of automatic text extraction particularly challenging and require advanced techniques and algorithms to achieve satisfactory results (Keechul Jung 2004).

There is currently a significant demand for converting information from paper documents into digital storage, which allows for later retrieval through a search process. One straightforward method to achieve this is by scanning the documents. When we scan these documents using a scanner, they are saved as images on the computer. However, these images, which contain text, cannot be edited by users. As a result, it becomes challenging for the computer system to read and search through the content line by line and word by word.

The information libraries which began their life with pure text compositions now add more value through multimedia elements which include images and videos as well as audio components. All these information systems require automated systems that can effectively perform efficient multimedia component retrieval. The automatic detection and segmentation and recognition of texts within images presents itself as a valuable semantic resource. (Wernicke 2002.)

Modern society places immense value on the automatic process of image annotation along with indexation and structure building according to content. (Wernicke 2002.) Therefore, content-based image annotation, structuring, and indexing of images is of great importance and interest in today's world.

Image recognition is used in many real-world applications: facial recognition on social media, analyzing medical scans, autonomous vehicles identifying road signs, and even translating text in photographs. Banks use OCR to process checks and extract account numbers, amounts, and signatures. Google's Translate app can take a photo of foreign text and instantly translate it by scanning the characters. Such use cases demonstrate that automating visual data capture can save manual effort and make information more accessible. In business, automating data entry from forms, receipts, and invoices can greatly improve efficiency. In assistive technology, reading text aloud from photos helps visually impaired users. Hence, there is a strong motivation to develop tools that can "find and read" objects or text in images automatically.

This thesis explores the implementation of an image recognition technique using modern AI. Specifically, we build a web application in which users upload an image via a browser (an HTML/CSS interface). The image is then sent to a visioncapable ChatGPT API (GPT-4 with image input), which returns information about the image. The Flask framework is used on the server side to handle uploads and display results. The purpose of the project is to demonstrate how such a system can be built and to evaluate the feasibility and limitations of using a large language model (ChatGPT) for image-based tasks. The remainder of the thesis covers the background of image recognition and OCR, the design and implementation of the system, and an evaluation of results.

## 1.1 INTRODUCTION TO TECHNOLOGIES

Here the main technologies used are introduced.

- **API (Application Programming Interface):** API functions as a software connection that helps computer programs exchange data. It enables communication between programs. An API specifies how different parts of the software should connect and work together to supply services to other software components.
- **ChatGPT API:** Through its ChatGPT API technology OpenAI lets developers utilize GPT3.5/GPT-4 language models within their platforms. It enables sending prompts or images to the GPT-4 language model and receive text output. (OpenAI 2023.)
- **CSS (Cascading Style Sheets):** Cascading Style Sheets serves as a declarative style language to define HTML document presentation. HTML documents display their visual design using simple style descriptors. The CSS language lets users define web page style properties through its controls. (Mozilla Developer Network 2025.)
- **Flask:** A lightweight web application framework for Python. Flask uses the WSGI toolkit and allows rapid development of web servers and APIs with minimal setup. It is commonly used to build simple web services and interactive applications. (Flask 2025.)
- **HTML (HyperText Markup Language):** HTML (HyperText Markup Language): The standard markup language for creating web pages. HTML helps determine how content appears within a document using tags and elements. (Mozilla Developer Network 2025.)
- **OCR (Optical Character Recognition):** The electronic or mechanical conversion of images (such as scanned documents or photos) of printed or handwritten text into machine-encoded text. OCR is widely used to digitize text so it can be edited, searched, or processed by computers

## 1.2 OBJECTIVES

The objectives of the implementation are listed below.

- To design and develop a web-based application that allows users to upload image files through a user-friendly interface.
- To utilize the ChatGPT (GPT-4 Vision) API for analyzing and extracting information (objects and text) from images.
- To implement a Flask-based backend for handling image uploads, API communication, and result display.
- To evaluate the feasibility and effectiveness of using a large language model (LLM) for image recognition and OCR tasks.
- To test the system with different types of images (e.g., documents, signs, labels) and assess the accuracy of extracted content.
- To explore limitations, challenges, and possible improvements in combining AI APIs with web technologies for real-world applications.
- To document the development process and provide a working proof-of concept for educational or research purposes.

# 2 BACKGROUND STUDY

As discussed earlier, text extraction or text recognition from images is still an active research area. Many researchers have proposed different ideas to address the issue of extracting text from images. Each approach tries to address this issue differently. In the upcoming section, we have proposed a detailed study to discuss different approaches that handle the issue related to text recognition and extraction from images.

