Summer Internship Report

Hiding Sensitive Information in Images and Documents

Boda Surya Venkata Jyothi Sowmya

A report presented for the summer internship at
AlgoLabs
Under the guidance of mentor
Prasun Agarwal
Year: 2025

Contents

Acknowledgement

I would like to sincerely thank AlgoLabs for giving me the opportunity to work on this project. It has been an highly valuable learning experience, allowing me to explore and implement advanced techniques in computer vision and large language models.

My deepest gratitude goes to my mentor, Mr. Prasun Agarwal. His guidance and constructive feedback significantly enhanced the quality of my work throughout the project.

I am also very thankful to Mr. Sudhir Kumar for introducing the concept behind this project. His vision gave me a strong foundation to build upon and inspired me to approach the problem with creativity and determination.

Although this project was independently executed, the guidance, trust, and resources provided to me played a huge role in making it a reality. This project has been both challenging and rewarding, and I am truly grateful to everyone who made it possible.

Abstract

This report presents the design and deployment of a robust, end-to-end system for the detection and redaction of Personally Identifiable Information (PII) in document images. Leveraging advancements in computer vision, Optical Character Recognition (OCR), and large language models (LLMs), the proposed pipeline automates the identification and masking of sensitive fields in scanned identity documents such as Aadhaar, PAN, and passports. The system integrates YOLOv8 for document localization, Aspose OCR and OpenCV for image alignment, PaddleOCR for text recognition, and Google Gemini for semantic analysis. Identified PII is then masked using the Pillow library. The complete solution is implemented as a user-friendly Streamlit application, containerized via Docker for portability. Evaluation on a diverse dataset demonstrates over 95% precision and recall. Key challenges such as OCR limitations and skew correction are discussed, along with avenues for future enhancements, including multilingual support and expanded document coverage.

1 Introduction

1.1 Background

In the digital age, documents such as ID cards, passports, and utility bills are frequently shared online, most of these contain Personally Identifiable Information (PII) such as our name, address, phone number, or ID numbers. If this data ends up in the wrong hands, it can easily be misused, leading to privacy breaches or even identity theft. With so much of our information being exchanged digitally, protecting it has become more important than ever.

1.2 Motivation

Currently, removing sensitive information from documents remains a complex and time-consuming task. Manual processing is time-intensive and prone to errors, often resulting in overlooked sensitive information. This makes the whole process slow and unreliable, especially if you have a large number of documents to handle. These challenges highlight the need for an efficient and automated solution — one that can detect and hide personal information quickly and accurately without depending on manual review.

1.3 Problem Statement

This project addresses the critical issue of protecting personal information from exposure in document images. The challenge was to create a system that can automatically identify and hide this information. This task is complex due to variations in format, layout, orientation, and image quality. A generic approach is insufficient to address the variability in document formats, layouts, and image quality.

1.4 Objectives

During my internship at AlgoLabs, my main aim was to develop and test an automated privacy protection system for document images. My specific goals were:

- To create a processing pipeline that can detect, extract, and mask sensitive details.
- To use computer vision techniques like YOLOv8 for object detection and OpenCV for image processing to handle document images.
- To integrate a large language model (LLM) such as Google Gemini to improve the accuracy of detecting PII fields.
- To build a functional and reliable application that can help protect user privacy.
- To make sure the system works well on different types of documents, in different orientations, and with varying image qualities.

2 Literature Survey

2.1 Overview of PII Detection Approaches

Different approaches have been explored for detecting and masking Personally Identifiable Information (PII) from documents. Some methods are rule-based, relying on predefined patterns such as regular expressions to identify sensitive details like phone numbers, email addresses, or identification numbers. While rule-based methods can work well for structured text, they struggle with unstructured or noisy data.

More recent approaches employ machine learning and deep learning models to automatically identify PII in textual content. For example, Sinha (2023) demonstrates the use of the Watson NLP library to extract PII entities by leveraging pre-trained Named Entity Recognition (NER) models and NLP pipelines designed for compliance monitoring. This method performs effectively on structured or unstructured digital text but is not directly applicable to scanned document images due to the absence of spatial and visual context.

