Lines Extraction

Introduction

In this report, I'm excited to share a new system I developed that can neatly pull-out lines of text from images of ID cards. What's special about this system is that it doesn't matter how the IDs are placed, it can handle them whether they're upright or on their side.

For this task, I chose a powerful model which is YOLOv8m-Seg from Ultralytics. It's part of the YOLO family, famous for its ability to quickly find and outline objects in pictures. The latest version is not just faster but also sharper than its predecessors, which helped a lot.

I trained this clever system with special images from Roboflow, where each picture has clear markings to guide the system in recognizing objects with masks and boundary boxes. This was crucial in teaching the system to spot and segment ID cards.

Another key component was ArabicOCR, specifically used for reading Arabic script. It was the best choice for the job, although it sometimes gave mixed results—often very good, but occasionally not as accurate. Despite this, it played a significant role in helping the system understand and process the Arabic text on the IDs.

Moreover, this system has the ability for adjusting images that aren't straight. It can switch them from vertical to horizontal and even fix any tilted angles. That way, the text is always set up right for reading.

All the hard work was worth it because the system turned out to be pretty accurate. It's great at spotting where text is and figuring out the start and end of each line. This makes me hopeful that it can make the job of transferring info from ID cards to computers much quicker and easier.

In the pages that follow, I'll dive deeper into how I built this system, prepped the images, and achieved solid results. I'm eager to demonstrate the potential of this

system in not just reading ID cards but perhaps in other document processing tasks as well.

Methodology

- **ID Card Detection:**
  - The first and crucial step in My text extraction pipeline was the accurate detection of ID cards within the images. Given the variability in orientation and potential skew, the system needed to be robust and flexible. To achieve this, I used YOLOv8m-Seg, an advanced neural network model known for its exceptional performance in object detection tasks.
- **Model Selection and Training:**
  - YOLOv8m-Seg 's ability to process images quickly and with high precision made it an ideal choice for the task. I trained the model using a dataset obtained from Roboflow, which provided me with a diverse set of annotated images of ID cards. These annotations included masks and boundary boxes, essential for teaching the model to localize and classify regions of interest within the images.
- **Orientation Handling:**
  - A common issue in document scanning is the misalignment of the document's orientation. My system tackles this by incorporating orientation detection that identifies the correct alignment of the ID card. Whether the card is presented vertically or horizontally, the system can correctly orient the image to prepare it for subsequent processing stages.
- **Skew Correction:**
  - In addition to orientation handling, the system also addresses the problem of skew correction. Utilizing geometric transformations, the system can detect and rectify any angular misalignment. This process ensures that the text on the ID cards is aligned horizontally, which is crucial for accurate line segmentation and OCR performance.
- **Binary Thresholding:**
  An essential preprocessing step in my pipeline is the conversion of the image to a binary format where the text is white, and the background is

black. This simplification is achieved through a carefully calibrated thresholding technique that strips away the nuances of color and shading, leaving behind a stark, high-contrast image that accentuates the text. By doing so, I eliminate any distracting elements, allowing my system to focus solely on the text.

- **Integration with OCR:**
  - With the ID card isolated and the image in binary form, I bring in ArabicOCR to interpret the Arabic script. Although ArabicOCR delivers varying levels of output quality, it remains the tool of choice for parsing and digitizing the Arabic text on the cards.

Data Collection

The dataset comprised a total of 1446 images, divided into training, validation, and testing sets. The training set, the largest of the three, contained 1260 images. This extensive collection was useful in teaching the model to accurately detect and segment ID cards. The validation set consisted of 120 images, which played a pivotal role in fine-tuning the model parameters and preventing overfitting. Lastly, the testing set included 60 images, serving as the ultimate benchmark for the model's performance and its ability to generalize to new, unseen data.