## 2.1 History of Image Recognition and OCR

The field of image recognition has roots in early pattern recognition and optical scanning devices. One of the earliest known devices was developed in 1914 by Emanuel Goldberg, who built a machine that could read printed characters and convert them into telegraph code.

Goldberg created what many consider to be the initial electronic document retrieval system after advancing from his previous work. Businesses used microfilm as their record storage solution in the 1920s yet fast access to particular records proved nearly impossible among film spools. During this development Goldberg adapted current technologies into a system which combined a photoelectric cell and film-projector to detect patterns. The US patent for his ‘Statistical Machine’ which IBM acquired became one of their products. (Britton 2022.)

Edmund Edward Fournier d'Albe, an Irish writer, inventor, and scientist, showed off an unusual machine at the Optical Society Convention in London on June 25, 1912. He called it an “exploring optophone," and his remarkable claim was that it allowed people who were completely blind to “hear" light. (Thomas 2021.)

In 1974, Ray Kurzweil founded Kurzweil Computer Products, Inc. and developed the first "omni-font" optical character recognition (OCR) technology. This innovative technology allowed computers to read and recognize printed or typed

characters, regardless of font style or print quality. Building on this breakthrough, he created the Kurzweil Reading Machine in 1976, which was the first device capable of reading printed and typed documents aloud. (Lemelson.MIT 2019.)

*FIGURE 2. Kurzweil Reading Machine (Kurzweil Technologies 2025)*

In the late 20th century, optical character recognition (OCR) began to gain commercial traction. The early OCR systems required individual training for each specific font and were limited to processing one character at a time. By the 1990s, software such as Cuneiform and Tesseract began to incorporate neural network techniques. Currently, modern OCR engines—like Tesseract and commercial solutions such as ABBYY FineReader—utilize advanced image processing and machine learning methods to achieve high levels of accuracy across a variety of fonts and languages. (ABBYY 2023.) Today's OCR technology is capable of preserving the original page layout and formatting, enabling the digitization of text for searchability and analysis. These advancements have been largely propelled by research in computer vision and the rise of deep learning.

Chandni Kaundilya et al in the goal of the OCR System, automated text extraction from images aims to automatically recover image contents and visual information for retrieval purposes. A system must apply different algorithms through optical character recognition to meet this requirement. Google now owns Tesseract as the optimal and accurate optical character recognition engine in the market that originally received development at HP Labs. This paper reveals an approach to text extraction from images that includes text localization, segmentation and binarization processes. Text detection identifies image parts including text and then text localization identifies precise text position followed by text segmentation separating text from its background and finally text extraction becomes possible using binarization which transforms coloured images to black and white format. An applied recognition method converts binary images into ASCII text format. Text extraction is used in creating e-books from scanned books, image searching from a collection of visual data, etc. (Chandni Kaundilya & Diksha Chawla 2019.)

Image recognition, more generally (beyond just reading text), has a rich and extensive history within the domain of computer science. The roots of computer vision can be traced back to the 1960s and 1970s, a period marked by the emergence of foundational algorithms such as edge detectors and feature-based methods, which aimed to enable machines to interpret visual data. A landmark was the development of convolutional neural networks (CNNs) by Yann LeCun and colleagues in the late 1980s and early 1990s.

Before CNNs, the standard methodology for training neural networks to classify images involved flattening the images into a linear array of pixels. This flattened representation was then fed into a conventional feed-forward neural network, which would ultimately output a classification for the image. However, this approach had a critical flaw: it completely disregarded the spatial data and essential features present in the original image, leading to a loss of valuable contextual information necessary for accurate classification.

In 1989, Yann LeCun and his team unveiled Convolutional Neural Networks, which have served as the foundation of Computer Vision research for the past 15

years. Unlike feedforward networks, CNNs preserve the 2D nature of images and are capable of processing information spatially. (Biswas 2024.)

CNNs have significantly enhanced the accuracy and efficiency of image classification systems, paving the way for advancements in numerous fields, from autonomous vehicles to medical imaging and beyond.

The modern resurgence of deep learning exemplified by AlexNet in 2012—significantly enhanced image classification and object detection capabilities. The new features that AlexNet brought to life now serve many different business fields and industries. In computer vision technology AlexNet now performs image analysis work, Classification, object detection, and segmentation. Healthcare applications medical teams use AlexNet-related models to find important findings in medical images like abnormalities in X-rays, CT scans, and MRI images. In retail and manufacturing, Using AlexNet-based models helps companies detect defective products in their production lines through image analysis. (Fahey 2024.)

