ID Detection: Classical and Deep Learning Approaches

Contents

1 Introduction

This report presents two methods for ID card detection and segmentation:

1. **Classical Approach**: This method uses basic image processing techniques like edge detection and contour detection to identify and segment ID cards.
2. **Deep Learning Approach**: This method involves training two YOLO models. The first model detects ID cards, and the second model segments and labels the specific text fields in the ID.

Both approaches aim to detect ID cards in images and extract relevant text fields such as name, address, and other details.

2 Classical Approach: ID Detection Using Image Processing

2.1 Methodology

The classical approach relies on traditional image processing steps. The main steps in this pipeline are:

1. **Image Loading**: The image is loaded from the file.
2. **Preprocessing**: The image is converted to grayscale, and Gaussian blurring is applied to reduce noise.
3. **Edge Detection**: The preprocessed image undergoes edge detection using the Canny edge detector. This highlights the boundaries of the ID card.
4. **Contour Detection**: The detected edges are used to find contours, and the largest contour is assumed to be the ID card.
5. **Corner Detection**: Corners of the ID are detected from the contour and used to correct the image's perspective.
6. **Image Rotation**: The detected corners are used to rotate the image to align the ID card properly.
7. **Region Cropping**: The ID region is cropped from the image.
8. **Segmentation**: The cropped ID is processed further to segment text fields such as the name or address.

2.2 Code Pipeline

```
def pipeline(image_path, output_path):
    img = load_image(image_path)
    processed_img = preprocessing(img)
    largest_contour = edge_detection(processed_img)
    points = detect_corners_from_contour(largest_contour)
    sorted_points = sort_coordinates_x_y(points)
    rotated_img, keypoints = rotate_image(img, sorted_points)
    region_id = crop_region_id(rotated_img, keypoints)
    segmented_img = preprocess_and_segment(region_id)
```

2.3 Sample Input and Output

Input: An image containing an Egyptian ID card.

Output: Images showing cropped, rotated, and segmented versions of the ID card, highlighting key text fields.

Below are sample input images of ID cards, followed by the corresponding output images showing cropped and segmented versions of the ID card.

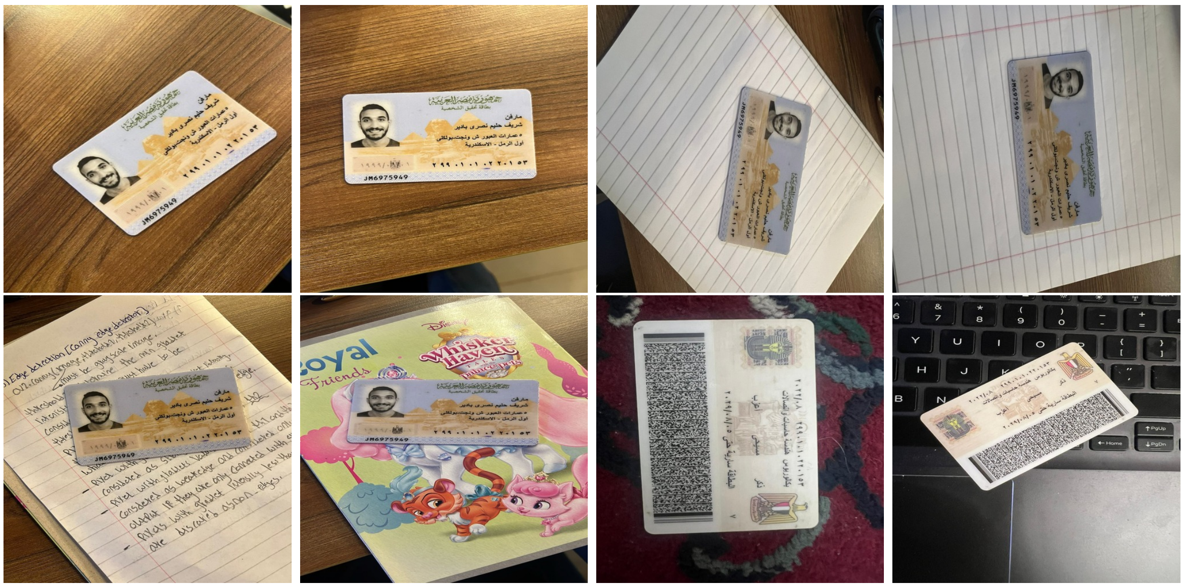
Figure 1: Sample input images containing ID cards.