For document images, computer vision-based techniques have increasingly gained attention due to their ability to work with unstructured visual data. These approaches typically detect regions of interest (e.g., ID numbers, names, photos) using object detection models, followed by targeted text extraction. For instance, Ikomia AI (2023) presents a deep learning-based pipeline for extracting information from ID cards, where the system first performs object detection and cropping, and then applies Optical Character Recognition (OCR) to extract textual fields. While the exact implementation from this work was not adopted, it provided conceptual inspiration for structuring the detection-first pipeline used in this project.

My approach combines insights from these resources with other references to build a custom pipeline tailored for this project's requirements, integrating both computer vision and language-based methods to detect and mask sensitive information effectively.

2.2 OCR and Text Extraction Methods

Optical Character Recognition (OCR) plays a key role in extracting text from scanned documents and images. It allows us to convert printed or handwritten text into machine-readable form, which is essential for tasks like data extraction, anonymization, and analysis.

Traditional OCR tools like **PyTesseract** are suitable for clean, high-quality images. However, they often fail to deliver good results when the images are noisy, blurry, tilted, or contain complex layouts. **EasyOCR** provides better support for different languages and text structures, but its accuracy also drops significantly on low-quality or unstructured documents.

Cloud-based services such as **Azure OCR** and **Google Vision OCR** typically offer higher accuracy and better performance in handling difficult cases. However, these services require an internet connection, may introduce latency, and usually involve recurring costs. For projects that need to process a large number of documents or must work offline, these limitations can be challenging.

To address these issues, this project uses **PaddleOCR's PP-OCRv3**, an advanced and open-source OCR system developed by Baidu. PP-OCRv3 is specifically designed for real-world document processing and combines several deep learning models into one efficient pipeline:

- A text detection model that identifies where the text appears in the image.
- An angle classification model that detects and corrects rotated or skewed text.
- A text recognition model that extracts the actual characters from the image.

In this project, the following pre-trained models are automatically downloaded and used:

- `en_PP-OCRv3_det_infer` – detects English text regions.
- `en_PP-OCRv3_rec_infer` – recognizes English characters.
- `ch_ppocr_mobile_v2.0_cls_infer` – classifies and corrects text angles.

These models work together to extract text accurately, even when the image quality is poor, the text is rotated, or multiple languages are present. PP-OCRv3 also returns the location (bounding boxes) of the detected text, which is especially useful for identifying and masking sensitive fields such as names, phone numbers, or addresses.

To improve accuracy further, preprocessing techniques like cropping, noise reduction, and rotation correction are applied before running OCR. These steps help ensure that the input to the model is clean and properly aligned, which leads to better detection and recognition.

By combining these features, PP-OCRv3 provides a strong and reliable foundation for downstream tasks, including Personally Identifiable Information (PII) detection and masking, making it well-suited for this project's goals.

2.3 Integration of Computer Vision and LLMs for Data Masking

Modern PII masking in document images increasingly relies on the combination of computer vision and large language models (LLMs) to achieve high accuracy and adaptability.

In this project, computer vision formed the backbone of the detection stage. Using YOLOv8, the system identified key document elements — such as ID cards — and generated bounding boxes to precisely locate regions of interest. To handle tilted or misaligned scans, Aspose OCR's calculate_skew function was used to estimate the skew angle, and OpenCV was applied for deskewing, ensuring that text regions were correctly oriented for recognition.

For text extraction, PaddleOCR was employed due to its ability to accurately detect and recognize text in varied orientations, fonts, and layouts. Importantly, PaddleOCR also provided coordinates for each recognized text block, enabling a direct link between the extracted text and its exact position in the image.

Once text was extracted, the process moved into the LLM stage. A large language model, such as Gemini, was used to semantically analyze the extracted text and determine which fields contained Personally Identifiable Information (PII). Unlike traditional regex-based methods, the LLM could interpret context, handle variations in formatting, and detect sensitive fields even when labels were ambiguous or missing.

Finally, identified PII regions were masked directly on the image using the Pillow library, ensuring privacy while retaining the document's structure. This synergy between computer vision for spatial detection and LLMs for contextual understanding provided a robust and adaptable solution for automated PII protection.