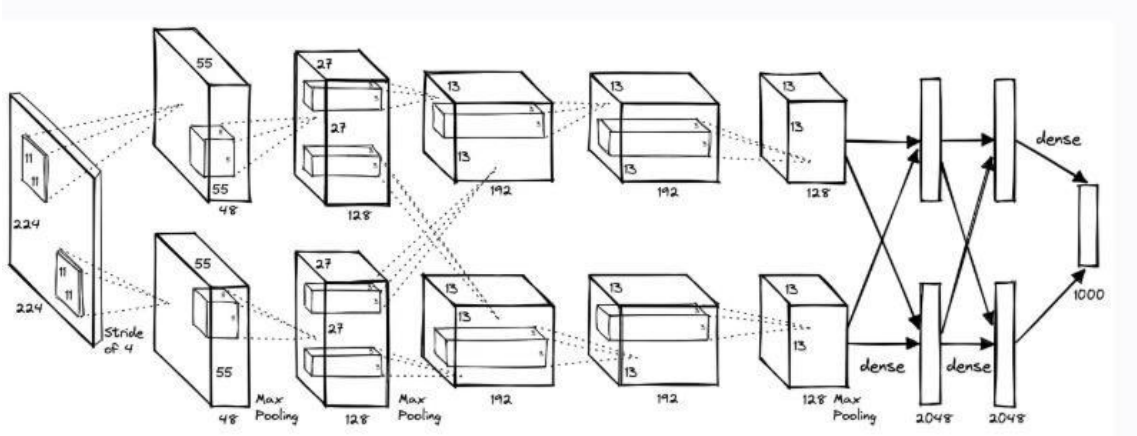



*FIGURE 3. Network Architecture of AlexNet (Pinecone s.a.)*

Today's leading image recognition systems typically utilize deep convolutional neural networks (CNNs) or transformer-based models to classify images (by assigning labels) and to detect and localize objects. For instance, models like YOLO ("You Only Look Once") can identify objects such as cars, pedestrians, and animals in real-time video streams by drawing bounding boxes around them. These tools are widely used in various applications, including surveillance and medical image analysis.

## 2.2 Modern Vision Model and GPT

Large machine learning systems have achieved significant progress by understanding multiple types of data called modalities. GPT-4 from OpenAI represents the latest breakthrough in this field. Its modern design presents as an advanced model that handles both visual and words before delivering organized sentence text. GPT-4 differs from earlier versions because instead of just handling texts this model works equally well with images to generate detailed written responses about what it sees. (Open AI 2023.)

For instance, when provided with an image, GPT-4 can discern various elements within that image and articulate a detailed description or offer insights based on its understanding. This capability transforms the way users can interact with the model, facilitating a more dynamic and intuitive experience.

The Microsoft Azure documentation confirms this breakthrough by explaining that "large multimodal models in vision-based chat can translate images into text responses." Developers now design AI systems with multiple data-handling capabilities as part of a growing AI industry movement to make these systems more useful for users.t. (Microsoft Azure 2025.)

Ashish Shenoy and colleagues, in their work "Lumos: Empowering Multimodal LLMs with Scene Text Recognition," propose an innovative and efficient system for Scene Text Recognition (STR). They claim it is one of the first intelligent multimodal assistants with strong text understanding capabilities, and it is compatible with various devices. Their comprehensive evaluation demonstrates that this hybrid approach—combining on-device STR with on-cloud multimodal large language models (MM-LLM)—outperforms existing methods in terms of accuracy. Furthermore, the proposed system meets stringent requirements for latency, size, memory, power, and computational resources necessary for on device deployment. (Ashish Shenoy 2024.)