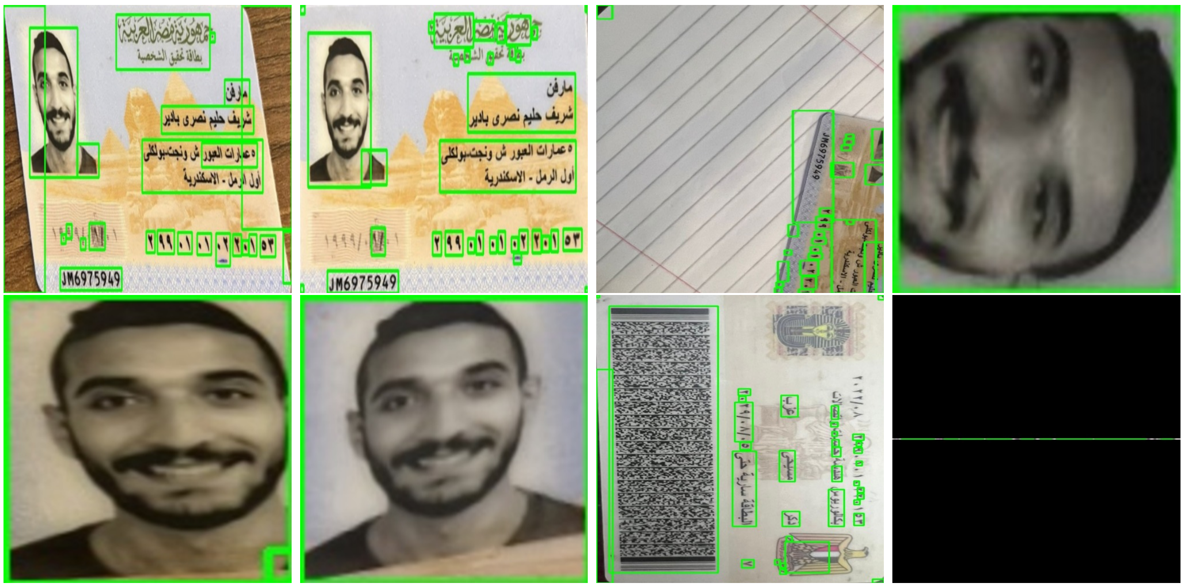
Figure 2: Output images after processing ID cards: cropped, rotated, and segmented versions.

2.4 Results

The classical approach works well in simple settings where the ID is easy to see and the lighting is good. However, in more complex situations with shadows, poor lighting, or unusual angles, the method becomes less reliable.

3 Deep Learning Approach: ID Detection and Segmentation Using YOLO

3.1 Methodology

The deep learning approach uses two YOLO models, each trained for a specific task related to Egyptian ID cards:

1. **YOLO Model 1: ID Card Detection**: The first YOLO model is fine-tuned using an Egyptian ID cards dataset from Roboflow. This model is specifically trained to detect the ID card by drawing a bounding box around it, ensuring accurate detection in diverse environments.

2. **YOLO Model 2: Text Line Segmentation and Labeling**: The second YOLO model is fine-tuned using a different Egyptian ID cards dataset, also from Roboflow, which focuses on text line segmentation and field labeling. After the ID card is cropped from the first model's output, this model detects and segments specific fields, such as first name, last name, address, gender, ID number, and other essential information.