3 Proposed Methodology

3.1 Project Pipeline

The proposed system for automated PII masking follows a structured pipeline that combines computer vision and natural language processing to detect, identify, and hide sensitive information in document images. The overall workflow consists of the following stages:

- **Image Acquisition**– Collecting document images in various formats and qualities.
- **Document Detection and Cropping** – Using object detection to locate the document within the image and crop it for further analysis.
- **Image Alignment** – Correcting any skew or tilt to ensure proper orientation.
- **Text Extraction** – Applying OCR to extract text and its location from the aligned document.
- **PII Identification** – Using large language models to detect sensitive information fields.
- **Masking** – Blurring the detected PII fields while keeping the rest of the document intact.
- **Application Development** – Integrating the pipeline into an interactive Streamlit application for user-friendly operation.
- **Deployment** – Containerizing the application with Docker and publishing it to DockerHub for portability and ease of distribution.

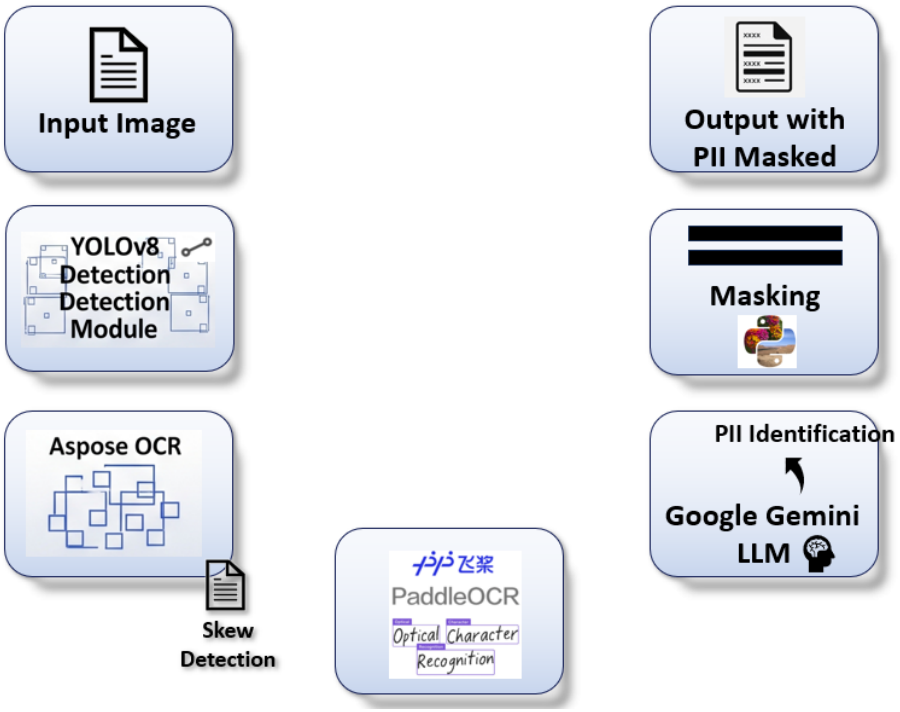


Figure 1: Pipeline of the project

3.2 Dataset Description

A robust and diverse dataset was essential for training the object detection model to accurately identify documents under varied real-world conditions. The dataset was curated from multiple sources, including Google image searches, Roboflow public projects, GitHub repositories, and Kaggle datasets. This ensured a broad and representative mix of document types and image conditions.

The dataset was split into 780 images for training and 72 images for testing. This separation allowed the model to learn from a wide range of examples while being evaluated on a distinct, unseen set to reliably measure performance.
The dataset covered a variety of official documents, including:

- Aadhaar Cards
- Passports
- PAN Cards
- Driving Licenses
- Voter Cards
- Delhi Water Bills

To ensure robustness, the images were chosen to reflect varied backgrounds, different lighting conditions, and mixed orientations. This diversity was critical for enabling the model to handle imperfect scans and photographs effectively.

Each image was manually annotated using Label Studio to create precise bounding boxes for the documents. These annotations were vital for training the YOLOv8 object detection model to achieve accurate and consistent detection results.

3.3 ID Detection and Cropping

The first stage of the project involved accurately detecting identity cards within input images. For this task, the YOLOv8 object detection model was chosen due to its impressive balance between speed and accuracy. Its real-time inference capabilities, lightweight architecture, and strong performance on custom object detection tasks made it particularly well-suited for identifying identity cards across a range of layouts, sizes, and orientations. Furthermore, YOLOv8's anchor-free detection design and improved feature aggregation contributed to better generalization, especially when working with a relatively small custom dataset.