Implementation

- **Training the Model:**
  - The implementation phase of the project focused on training the YOLOv8m-Seg model to recognize and segment text lines from the ID cards. The training was set to run for a maximum of 20 epochs, which provides the model with sufficient exposure to learn from the data without unnecessarily extending the training time.
- **Configuration Details:**
  - To optimize the learning process, I configured the training with a batch size of 16, ensuring a balance between computational efficiency and model performance. The images were resized to a uniform dimension of 640x640 pixels, which is large enough to capture the necessary details without overburdening the computational resources.

    - The learning rate, a critical hyperparameter that determines the step size at each iteration while moving toward a minimum of the loss function, was set at 0.01, providing consistency in the learning rate throughout the training process. To maintain the stability of the update steps, I used a momentum value of 0.937, which helps in accelerating the gradients vectors in the right directions, leading to faster converging. Furthermore, I applied a weight decay of 0.0005, which aids in regularizing the model by preventing its weights from growing too large.

- **Early Stopping Mechanism:**
    - To prevent the model from overfitting, I used an early stopping technique. This method monitors the model's performance on the validation set after each epoch and halts the training if the model begins to perform worse on the validation data—indicative of overfitting. In practice.
    -

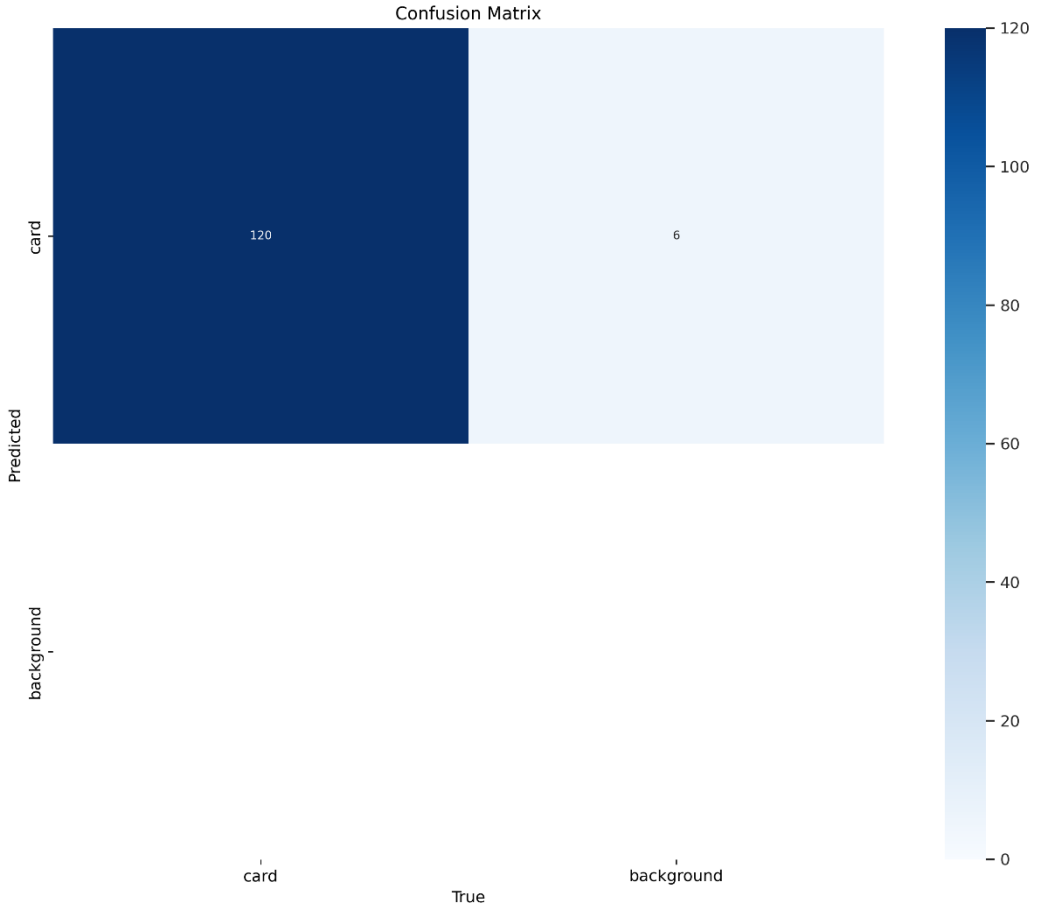
Results and Evaluation Metrics

The model was trained for a total of 4 epochs, with early stopping applied to prevent overfitting. This decision was based on the model's performance on the validation set, ensuring that I captured the model's learning at its most effective point.

