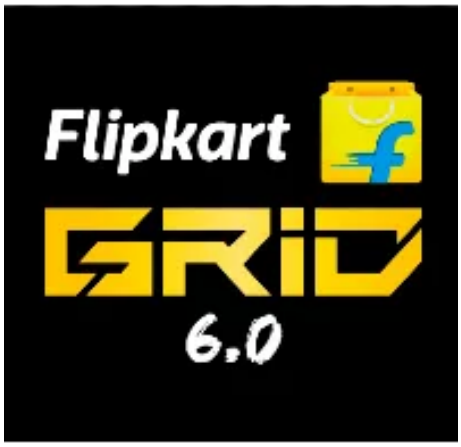

OCR System for Product Packaging Recognition

Team: ajitesh.kumar

December 11, 2024

1 Introduction

In this project, we implemented an Optical Character Recognition (OCR) system using `PaddleOCR` to extract text from images and video frames. The primary objective is to detect relevant information such as manufacturing dates, expiry dates, and MRP details from product packaging images. The OCR engine utilized, `PaddleOCR`, is known for its accuracy and performance in detecting text in images with various orientations and distortions.

2 System Flow Diagram

Below is the flow diagram illustrating the OCR system process:

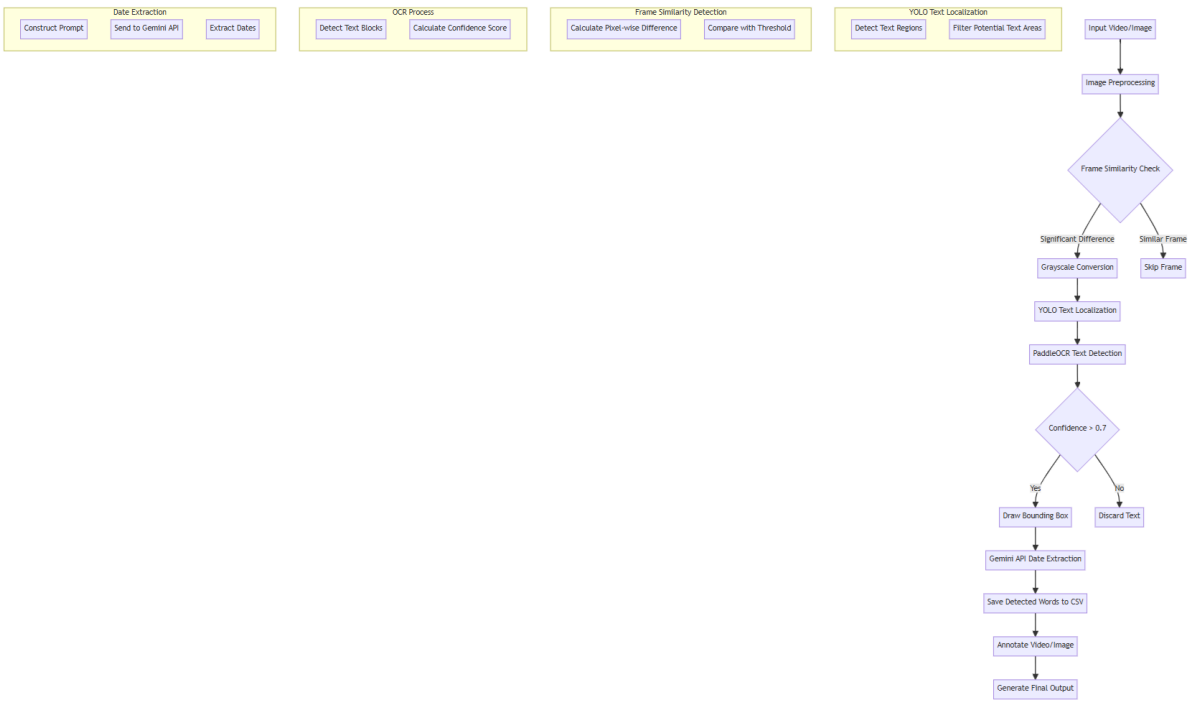

Figure 1: Flow diagram of the OCR system

3 Methodology

3.1 Libraries and Dependencies

The following Python libraries and tools were used in the project:

- **OpenCV**: For image processing, resizing, and annotation.
- **PaddleOCR**: A powerful library for Optical Character Recognition, enabling text extraction from images.
- **Google Gemini API**: Used to process and predict relevant information, such as manufacturing and expiry dates, from the OCR text.
- **Gradio**: A tool to create a simple user interface for video and image processing tasks.
- **YOLO (You Only Look Once)**: A state-of-the-art object detection algorithm for localizing regions in an image that contain text.