3.3.1 Dataset Preparation:

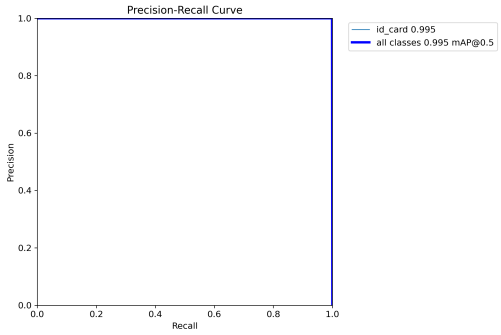
To train the model effectively, a total of 200 annotated images were curated, covering a variety of identity card types and layouts. The dataset was divided into 160 images for training and 40 for validation, ensuring a balanced representation across both sets.

3.3.2 Model Training:

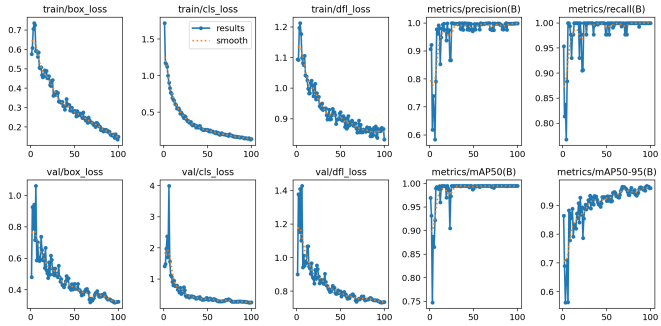
A pre-trained YOLOv8n model (`yolov8n.pt`) was fine-tuned on the prepared dataset over the course of 100 epochs. The model demonstrated strong performance across key metrics:

- Precision: ∼99.8
- Recall: ∼99.8
- mAP@0.5: ∼99.5
- mAP@0.5:0.95: ∼97.3

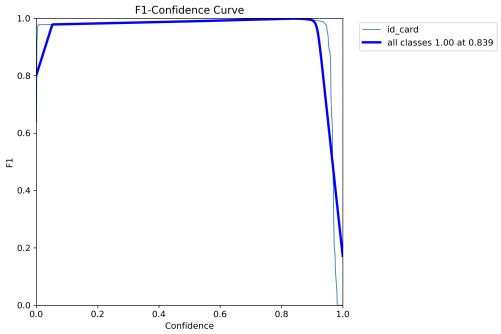
These results confirmed that YOLOv8 was a reliable and efficient choice for document detection in this project, forming a solid foundation for subsequent steps like alignment and text extraction.



(a) PR Curve



(b) Training and Validation Results



(c) F1 Curve

Figure 2: Training performance curves: (a) Precision–Recall (PR) Curve, (b) Training results showing loss and accuracy trends, and (c) F1 Curve.

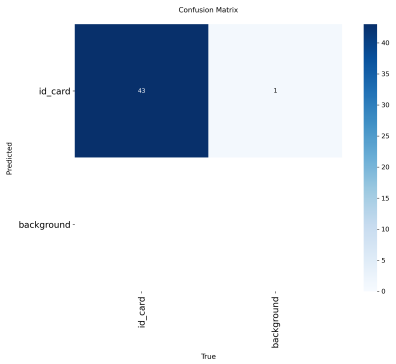
3.3.3 Training Curves:

- **Results Curve** – Box loss, classification loss, and DFL loss steadily decreased throughout training, while precision, recall, and mAP quickly rose and stabilized near 1.0, indicating strong learning without overfitting.

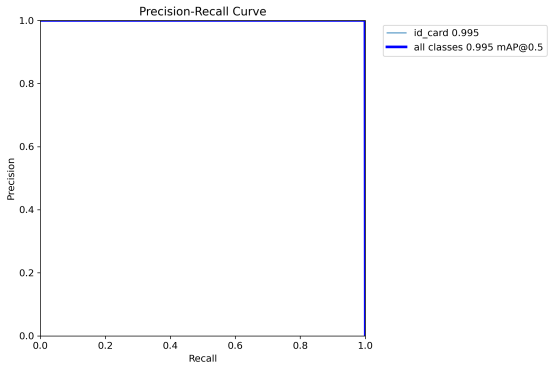
- **Precision-Recall (PR) Curve** – The curve nearly touches the top-right corner, reflecting 0.998 precision and recall, meaning the model detects almost all ID cards with exceptional accuracy.
- **F1-Confidence Curve** – The maximum F1 score of 1.00 occurred around a confidence threshold of 0.84, with stable performance across a wide confidence range.