- **Key Metrics:** Ultralytics utilized several metrics to gauge the performance of my model, including precision, recall, and mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds. These metrics are crucial for understanding how well my model performs in terms of both accuracy and reliability.
    - **Precision and Recall:** The model achieved high precision and recall, indicating its ability to correctly identify and segment ID cards where precision is 99% recall is 99% also.
    - **Mean Average Precision (mAP):** mAP evaluated at two IoU thresholds - 50% (mAP50) and 50-95% (mAP50-95). These metrics provide insights into how well the model performs across different levels of detection difficulty. A high mAP at these thresholds is indicative of the model's

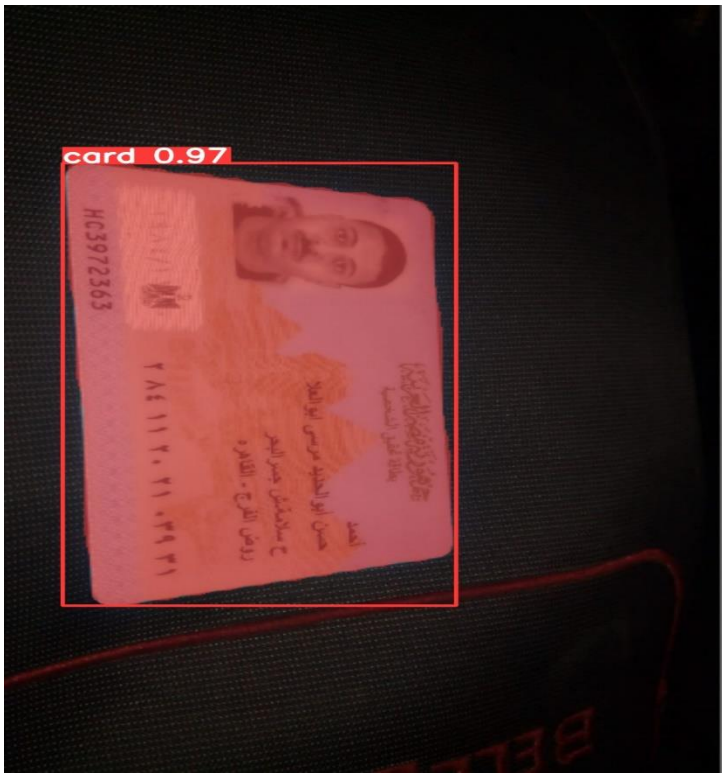
robustness in accurately detecting and segmenting ID cards where mAP50 is 99.5% and mAP95 is 97.65.

- **Confusion Matrix:** The confusion matrix further corroborated the model's effectiveness. It revealed a high number of True Positives (TP), indicating successful ID card detections, and a low number of False Negatives (FN), suggesting few misses. Since the dataset exclusively contained "card" instances without any "background," False Positives (FP) and True Negatives (TN) were not applicable. This matrix provides a clear picture of the model's ability to correctly detect ID cards.