Both datasets are referenced in the reference section, providing the source for the fine-tuning of these models. The bounding box from the first model is used as input to the second model, which then segments and labels the fields inside the ID card.

3.2 Code Pipeline

```
# Load YOLO models
model_id_detection = YOLO("id_detection_model.pt")
model_text_segmentation = YOLO("text_segmentation_model.pt")

# Detect ID card using the first model
results_id = model_id_detection(image)

# Extract bounding box of the detected ID
for box in results_id.xyxy[0]:
    x1, y1, x2, y2, conf, cls = box
    cropped_id = image[int(y1):int(y2), int(x1):int(x2)]

    # Pass the cropped ID to the second YOLO model for segmentation
    results_text = model_text_segmentation(cropped_id)

    # Extract and label detected fields
    for text_box in results_text.xyxy[0]:
        x1_text, y1_text, x2_text, y2_text, conf_text, cls_text = text_box
        label = model_text_segmentation.names[int(cls_text)]
        cropped_text = cropped_id[int(y1_text):int(y2_text), int(x1_text):int(x2_text)]
```

3.3 Sample Input and Output

Input: An image containing an Egyptian ID card.

Output:

1. **YOLO Model 1**: Detects and crops the ID card.
2. **YOLO Model 2**: Segments specific text fields, such as name, address, religion, and other details.

Below are sample input images of ID cards, followed by the corresponding output images showing cropped and segmented versions of the ID card.

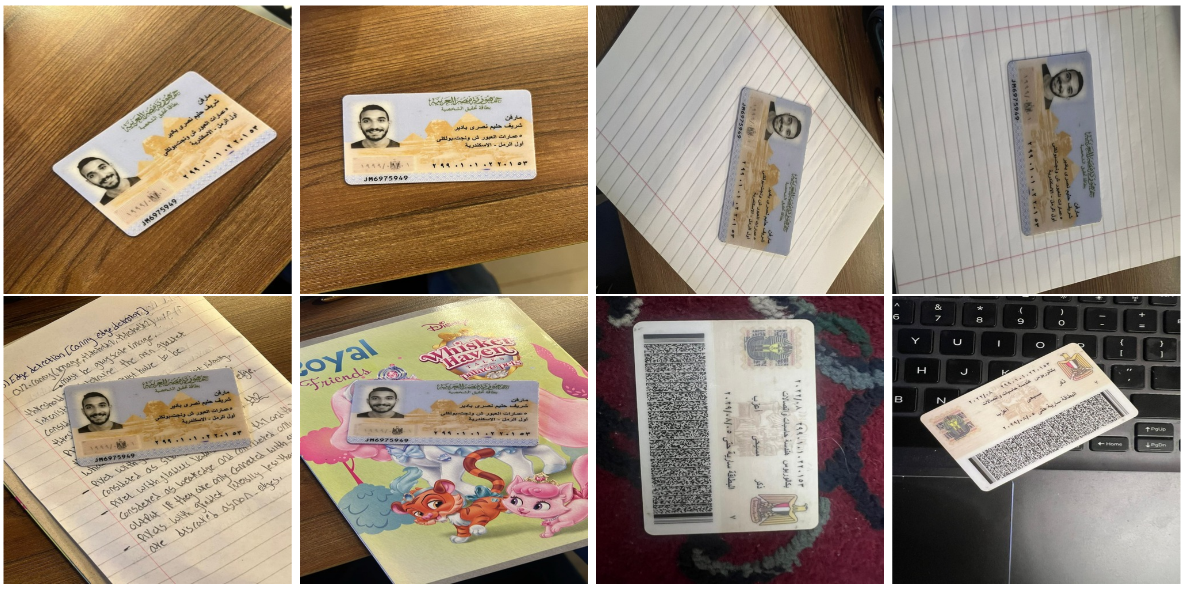

Figure 3: Sample input images of ID cards.