3.3.4 Validation Curves:

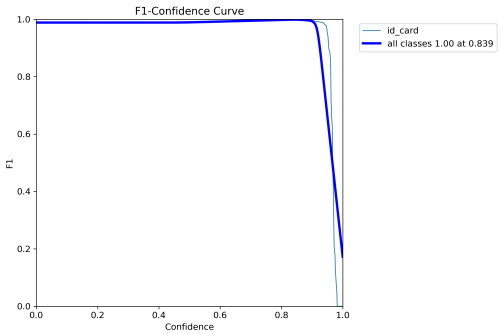
- **Confusion Matrix** – The model correctly detected 43 ID cards with only 1 false positive and no false negatives, showing near-perfect classification accuracy.
- **Precision-Recall (PR) Curve** – Stays close to the top-right corner with 0.995 mAP@0.5, reflecting consistently high precision and recall across thresholds.
- **F1-Confidence Curve** – Peaks at 0.839 confidence with an F1 score near 1.0, maintaining excellent balance between precision and recall over a wide range.



(a) Confusion Matrix



(b) PR Curve



(c) F1 Curve

Figure 3: Validation performance curves: (a) Confusion Matrix (b) PR Curve (c) F1 Curve.

3.3.5 Cropping Strategy

After training, the best model weights (best.pt) were used to run inference on all 850 collected images to detect identity cards. The objective was not only to detect the cards but also to crop them precisely for further processing.

The cropping strategy involved:

- **Selecting the most confident detections :** Detections with a confidence score of at least 0.7 were prioritized.
- **Handling multiple detections :** When multiple bounding boxes were detected in the same image, the highest-confidence bounding box was selected.
- **Allowing multiple valid detections :** If two high-confidence detections were found (e.g., front and back of a card in the same image), both were retained.
- **Removing unnecessary background :** Cropping was performed tightly around the detected bounding box to isolate the document region and remove distractions.

This approach ensured that the extracted images were clean, background-free, and ready for the Image Alignment stage, which handled skew correction and orientation adjustments before text extraction.

3.4 Image Alignment

Once the identity cards were cropped from the original images, the next step was to ensure they were properly aligned for optimal Optical Character Recognition (OCR) performance. Even small amounts of skew in a document can reduce OCR accuracy, as text lines become tilted and harder to interpret.
Initial Attempts Several deskewing techniques were tested, including:

- OpenCV-based skew detection.
- Other algorithmic approaches based on text-line projections.

However, these methods did not work effectively and often failed to detect the correct skew angle.

3.4.1 Aspose OCR for Skew Detection

To address this, the Aspose OCR library was integrated into the pipeline. Aspose OCR provides a calculate_skew function, which can estimate the skew angle of a document image with high reliability. This function analyzes text alignment and returns a rotation angle that can be used to correct the skew.

3.4.2 Deskewing with OpenCV

Once the skew angle was obtained from Aspose OCR, OpenCV was used to perform the actual image rotation:

- Read the cropped image and determine its dimensions.
- Calculate the new dimensions after rotation to preserve all content without clipping.
- Rotate the image around its center using the skew angle.
- Save the corrected, deskewed image for use in the OCR step.

This method consistently produced accurate results for the majority of images, producing straight, well-aligned documents ready for text extraction.

3.4.3 Special Case Handling

While the Aspose OCR worked well for most cases, a few images (10–15) were already perfectly straight but were incorrectly assigned a skew angle. As a result, they were rotated unnecessarily, making them slightly misaligned. This was a rare occurrence and had a negligible impact on overall system performance, but it suggests that an additional validation step could further improve accuracy.

3.4.4 Outcome

By integrating Aspose OCR's skew detection with OpenCV's precise rotation, the alignment step ensured that nearly all documents were correctly oriented. This alignment significantly improved OCR accuracy in the subsequent Text Extraction stage.

3.5 Text Extraction

Once the documents were properly cropped and aligned, the next step involved extracting text for further analysis, particularly for identifying Personally Identifiable Information (PII). For this task, PaddleOCR was selected due to its strong accuracy when handling multilingual content and complex document layouts.