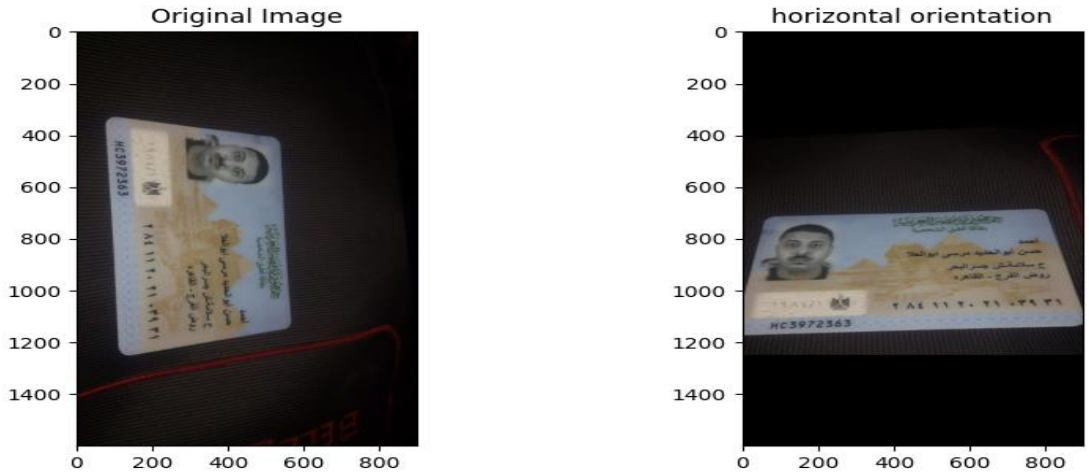


Demo

Initially, I obtained the predictions for boundary boxes and masks in the following manner:



I identified the vertical orientation of the image and rectified it. This adjustment also involved transforming the boundary boxes to align with the horizontal orientation. This was achieved using the **`ensure_horizontal`** function, which



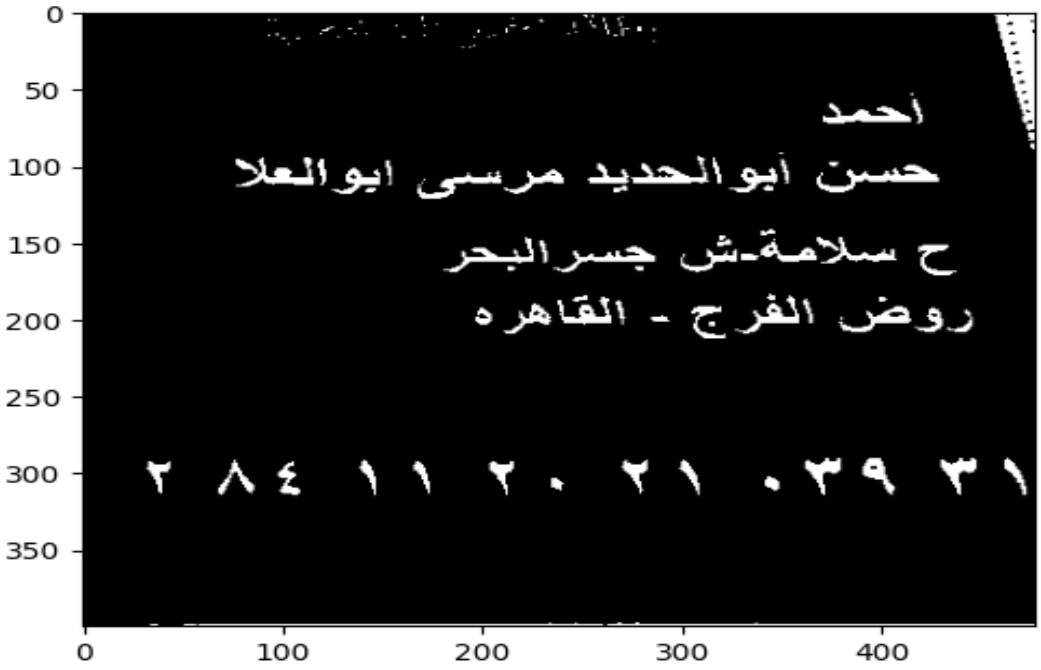
accepts the original image and predicted boxes as inputs and returns the rotated image along with updated coordinates in cases where the image is detected as vertically oriented.