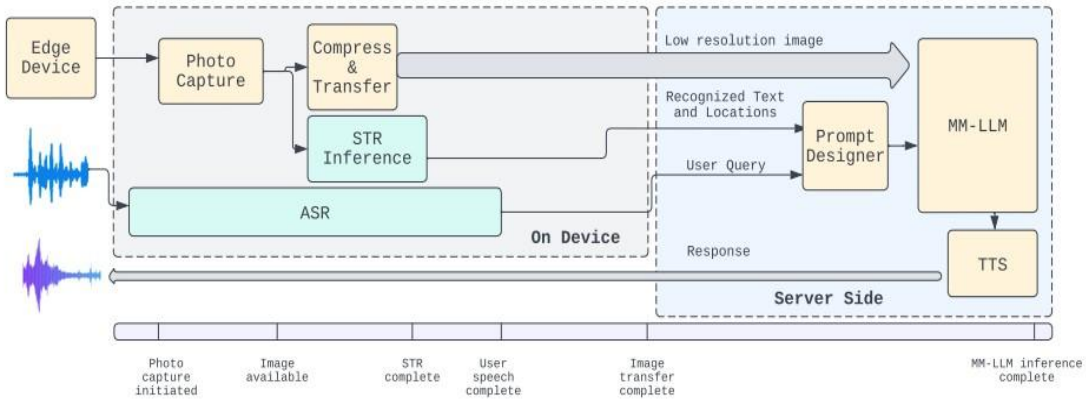



*FIGURE 4. Overall Architecture of Lumos (Ashish Shenoy 2024)*

M.S. Akopyan et al. proposed a system that outlines text extraction pipeline to address the issue of text extraction from pictures. Incoming photos are divided into groups in this pipeline, and each group is subjected to particular preprocessing methods. The processed images are then used as input for the OCR (Optical Character Recognition) engine. Additionally, post-processing may be applied to the output generated by the OCR engine. The researchers collected a dataset of images from a social network and conducted experiments on this dataset. The proposed pipeline effectively extracts text information from social media, especially where images are of poor quality or contain a mix of text and non-text elements. (Akopyan 2019.)

Mehar Prateek Kalra et al., in their work titled "LLM Powered HTR: Integrating Handwritten Text Recognition System with Large Language Model," discuss the importance of handwritten text recognition (HTR) systems in converting handwritten text images into machine-readable formats. These systems are utilized in various applications, including handwriting interpretation, signature recognition, and the analysis of ancient scripts.

Given the complexity and variability of handwritten text, the accuracy of HTR systems is crucial. However, these systems often face challenges that lead to errors in recognition. The authors propose a solution to enhance the accuracy of HTR systems by integrating Large Language Models (LLMs) as a postprocessing

correction unit. LLMs, which are trained on vast amounts of language data, have demonstrated high efficiency in processing natural language tasks.

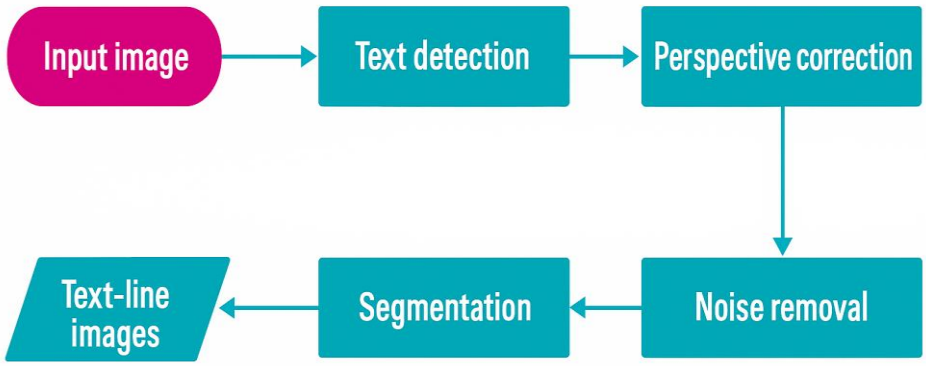



*FIGURE 5. Flow Chart for Handwritten Text Recognition (Mehar Prateek Kalra 2024)*

Their approach suggests that by leveraging the capabilities of LLMs with HTR systems, a significant improvement in recognition accuracy—up to 10%—can be achieved (Mehar Prateek Kalra 2024).

# 3 METHODOLOGY

The Proposed system has three main components: a web interface (front end), a backend server, and the ChatGPT image-processing API. The methodology can be described in sequential steps.

## 3.1 User Interface

The HTML form presents a friendly interface for users to either select or drop images into the form for submission. User experience easy usability through an interactive interface, using which file uploads become simpler. A CSS application provides an attractive layout to enhance the quality of the user interface in the form. Data transmission between the server and client operates with multipart/form-data encoding to process submitted files properly and other form data. Using this integrated system, the user can interact easily and fast and file moving can also be secure.

## 3.2 Flask Backend

Flask framework stands as the essential element in the second step of this proposed system. When the form is submitted, it is Flask's responsibility to handle the submitted request. In Flask's code, an endpoint is defined with the GET and POST methods, which is responsible for the retrieval of the uploaded image. The server code retrieves the uploaded image, and Flask reads the binary image data. At this point, the server has the data and is ready to forward it.

## 3.3 Calling ChatGPT API

This was the core component of the methodology. In this component the Open AI's ChatGPT API is used to analyze the image. The approach is to use **openai.ChatCompletion.creat** with a vision enabled model. In the message content there is a text prompt which is simply instruction for the LLM.