3.5.1 OCR Implementation

PaddleOCR was applied to each aligned image to detect and recognize the text. The output included:

- The recognized text content.
- Bounding box coordinates for each detected text region.

To maintain a structured format, the extracted text segments were initially separated using semicolons. This made it easier to distinguish between individual entries and preserve the spatial and logical arrangement of the text.

3.5.2 Addressing OCR Challenges

In some cases, particularly with lower-quality images, PaddleOCR faced difficulty recognizing certain characters like semicolons or colons. This often led to the merging of field labels and their associated values into a single block with one bounding box, making it difficult to differentiate between the two.

To mitigate this issue, a keyword-based splitting method was introduced. This involved scanning for common field labels such as "Name", "Date of Birth", "Father's Name", and "Address" and using them as anchors to separate labels from their values. While this approach helped in many instances, it wasn't always reliable—especially when field names appeared in unexpected formats (e.g., abbreviated, differently cased, or missing delimiters). Some edge cases remained unresolved.

For images with severe noise, poor resolution, or heavy blurring, the OCR engine sometimes failed to detect text correctly or produced inaccurate results. These limitations were expected, as such image conditions significantly reduce OCR performance.

3.5.3 Outcome

Despite the challenges posed by a small number of low-quality images, PaddleOCR successfully extracted the majority of text fields along with their positions. This structured output provided a reliable basis for the next phase of the pipeline: PII field detection and masking.

3.6 PII Identification and Masking

Once the text and its coordinates were extracted from the aligned images, the next step was to determine which portions of this text contained Personally Identifiable Information (PII) and mask them to protect user privacy.

This was achieved using the **Gemini 1.5 Flash Large Language Model (LLM)**. Gemini was chosen due to its strong multimodal capabilities—being able to understand both document images and OCR-extracted text in context—making it well-suited for semantic-level PII detection. It was capable of identifying a wide range of sensitive fields such as names, addresses, ID numbers, and dates of birth, even when explicit labels were missing, ambiguous, or inconsistent across different documents.

Another reason for selecting Gemini was its accessibility. The model is available via a free API, which made it ideal for solo or small-scale projects. Although there are daily usage limits, they were sufficient for the scope of this work and did not pose any significant constraints. Additionally, Gemini provided a balance between capability and ease of integration.

During the experimentation phase, other models such as **DeepSeek R1** were also tested. While DeepSeek showed some potential, it failed to detect some key PII fields. In comparison, Gemini performed more consistently and was more accurate in extracting PII, even in challenging scenarios. This reliability made it a preferred choice.

3.6.1 Process

Gemini was provided with:

- The OCR-extracted text and its bounding box coordinates.
- The document image itself.

Gemini used the image mainly for document type detection and for enhancing text accuracy in cases where the OCR struggled, such as low-quality scans or blurred photographs. For example, in some images, PaddleOCR failed to detect characters like colons (:) or read certain fields correctly due to poor image quality. Gemini helped interpret these cases by using its visual and contextual understanding.

However, a key limitation of Gemini is that it does not return the spatial coordinates of the detected PII fields. Therefore, positional information had to be retrieved from the OCR output. This OCR-Gemini hybrid setup allowed semantic understanding from Gemini to be paired with bounding box data from OCR for precise masking.

3.6.2 Output and Masking

The output of this step was a structured JSON file containing:

- The PII field name (e.g., Name, Date of Birth).
- The actual PII value.
- The coordinates of that field in the document.

For masking, the Pillow library was used to accurately cover only the detected PII regions. These regions were filled with solid rectangles, making the sensitive information completely unreadable while keeping the rest of the document unchanged. This approach ensured that the layout and appearance of the document were preserved, so it remained clear and usable after masking.

This method provided a strong level of privacy, as the hidden information could not be recovered from the image. The combination of Gemini for understanding the meaning of the text and OCR for locating it made the process both accurate and adaptable. It worked well across different document types, layouts, and image qualities, making it suitable for real-world use.

4 Results and Analysis

4.1 Implementation of Streamlit App

To make the solution accessible and user-friendly, I developed a Streamlit-based web application that integrates the complete PII masking pipeline. The app allows users to upload images or PDF files, processes them through the detection and masking pipeline, and returns the masked output for download.