At this stage, I proceed to crop the image based on the region of interest that was previously predicted. This is accomplished using the **`crop_image_with_bbox`** function, which yields the cropped image.

Next, the skewed angle is corrected using the **`correct_skew`** function, which delivers an image with the corrected orientation.

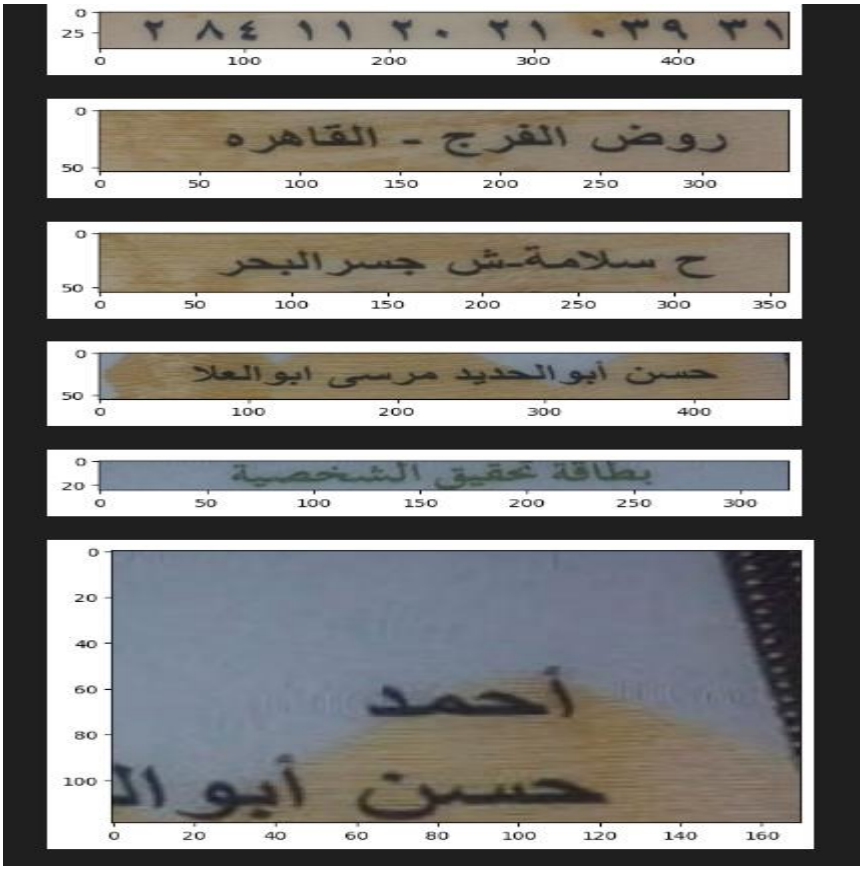
Now, I divide the image into two distinct parts based on our specific requirements. For our use case, one part consists of the textual data from the ID card, while the other part includes the individual's image. This division aids in the efficient extraction of text, and simultaneously, if needed, allows for facial verification against another image. This segmentation is executed using a constant split, based on the area of interest, where 40% of the ID contains the face, and the remaining 60% comprises the data we aim to extract.

Moving on to a crucial step in preparation for OCR, I apply binarization. This involves setting a threshold on the cropped image to eliminate all pixels, leaving only black and white, for simplifying the image for text extraction.