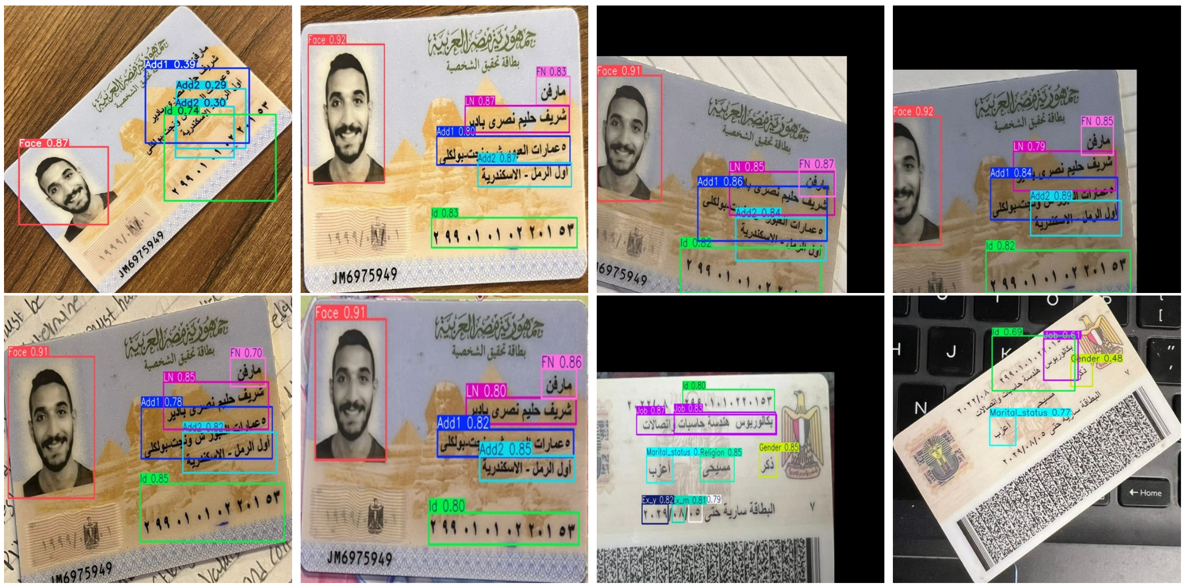

Figure 4: Output images after processing: cropped and segmented ID cards.

3.4 Results

The deep learning approach provides a more accurate and flexible solution. It works better in complex environments where lighting may be poor, or the ID card may be partially obscured. The use of two YOLO models ensures both detection and segmentation are handled efficiently. However, we encountered challenges with the object detection aspect of the first YOLO model. Specifically, the bounding box detected by YOLO is often parallel to the x-axis, but if the card is not horizontally aligned, this misalignment affects the output of the second model responsible for segmentation. We attempted to rotate the bounding box to make the card inside it parallel to the x-axis. This approach worked in many cases but failed in some instances, particularly when the bounding box was nearly square-shaped. In such cases, it became difficult to determine the card's orientation accurately, leading to segmentation errors. Further improvements in handling the alignment of the detected box could help address this issue

4 Conclusion

Both methods have their strengths:

- **Classical Approach**: Effective in well-controlled environments with good lighting. However, it struggles in more challenging scenarios.
- **Deep Learning Approach**: More flexible and accurate, especially in complex settings. It's better suited for real-world applications with varying conditions.

For practical use, especially in challenging environments, the deep learning approach is recommended.

5 References

- YOLO: `https://www.datacamp.com/blog/yolo-object-detection-explained`
- ultralytics YOLOv8: `https://docs.ultralytics.com/#how-can-i-get-started-with-yolo-installation-and-setup`
- Dataset 1 (ID Detection): `https://universe.roboflow.com/shalaby/detect-egyptian-national-id/dataset/3`
- Dataset 2 (Text Line Segmentation): `https://universe.roboflow.com/elsoudy/card-hleyg/dataset/1`