The application coordinates multiple components — YOLOv8 for ID card detection, Aspose OCR and OpenCV for alignment, PaddleOCR for text extraction, Gemini for PII identification, and Pillow for masking. All processing runs seamlessly in the backend while the user interacts with a simple, intuitive interface.

One notable enhancement I implemented during the app development was an intelligent cropping decision. In certain cases, ID cards may appear embedded inside larger documents that also contain sensitive text outside the card's bounding box. To avoid missing such information, the app first checks for any detected text outside the YOLO bounding box. If found, the original image is processed without cropping; otherwise, the ID card is cropped for faster and cleaner processing.

This integration ensured that the Streamlit app not only performed accurate PII masking but also handled edge cases intelligently, making it robust for real-world use.

4.2 Docker Containerization

To make the application easily deployable and platform-independent, I containerized it using Docker. This ensured that all dependencies, libraries, and configurations were packaged together, allowing the application to run seamlessly on any machine without manual setup.

The final application image was pushed to Docker Hub under the repository:

`bodasowmya235/pii_masking_app_v2`

Users can run the application locally in just two steps:

1. **Pull the Docker Image**

   ```
   docker pull bodasowmya235/pii_masking_app_v2
   ```

   This downloads the latest version of the image from Docker Hub.

2. **Run the Container with GEMINI API Key**

   ```
   docker run -p 8502:8502 -e GEMINI_API_KEY=<KEY> bodasowmya235/pii_masking_app_v2
   ```

The `GEMINI_API_KEY` environment variable is required for the PII detection feature to function.

Once the container is running, the application can be accessed in a browser at:

`http://localhost:8502`

This approach makes it simple to share and run the application without worrying about environment configuration or dependency issues.

4.3 Overall PII Detection Results

The system was evaluated on the collected test dataset, and the results demonstrate strong detection accuracy:

- Precision: 94.98%
- Recall: 95.07%
- F1-Score: 95.03%
- Accuracy: 90.52%

Figure 4: Overall Results of the Model

In total, the system correctly identified 984 true positives, with 52 false positives and 51 false negatives. The **Coordinate Validation Rate** (i.e., cases where both the field value and its bounding box coordinates were correct) was 27.05%. While field detection performance was strong, precise coordinate matching proved more challenging.

This lower coordinate validation rate was primarily due to three key factors:

- **Repeated Fields:** In documents like water bills, certain fields such as `Name` appear more than once (e.g., at the top and bottom). Gemini occasionally detected only one of the instances, leading to partial matches.
- **Overlapping Field Types:** In identity documents containing both `Name` and `Father's Name`, the LLM sometimes misclassified `Father's Name` as `Name`, resulting in mismatches during coordinate validation.
- **Multi-box Outputs:** Some fields had multiple bounding boxes (e.g., split names), but only one was used during matching, reducing the total coordinate match rate even when the value was correctly detected.

Despite these limitations, the combination of YOLOv8, PaddleOCR, and Gemini LLM offered a robust solution for end-to-end PII masking. The system generalizes well across diverse document types and layouts, and its accuracy can be further improved by enhancing OCR fidelity and post-processing logic.

4.4 Evaluation Metrics

To measure the effectiveness of the system, the following metrics were used:

- **Precision** – Measures how many of the detected PII fields were actually correct.
- **Recall** – Measures how many of the actual PII fields present in the document were detected.
- **F1-Score** – The harmonic mean of precision and recall, providing a balanced measure of accuracy.
- **Accuracy** – Overall correctness of predictions.
- **Coordinate Validation Rate** – Proportion of correctly detected fields where the bounding box coordinates also matched.

These metrics provide a comprehensive view of the system's performance, showing not only its ability to detect PII but also how precisely it localizes the sensitive information within the image.

5 Challenges and Future Scope

5.1 Challenges Encountered

During the development of this project, several challenges were encountered that required thorough analysis and problem-solving.