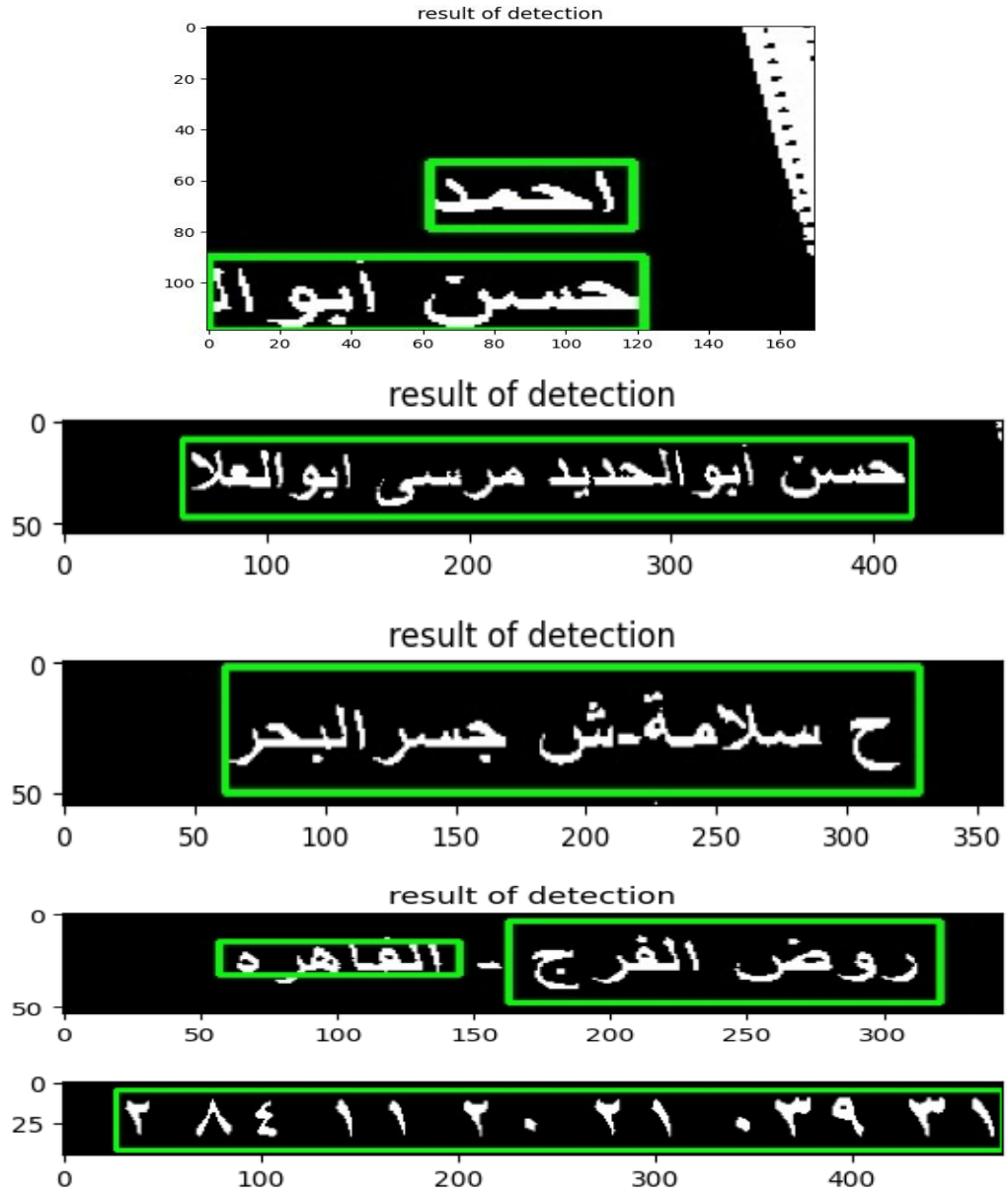


Now, I will create a mask for each line to segment them individually, followed by cropping these segments using the **`creat_masks`** function.

Next, I find contours to derive bounding boxes, which is achieved using the **`get_segments`** function.



Now, our ArabicOCR is ready to process the inputs, which consist of segmented and cropped lines, each preprocessed individually before analysis.




```
[['٢ ٨٤ ١١ ٢٠ ٢١ ٠٣٩ ٣١'],
 ['روض الفرج' و 'القاهره'],
 ['ح سلامة ش جسر البحر'],
 ['حسن ابو الحديد مرسى ابو العلا'],
 [],
 ['احمد' و 'حسن آبوا']]
```