Line extractor

1. Introduction

The "Line Extractor" project represents a significant advancement in the field of document processing and information extraction, specifically tailored to the context of Egyptian National Identification Cards. National identification cards are fundamental components of a country's administrative and legal infrastructure, playing a pivotal role in citizen identification, access to government services, and a wide range of socio-economic activities.

The "Egyptian National ID Line Extractor" project is a comprehensive solution designed to automate and streamline the process of extracting essential information from these identification cards. This project employs cutting-edge computer vision and optical character recognition (OCR) techniques to accurately identify, segment, and extract text lines from scanned or captured images of Egyptian National IDs.

This report provides an in-depth exploration of the methodologies, technologies, and processes implemented in the "Egyptian National ID Line Extractor" project. It delves into the technical aspects of object detection, rotation, and data extraction.

1. **Dataset**

The dataset utilized for the "Line Extractor" project was sourced from Roboflow, a platform that provides a diverse range of pre-annotated image datasets for computer vision tasks. The dataset obtained from Roboflow consisted of a collection of scanned and captured images of Egyptian National Identification Cards.

1. **Methodology**
   1. **Object detection**

To overcome these challenges, YOLO can be used for Egyptian id images object detection. YOLO is a deep learning-based object detection algorithm that has shown promising results in various applications. The key feature of YOLO is its single-stage detection approach, which is designed to detect objects in real time and with high accuracy. Unlike two-stage detection models, such as R-CNN, that first propose regions of interest and then classify these regions, YOLO processes the entire image in a single pass, making it faster and more efficient. In this project the Yolo that used was YOLOv8 which has better accuracy than previous YOLO models.

* **New convolutions in YOLOv8**

**Anchor-free detection** is when an object detection model directly predicts the center of an object instead of the offset from a known anchor box.

**Anchor boxes** are a pre-defined set of boxes with specific heights and widths, used to detect object classes with the desired scale and aspect ratio. They are chosen based on the size of objects in the training dataset and are tiled across the image during detection.

The network outputs probability and attributes like background, IoU, and offsets for each tiled box, which are used to adjust the anchor boxes. Multiple anchor boxes can be defined for different object sizes, serving as fixed starting points for boundary box guesses.

The advantage of anchor-free detection is that it is more flexible and efficient, as it does not require the manual specification of anchor boxes, which can be difficult to choose and can lead to suboptimal results in previous YOLO models such as v1 and v2.

* **Implementation**

1. **Required installed libraries**

!pip install ultralytics

!pip install squarify

1. **Training phase**

During the training phase, the YOLOv8 model undergoes a process of optimization to learn and recognize the unique features and patterns present in Egyptian National ID cards. This involves the model learning to identify ID region. These learned features allow the model to perform precise object detection.

Training Metrics and Loss

* Train Box Loss: The train box loss metric measures the difference between the predicted bounding boxes and the actual bounding boxes of the objects in the training data. A lower box loss means that the model's predicted bounding boxes more closely align with the actual bounding boxes.
* Train Class Loss: The train class loss metric measures the difference between the predicted class probabilities and the actual class labels of the objects in the training data. A lower-class loss means that the model's predicted class probabilities more closely align with the actual class labels.
* Train DFL Loss: The train DFL (Dynamic Feature Learning) loss metric measures the difference between the predicted feature maps and the actual feature maps of the objects in the training data. A lower DFL loss means that the model's predicted feature maps more closely align with the actual feature maps.
* Metrics Precision (B): The metrics precision (B) metric measures the proportion of true positive detections among all the predicted bounding boxes. A higher precision means that the model is better at correctly identifying true positive detections and minimizing false positives.
* Metrics Recall (B): The metrics recall (B) metric measures the proportion of true positive detections among all the actual bounding boxes. A higher recall means that the model is better at correctly identifying all true positive detections and minimizing false negatives.
* Metrics mAP50 (B): The metrics mAP50 (B) metric measures the mean average precision of the model across different object categories, with a 50% intersection-over-union (IoU) threshold. A higher mAP50 means that the model is better at accurately detecting and localizing objects across different categories.
* Metrics mAP50-95 (B): The metrics mAP50-95 (B) metric measures the mean average precision of the model across different object categories, with IoU thresholds ranging from 50% to 95%. A higher mAP50-95 means that the model is better at accurately detecting and localizing objects across different categories with a wider range of IoU thresholds.