- **Embedded ID Cards in Larger Documents :** In some cases, the ID card was part of a bigger document that also had sensitive text outside the card area. Simply cropping the card would miss that extra information. I solved this by checking the entire image with OCR first, and if text was found outside the detected card, I processed the full image instead of just the cropped part.
- **Image Alignment Issues:** Correcting tilted images presented significant challenges. While the skew detection worked well most of the time, a few images that were already straight ended up being slightly rotated because the algorithm thought they were tilted.
- **Poor-Quality Images for OCR:** Low-resolution or blurry images caused OCR to miss some text or read it incorrectly. This reduced the accuracy of PII detection for those cases.
- **Text Separation and Coordinates:** Sometimes, the OCR output did not clearly separate labels and their values, and the detected text boundaries were not always precise. This made it harder to accurately map PII fields to their exact locations in the document.
- **Multiple Detections in Complex Backgrounds:** In images with busy backgrounds, the detection model sometimes found more than one region that looked like an ID card. I had to use confidence scores to pick the most reliable detection.
- **Gemini API Request Limits:** The Gemini LLM used for PII detection and text enhancement had a daily request limit of 50 requests per day.

5.2 Future Enhancements

There are several ways this project can be improved and expanded in the future:

- **Multi-Language Support :** Extend PII detection to support non-English languages like Hindi, Telugu, and other regional languages for broader applicability.
- **Better OCR Model :** Integrate a more advanced OCR tool to improve text recognition accuracy and coordinate extraction, especially on low-quality or noisy images.
- **Support for More Document Types :** Expand beyond ID cards and bills to include bank statements, credit/debit cards, hospital records, and other sensitive documents for a more complete PII-masking solution.
- **Enhanced PII Detection:** Integrate advanced LLMs or deep learning-based techniques to improve semantic identification and precise coordinate mapping of PII fields, thereby increasing the overall coordinate validation rate.

6 Conclusion

This project demonstrated a robust, end-to-end solution for automated detection and masking of Personally Identifiable Information (PII) in document images. By combining computer vision, OCR, and large language models (LLMs), the system achieved high accuracy in identifying and redacting sensitive content across a wide variety of real-world document types.

The approach was thoroughly validated through quantitative metrics, achieving over 95% precision and recall. The deployment of the solution as a containerized Streamlit application further highlights its practicality and scalability for real-world use cases.

While challenges such as skew detection errors, OCR inaccuracies on low-quality images, and API limitations were encountered, the project laid a solid foundation for future improvements. With planned enhancements like multi-language support, better OCR models, and support for additional document types, the system has the potential to become a highly reliable and versatile PII-masking solution for real-world use cases.

7 References

1. Lefevre, P. (2023). *ID documents detection with YOLOv8 (plus rotation!)* Medium. Retrieved from https://medium.com/@paul_lefevre/id-documents-detection-with-yolov8-plus-rotation-e991192e74d2

2. Ikomia AI. (2023). *Text extraction for ID card using deep learning.* Retrieved from https://www.ikomia.ai/blog/text-extraction-for-id-card-using-deep-learning

3. Aspose. (2023). *Skew correction in image processing using Aspose OCR for Python.* Retrieved from https://blog.aspose.com/ocr/skew-correction-in-image-processing-using-csharp/

4. Abrha, J. (2023). *Text extraction and parsing from contemporary maps using PaddleOCR.* Medium. Retrieved from https://medium.com/@joeabrha/text-extraction-and-parsing-from-contemporary-maps-by-leveraging-ocr-engine-paddle-ocr-using-31a79ae48837

5. Li, C., et al. (2022). *PP-OCRv3: Improving ultra lightweight OCR systems.* arXiv preprint. Retrieved from https://arxiv.org/abs/2212.09420

6. Sinha, V. (2023). *Personal identifiable information (PII) extraction using Watson NLP library.* Medium. Retrieved from https://medium.com/towards-generative-ai/personal-identifiable-information-pii-extraction-using-watson-nlp-library-c8e506a3dbe8

A Appendix

A.1 Appendix A – Tools and Libraries Used

Component	Tool / Library
Object Detection	YOLOv8n (Ultralytics)
OCR	PaddleOCR
Skew Detection	Aspose OCR
Image Alignment & Preprocessing	OpenCV
PII Field Identification	Gemini 1.5 Flash (LLM)
Masking	Pillow
Web Application	Streamlit
Deployment	Docker

A.2 Appendix B – YOLOv8 Training Hyperparameters

Parameter	Value
Epochs	100
Batch Size	4
Workers	2
Input Image Size	416 × 416
Flip Left-Right	0
Mosaic	0.0
HSV (Hue / Saturation / Value) Augmentations	h=0.0, s=0.0, v=0.0