### 3.4 Processing the Response

In this step of the system, the ChatGPT API returned the processed response in JSON format. The response includes the assistant's message about the image. The content field usually contains the description. It includes information about the detected object or the recognized text from the image. The content is parsed as plain text. If the API returns the response as structured JSON, it can be parsed as well, but it is optional. In this case, the response is treated as text.

### 3.5 Displaying Results

In this last step of the implemented project, the final result is displayed to the user. The processed response is sent back to the HTML template, which renders the analysis. In the result page the original uploaded image and the detected information about the image are displayed.

The Flask template renders {{response_text}}, which is the text from the ChatGPT response. This gives the user visible feedback on what was found in the image.

In summary, the methodology follows a typical client-server pattern with the novel step of passing images to an AI model. Key points: using HTML/CSS for the front end, Flask for the backend server, and ChatGPT (GPT-4 vision) API for image analysis. The system architecture can be diagrammed as:

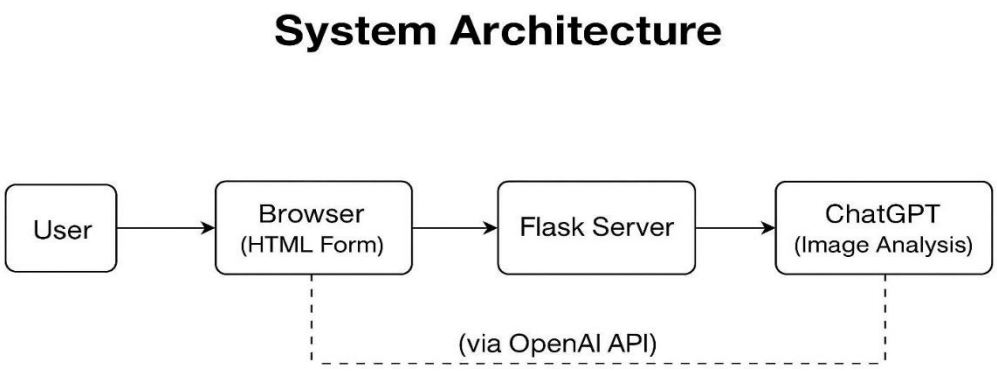



*FIGURE 6. System Architecture*

# 4 TOOLS USED IN THE PROPOSED SYSTEM

In the development of this project, several essential tools and technologies were used to build a smooth, functional, and interactive image recognition system. Each tool has a specific role in making the system work as desired, from the frontend interface to the backend logic and AI-powered image analysis.

## 4.1 Python 3.1

**Why was it used?**

Python was chosen as the core programming language for this project because of its simplicity, readability, and rich ecosystem of libraries and frameworks. Python Version 3.11 incorporates performance improvements through runtime operation optimization and responds directly to user demands.

**How was it used?**

Python serves as the backbone of the entire application. Users send images to the server logic, which triggers the AI model to process the content before returning the results to clients. Integrating different

components into one solution is what Python excels at through its support for APIs, along with web frameworks.

## 4.2 Flask 2.1

### Why was it used?

Flask is a lightweight yet powerful web framework for Python. Using Flask the development of web applications, without the overhead of a full-stack framework like Django, is easy and quick. Flask is simple, easy and highly flexible for small to medium-sized projects like this one.

### How was it used?

The server operations allowed the web interface to upload images and perform API communication with ChatGPT due to Flask. Using Flask, the system was able to interact easily by managing the routing and displaying responses from the AI server to the front end.

## 4.3 OpenAI API

### Why was it used?

Instead of training up an image recognition model from scratch, the OpenAI API—i.e., GPT plus image input functionality—was used to tap into a discrete state-of-the-art artificial intelligence. It also saved a huge amount of development time and enabled powerful image and text detection.

### How was it used?

The uploaded image is then sent to the ChatGPT model via the API. The model, upon receiving this information, interprets the visual data and offers valuable insights or reads text from the image from its database.

## 4.4 HTML

### Why was it used?

Standard language for creating web pages - HTML. It was the obvious choice for constructing the structure of the image upload page in the browser.

### How was it used?

HTML set the basic structure of the web page, into which the form popped up for others to upload images. It established the form of buttons, headings, and other elements that the user interacts with them.

## 4.5 CSS

### Why was it used?

CSS is used to make the web page look nice. It helps govern the layout, colours, margin, and entire climatic condition of the HTML centre.

### How was it used?

CSS styles were added to give the user interface a cleaner and more user-friendly look. With a nicely laid out upload page, users can have a more enjoyable experience interacting with the system to provide a more professional product in the end.