The environment is configured with the necessary keys for Google Gemini API integration, PaddleOCR, and YOLO.

3.2 Frame Preprocessing and OCR

The OCR system follows these steps for processing images and video frames:

1. **Image Preprocessing**: Each input frame is resized to a standard width of VGA size (640 x 480 pixels) while maintaining the aspect ratio. The resized frame is then converted to grayscale to enhance text detection performance.
2. **OCR Application**: After preprocessing, `PaddleOCR` detects and recognizes the text in the image or video frame.
3. **Bounding Box and Text Annotation**: For each detected word, a bounding box is drawn around the text, and the OCR confidence score is annotated on the frame.
4. **Frame Similarity Detection**: To reduce redundant OCR processing in video streams, the algorithm calculates pixel-wise differences between consecutive frames and skips frames with no significant changes.

3.3 Algorithms

Below are the core algorithms used in the OCR system.

3.3.1 Frame Similarity Detection Algorithm

The following algorithm detects the similarity between consecutive frames in a video. It calculates the pixel-wise difference between frames to identify frames with significant differences, helping to reduce redundant OCR processing.

Algorithm 1 Frame Similarity Detection

Require: *video_path* (path to video file), *threshold* (threshold for pixel-wise difference)

Ensure: *non_similar_frames* (list of frame indices with significant differences)

```
cap ← Open video capture from video_path
prev_frame ← None
frame_count ← 0
non_similar_frames ← empty list
while frame is read from cap do
    gray_frame ← Convert frame to grayscale
    if prev_frame ≠ None then
        frame_diff ← Compute absolute difference between prev_frame and gray_frame
        diff_sum ← Sum of all pixel values in frame_diff
        if diff_sum > threshold then
            Add frame_count to non_similar_frames
        end if
    end if
    prev_frame ← gray_frame
    frame_count ← frame_count + 1
end while
return non_similar_frames
```

3.3.2 YOLO Text Localization Algorithm

This algorithm uses YOLO to detect and localize the areas in the image or video frame that contain text. YOLO is a deep learning model trained for

object detection tasks, and in this case, it is used to identify regions where text might be present.

Algorithm 2 YOLO Text Localization

Require: *frame* (input image or video frame), *yolo_model* (pre-trained YOLO model for text detection)
Ensure: *text_regions* (list of bounding boxes around detected text)

```
Apply YOLO model to frame
detections ← YOLO model output
text_regions ← empty list
for each detection in detections do
    if detection corresponds to text then
        Add bounding box of detection to text_regions
    end if
end for
return text_regions
```

3.3.3 Text Detection and OCR Algorithm

This algorithm detects text from the processed image or video frame and annotates it with confidence scores.

Algorithm 3 Perform OCR on Image or Frame

Require: *frame* (input image or video frame)

```
Apply OCR using PaddleOCR on the image/frame
for each detected text block do
    if confidence of text detection is above 0.7 then
        Annotate bounding box around the text
        Display text with confidence score
    end if
end for
```

3.3.4 Text Extraction and Date Prediction Algorithm

This algorithm uses the Google Gemini API to extract manufacturing and expiry dates from the OCR results.

Algorithm 4 Extract Dates from OCR Text

Require: *text* (OCR extracted text)

1: Construct prompt on text
2: Send the prompt to Google Gemini API
3: Extract the predicted dates from the response
4: **return** Extracted dates

3.4 Final Output

The processed video is saved with annotations of the detected text and the confidence scores. Additionally, the detected words are saved in a CSV file for further analysis or reporting. The final output includes:

- An annotated video with bounding boxes around detected text.
- A CSV file containing the detected words, their confidence scores, and their positions.

4 Results

The system successfully detects text from product packaging images and video frames. The bounding boxes and confidence scores are accurately displayed on the video frames. The Gemini model was able to predict the manufacturing and expiry dates based on the extracted text with good accuracy. The following key points were observed during testing:

- OCR accurately detected product-related text such as dates and MRP.
- The frame similarity detection function helped reduce redundant OCR processing.
- The YOLO algorithm successfully localized the text regions in the frames, improving the overall text detection.
- The system's ability to predict dates based on extracted text was effective, with a reasonable success rate in predicting valid manufacturing and expiry dates.

5 Conclusion

This OCR system demonstrated strong capabilities for extracting and processing text from product packaging images and video frames. The integration with the Gemini model enhanced the system's ability to predict relevant product information, such as manufacturing and expiry dates. The addition of YOLO for text localization improved the text detection accuracy. The project shows potential for real-world applications in inventory management and product tracking.