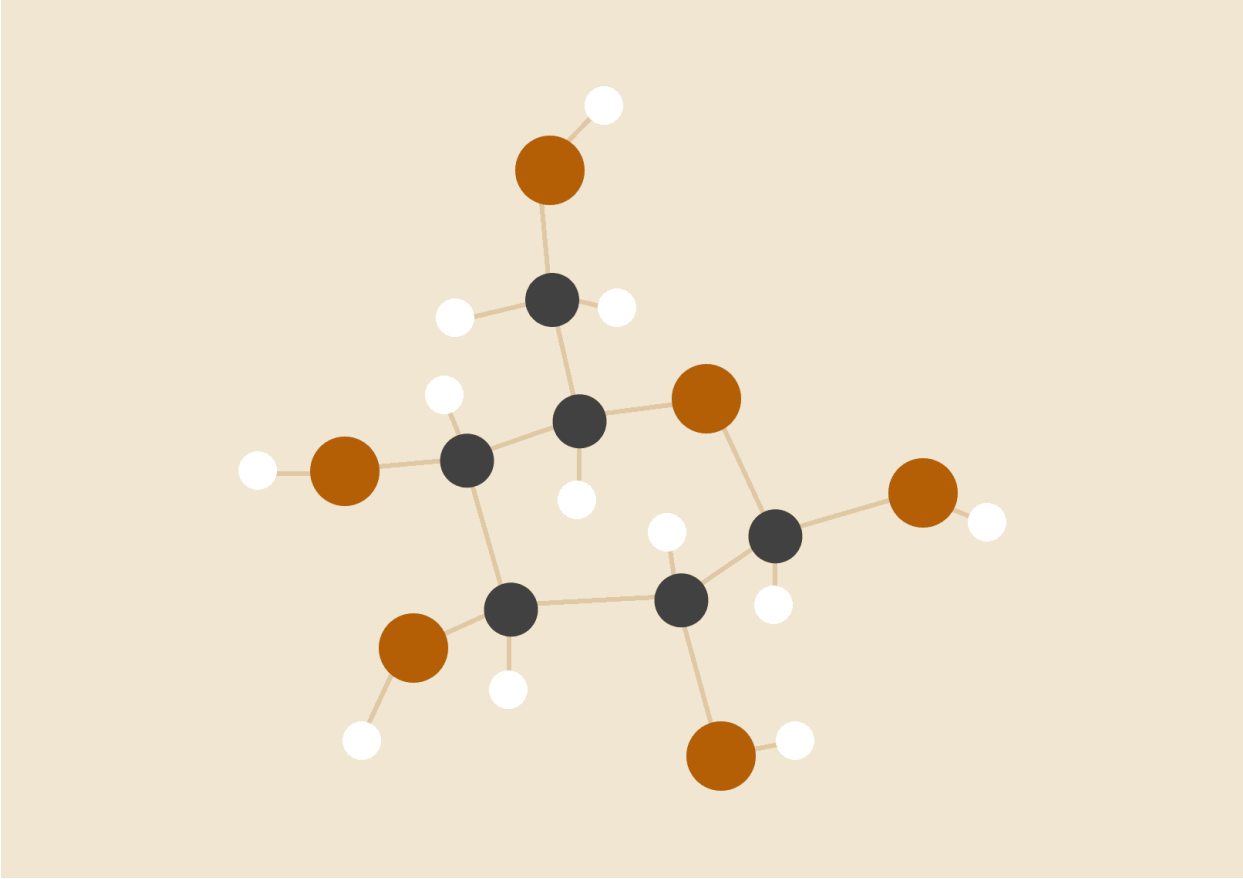
# Fake Invoice Detector using Computer Vision



**Sofia Raheel**

## PROJECT OVERVIEW

The **Fake Invoice Detector** is a mini web application that uses computer vision and OCR to verify the authenticity of uploaded invoices. By comparing the visual layout and extracted key fields (like invoice number, date, and amount) with known authentic templates, the system flags invoices as *Original* or *Fake* based on detected anomalies. This project combines OpenCV, OCR (Tesseract/EasyOCR), and basic template matching to automate invoice verification.