# 5 IMPLEMENTATION

The project was developed using Python with Flask, the OpenAI API client, and standard web technologies. The process involved setting up the Flask application, designing the web interface, integrating the ChatGPT API, and managing the response.

## 5.1 Flask Application Structure:

In the app.py file, the first section is the import section. Where all the mandatory libraries were imported. Then the environment variables were loaded from the .env file, and prompt engineering was done. It has also been ensured to create a directory for storing the images. After that the OpenAI client is initialized, and the image is encoded to base64 using “encode_image” function. Following is the code snippet.

****Code Begins****

```
import os
import base64
from flask import Flask, render_template, request
from dotenv import load_dotenv
import openai

# Load environment variables from .env file
load_dotenv()
prompt_eng = '''Your role is an image text extractor or object recognizer
where you will be provided with the
 images and you have to extract text from those images and recognize
objects if there is any object present
in the image  and if you don't find a field just make it null '''

# Ensure the directory for saving images exists
IMAGE_UPLOAD_PATH = os.path.join('static', 'images')
if not os.path.exists(IMAGE_UPLOAD_PATH):
    os.makedirs(IMAGE_UPLOAD_PATH)
```

```
# OpenAI Client initialization
def initialize_openai_client():
    return openai.OpenAI(api_key=openai_api_key)

# Function to encode image to base64
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode('utf-8')
```

*****Code Ends*****

### 5.2 ChatGPT API Call:

In the subsequent section, a router is created to render the uploaded form and handle image and prompt submission. Also, a chat completion request is initiated, with a try and except block which ensures error handling, which sends the image along with the prompt to the Chat GPT model for analysis. The Following is the code snippet attached.

****Code Begins****

```
 try:
            # Create chat completion request
            chat_completion = client.chat.completions.create(
                model="gpt-4o",
                max_tokens=1000,
                messages=[
                    {
                        "role": "user",
                        "content": [
                            {
                                "type": "text",
                                "text": prompt_eng   # Combining prompt
and question into one field
                            },
                            {
                                "type": "image_url",
```

```
                                "image_url": {
                                    "url":
f"{base64_image}"
                                }
                            },
                        ]
                    }
                ]
            )
            # Get response content
            response_text = chat_completion.choices[0].message.content
            return render_template("index.html",
response_text=response_text)

        except Exception as e:
            error_message = f"An error occurred: {e}"
            return render_template("index.html",
error_message=error_message)
```

***** Code Ends*****

After successfully executing the above code the HTML template is rendered.

### 5.3 Design Considerations:

During implementation error handling is considered carefully. For example if a user try to upload a non-image file, Flask is responsible to validate the file and if the API is not able to process it, an error message is shown to the user. The design is kept simple that is one page input and output because the main focus was on the core functionality. The Flask app was locally tested on localhost: 5000. There is no database involved in this system as the data is not for long term storage. No additional libraries were needed except Open AI and Flask, which kept this project lightweight.

# 6 RESULTS AND DISCUSSION

The completed system successfully demonstrated its expected outcomes: The image uploaded to the web interface was successfully analysed by the Chat GPT model and the results were displayed in the form of textual description. For example, when a car image was uploaded to the model it successfully identified the object and extracted the text e.g. car number plate, car logo and surroundings etc. Result is shown in the following images:


*FIGURE 7. Input Image*

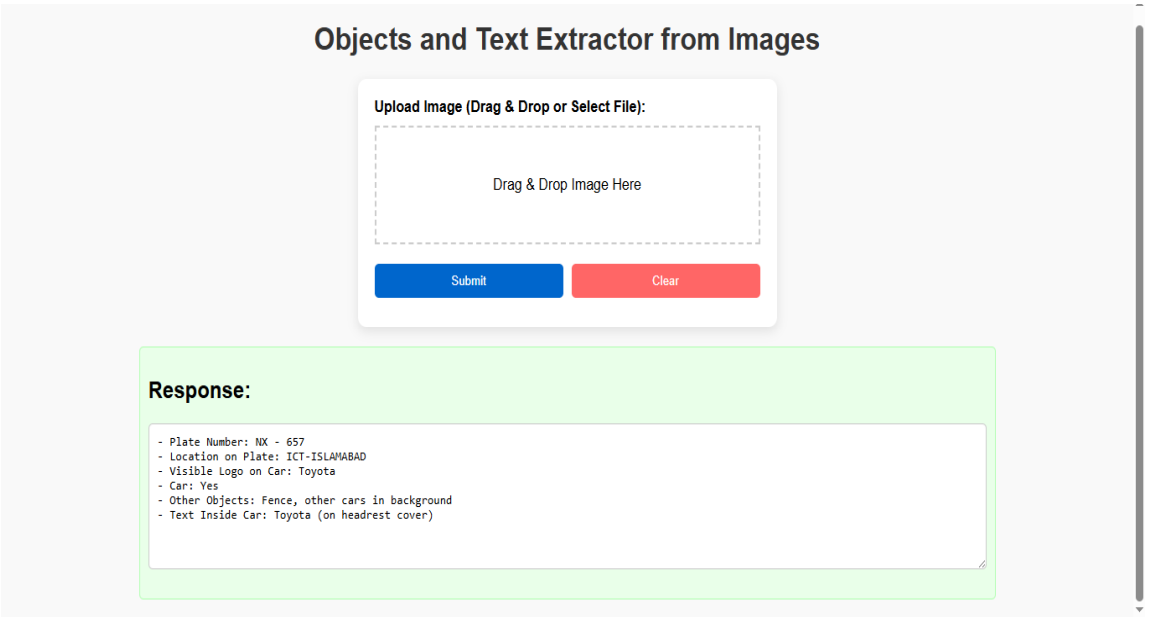



*FIGURE 8. Output*

- Plate Number: NX – 657
- Location on Plate: ICT Islamabad
- Visible Logo on Car: Toyota
- Car: Yes
- Other Objects: Fence, other cars in background

-Text Inside Car: Toyota ( on headrest cover)

Here is another example of the system successfully extracting text from an image:


*FIGURE 9. Input Image*

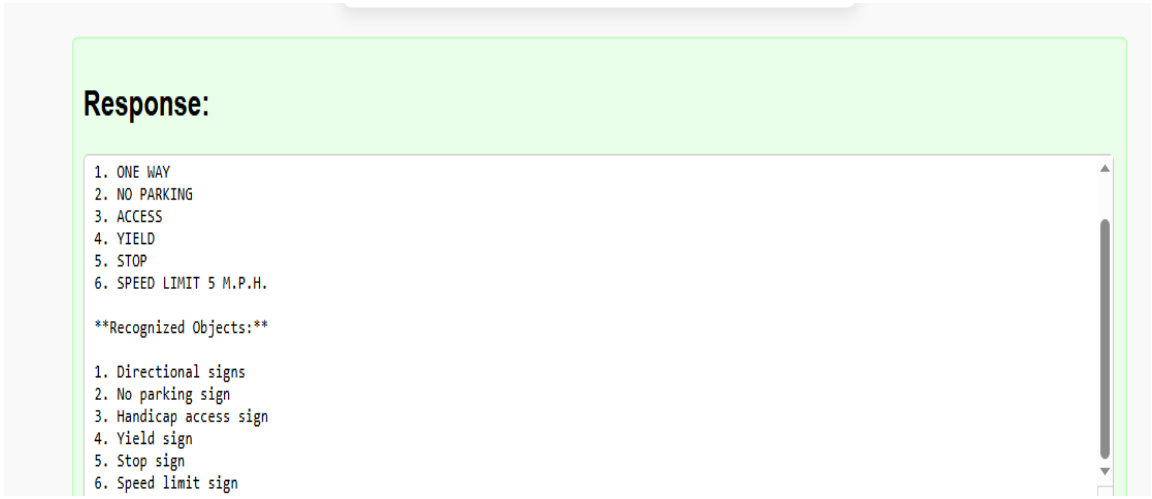



*FIGURE 10. Output*

In the above shown pictures, a relatively low resolution picture, containing traffic signs and their descriptions, was provided to the model, however, it was successfully analysed and all the text and signs were identified correctly.

In a test with images of printed invoices, the system was able to extract the following information:

**Extracted Text:**

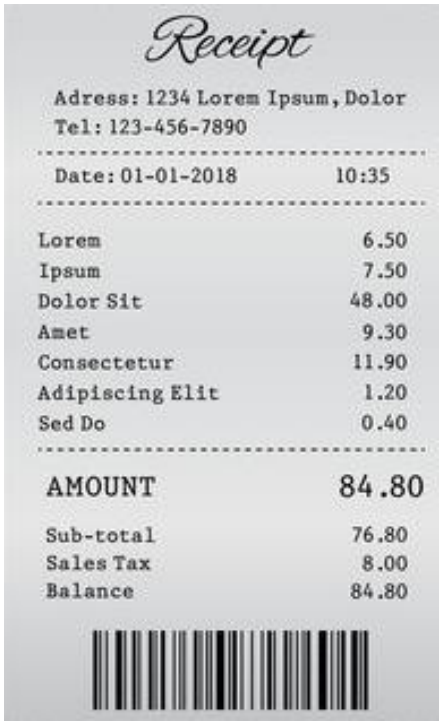
- **Header:** Receipt

- **Address:** 1234 Lorem Ipsum, Dolor
- **Tel:** 123-456-7890
- **Date:** 01-01-2018
- **Time:** 10:35