**A graph of a training metrics

Description automatically generated**

1. **Evaluation phase**

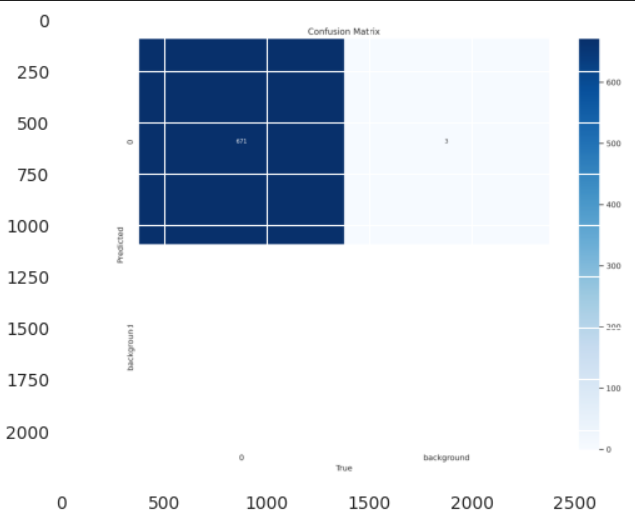
After an intensive training phase, the YOLOv8 model for the "Line Extractor" project proceeds to the evaluation phase. This phase is of paramount importance in assessing the model's performance, accuracy, and its readiness for real-world deployment.

Metrics and Benchmarks

During the evaluation phase, several key metrics and benchmarks are used to gauge the effectiveness and efficiency of the YOLOv8 model. These metrics provide insights into how well the model generalizes to real-world data and handles the intricacies of Egyptian National ID cards. Some of the primary evaluation metrics include:

* Mean Average Precision (mAP): mAP is a pivotal metric in assessing the overall object detection performance. It quantifies how well the model correctly identifies and localizes text lines on the ID cards.
* Precision and Recall: Precision and recall metrics provide insights into the model's ability to make accurate predictions and capture all relevant text lines in the ID card images.
* F1 Score: The F1 score combines precision and recall into a single metric, offering a comprehensive view of the model's performance.

A graph showing a number of different colored squares

Description automatically generated with medium confidence 

1. **Prediction phase**

Prediction phase is testing the YOLOv8 model with real-world data comprising scanned and captured images of Egyptian National ID cards.



* 1. **Rotation id**

One critical step in the document processing pipeline involved the rotation of detected ID cards before utilizing the YOLOv8 detector for extraction and text recognition. This pre-processing step was essential to ensure that the ID cards were correctly oriented, as some ID cards were skewed or not aligned horizontally.

A person's hand pointing at a passport

Description automatically generated A close-up of a currency

Description automatically generated

Before rotation

After rotation 180

* 1. **Line segment**

line segmentation plays a vital role in isolating individual lines of text from a larger document, making it more amenable for subsequent processing and analysis. To achieve this, I employed pytesseract, a powerful Optical Character Recognition (OCR) tool

* **Optical Character Recognition (OCR)**

Optical Character Recognition (OCR) is the process that converts an image of text into a machine-readable text format. For example, if you scan a form or a National ID, your computer saves the scan as an image file. You cannot use a text editor to edit, search, or count the words in the image file. However, you can use OCR to convert the image into a text document with its contents stored as text data.

**The OCR engine or OCR software works by using the following steps:**

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 such as Deskewing, Despeckling
3. Text recognition: The two main types of OCR algorithms or software processes that an 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 similar font and scale 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 a computerized file. Some OCR systems can create annotated PDF files that include both the before and after versions of the scanned document.

* **Implementation**
* **Requirement before execute code**

1. Download tesseract exe from <https://github.com/UB-Mannheim/tesseract/wiki>.
2. Install this exe in C:/Program Files/Tesseract-OCR
3. Open virtual machine command prompt in windows or anaconda prompt.
4. Run pip install pytesseract
5. **Text Recognition with Pytesseract**: I began by applying pytesseract to the entire document image. Pytesseract employs machine learning models to recognize text within an image and extract it. The recognized text is stored as a string.
6. **Customized OCR Settings:** For Arabic text, I adjusted pytesseract's configuration by setting the Page Segmentation Mode (PSM) to '6,' which is suitable for multi-line text. This configuration ensured that pytesseract correctly recognized Arabic text and maintained the text's integrity within the lines.
7. **Results**

The Egyptian national id dataset was trained for 50 epochs in YOLOv8 for object detection. Key assets of the training model were checked regularly which are: the loss function value, precision, recall, default learning rate. The performance of YOLOv8 measured by Mean Average Precision (mAP) where mAP50 was 99.4%, mAP75 was 99.4%, and mAP50-95 was 97.6%. After that, the Rotation ID method created to rotate the ID by any rotation angle from horizontal to vertical. Finally, the Pytesseract model used to extract the text line from detected id.