## CODE BASE

The project combines **computer vision**, **OCR (Optical Character Recognition)**, and a **Flask web interface** to detect whether an uploaded invoice is authentic or fake.

## APPLICATION SETUP

The project begins by importing necessary libraries:

- `cv2` and `numpy` for image processing using OpenCV.
- `pytesseract` for OCR to extract text from images.
- `Flask` and related modules for building the web server and handling file uploads.

The `Flask` app is then initialized. It includes configuration for:

- Allowed file extensions (`jpg`, `jpeg`, `png`, `pdf`).
- A temporary directory for uploads.
- A maximum file size limit of 16 MB.

Tesseract OCR is initialized with its path, which must be set appropriately depending on the operating system.

## FILE VALIDATING PROCESS

Before analyzing an uploaded invoice, the code first validates the file format and saves it securely. The helper function `allowed_file(filename)` ensures only files with supported extensions are processed.

Once the file is saved, it is passed to the core image processing pipeline.

The image is then **preprocessed** for OCR. This involves:

- Converting the image to grayscale for uniformity.
- Applying binary thresholding to enhance contrast between text and background.
- Using Gaussian blur to reduce noise and improve OCR accuracy.

## DATA3. Feature Extraction

The main logic of invoice verification is in the extract_features(img) function. It performs two key tasks:

**1.Text Field Extraction**

Using Tesseract OCR, the text content from the invoice image is extracted. The code then attempts to identify key fields by searching for known keywords such as:

- “Invoice Number” or “Invoice No”
- “Date” or “Invoice Date”
- “Vendor” or “Supplier”
- “Total” or “Amount Due”

The extract_field() function scans the OCR output line-by-line and isolates the value following these keywords.

**2. Layout Analysis**

To assess the visual structure of the invoice, the function extract_layout_features(img) performs:

- Edge detection using the Canny algorithm.
- Line detection using the Hough Transform.
- Counting of horizontal and vertical lines.
- Calculation of pixel variance in the header region (top 100 pixels), which gives an estimate of header/logo complexity.

This combination of text and layout features provides a comprehensive representation of the invoice.

### Invoice Analysis Logic

The core function analyze_invoice(file_path) brings together the extracted data to classify the invoice.

Three basic checks are performed to flag a fake:

- Missing invoice number
- Missing total amount
- Insufficient layout structure (e.g., very few horizontal lines)

If any of these are true, the invoice is flagged as fake. Otherwise, it's assumed to be original.

The result includes:

- Boolean status (is_fake)
- Extracted text
- Extracted features (field values and layout metrics)

### Web Interface and Routes

The application has two main routes:

**a. / (index)**

Renders the frontend form where users can upload invoice files.

**b. /upload (POST)**

Handles the file upload, runs the analysis pipeline, and returns the result as a JSON response. The response contains:

- Invoice status (Original or Fake)
- Extracted field values
- Raw OCR text

Error handling is in place to catch invalid uploads or processing failures.

### Template Matching (Extended Version)

The extended version (from the command-line script) introduces **template-based**

**matching**:

- Known original invoice templates are stored and analyzed.
- The uploaded test invoice is compared with templates based on:
    - Similarity in line counts
    - Header complexity
    - Matching presence of fields

A score is computed based on these comparisons, and a threshold determines if the invoice is original or fake.

This version provides visual output using matplotlib, showing:

- The original and processed images
- The extracted text
- Final decision with scoring

## CONCLUSION

This project demonstrates how basic computer vision and OCR techniques can be used for document authenticity detection. The modular structure of the code allows for further enhancement, such as using machine learning models or expanding to multi-page PDFs. The web interface offers a simple way for end users to interact with the model and test different invoices for validation.