**Items:**
- Lorem: 6.50
- Ipsum: 7.50
- Dolor Sit: 48.00
- Amet: 9.30
- Consectetur: 11.90
- Adipiscing Elit: 1.20
- Sed Do: 0.40

**Total Amount:**
- **Amount:** 84.80
- **Sub-total:** 76.80
- **Sales Tax:** 8.00
- **Balance:** 84.80

**Objects:**
- Barcode




Following are the images for the above described test:

## 6.1 PERFORMANCE AND LIMITATION:

The response time was reasonable, i.e. 5~6 seconds per query, depending on the size of the image and internet speed. Flask API handled multiple tests in a session without any problem.

In GPT-4 there are several limitations about its usage for image recognition. One of them is Accuracy: Although it showed impressive results during its testing, however the proposed system may misread small or stylized text or text other than the English language family, because it is not a dedicated OCR engine.

Another concern is dependence on an external API: the system requires an internet connection and a valid API key. Finally, the model can “hallucinate” or make up details if unclear; although less common for straightforward OCR tasks, it is possible (for example, miss-labeling an object or adding fictional text).

## 6.2 FUTURE WORK

To increase accuracy, a hybrid approach could also be used. For instance, run first a standard OCR library (Tesseract) on an image and then feed the text to understand or model it from ChatGPT. Or instead of using the GPT to identify items, object-detection models (YOLO, OpenCV) could be used. Results may also be improved by fine-tuning a smaller vision model on our data. There are the UI improvements that may include bounding boxes or confidence scores. Future steps include caching results in repeated images and adding user authentication and up-load of multiple images. Last but not the least, there will be a need to monitor and log for a production deployment (to capture usage and debugging issues).

## 6.3 CONCLUSION

In conclusion, the project was able to achieve the goal of reading objects and text from images using ChatGPT API as shown in the end-to-end flow from web interface to AI processing. The approach functioned as proof of concept and both opportunities and challenges of using large language models for visual data extraction became visible. Such applications may become more robust as large multimodal models grow (e.g. GPT-4 Vision improvements or GPT-4o). This work provides a platform to build upon when pursuing integration of web technologies with high level AI services.

# 7 REFERENCES


ABBYY 2023. *ABBYY FineReader Engine* s.l.: ABBYY.

Ashish Shenoy Y. L. S. J. D. C. e. a. 2024. Lumos: Empowering Multimodal LLMs with Scene Text Recognition. *ACM.*

Biswas, A. 2024. The History of Convolutional Neural Networks for Image Classification (1989 - Today). *Medium.*

Britton, S. 2022. Optical character recognition (OCR) technology: A brief history.

Chandni Kaundilya & Diksha Chawla, Y. C. 2019. *Automated Text Extraction from Images using OCR System.* New Delhi, India, IEEE.

Fahey, J. 2024. *The Story of AlexNet: A Historical Milestone in Deep Learning,* s.l.: Medium.

Keechul Jung, K. I. K. A. K., 2004. *Text information extraction in images and video: a survey.* s.l., Science Direct.

Kurzweil Technologies, 2025. *Kurzweil Technologies.* URL: https://www.kurzweiltech.com/kcp.html Accessed:14.5.2025

Lemelson.MIT 2019. Raymond Kurzweil.

M.S. Akopyan, O. B. 2019. Text recognition on images from social media. *IEEE.*

Mehar Prateek Kalra, A. K. 2024. LLM Powered HTR: Integrating Handwritten Text Recognition System with Large Language Model. *IEEE XPlore.*

Microsoft Azure 2025. *Vision-enabled chat model concepts,* s.l.: s.n.

Mozilla Developer Network 2025. *HTML & CSS Documentation*. URL: https://developer.mozilla.org Accessed:14.5.2025

Open AI 2023. *GPT-4 Technical Report,* s.l.: s.n.

Pinecone, s.a. *AlexNet and ImageNet: The Birth of Deep Learning.* URL: https://www.pinecone.io/learn/series/image-search/imagenet/ Accessed:14.5.2025

Thomas, E. 2021. A Century Ago, the Optophone Allowed Blind People to Hear the Printed Word. *IEEE Spectrum.*

Wernicke, R. L. a. A. 2002. *Localizing and Segmenting Text in.* s.l., IEEE .