ID CARD DETECTION USING YOLO V7

1. INTRODUCTION

1.1 Overview

ID card detection and data extraction is an important task in various applications such as security, identity verification, and access control. In this process, an image of an ID card is captured and the relevant data is extracted from it. This data can include personal information such as name, date of birth, and ID number. The process of ID card detection and data extraction involves several steps, including image acquisition, image preprocessing, feature extraction, and data recognition. Various techniques such as machine learning, computer vision, and deep learning are used to perform these tasks and improve the accuracy of the process. The successful implementation of ID card detection and data extraction has numerous benefits, including increased security, faster and more accurate identity verification, and streamlined access control processes. This technology has become increasingly important in recent years, especially with the rise of digital identification and remote verification. Overall, ID card detection and data extraction is a crucial process in various industries and has the potential to revolutionize the way we approach identity verification and access control.

1.2 Purpose

Recently, an incident happened in Manipur. Chief Minister Biren Singh of Manipur emphasized the pressing concern of illegal immigration, pointing out that a significant number of individuals from Myanmar, around 410 in total, had entered the state without proper documentation. Ensuring a proactive stance, the government has devoted considerable attention and efforts to address the issue of illegal immigrants. In this regard, the Chief Minister expressed the belief that implementing a facial detection system would greatly aid in effectively verifying the status of Inner Line Permit holders whose permits have expired, thereby facilitating more efficient management of the Inner Line Permit System. So, this id card detection helps in many ways. There are many problems with manual id card detection. People have to wait in queue for more time which will be very uncomfortable, also there are chances that

data entry person may enter wrong data which leads to errors. This manual inspection of data is unmanageable sometimes. So, to overcome these problems, automated id detection technique will be useful. Our primary objective is to streamline and automate the process of image analysis for ID cards, specifically focusing on creating a comprehensive database that encompasses not only the ID card images but also crucial information contained within them.

2. LITERATURE SURVEY

2.1 Existing problem

This kind of recognition of characters from the image are growing very rapidly. In 2005, Gabor filters are used for characters that are there in low quality images and for Chinese-readable characters. In 2011, By using histogram analysis document segmentation has been developed . In 2012,Morphological transition method has been developed. In 2015 , a research has been conducted on Indonesian people’s Id card using machine learning algorithms. But, all previous researches used machine learning algorithms and many more but here we are using new version of you only look once i.e YOLO v7 and has increased the accuracy of id card detection.

2.2 Proposed solution

Our project focusses on two phases. In this first phase, we have to choose a valid id card and perform auto-cropping, locating horizontal, facial photo removal and text/non-text segmentation. We have to focus on data extraction. In the second phase, we have to find a method for data extraction and also to retrieve information. Later, this retrieved information should be stored in an excel sheet.

3. THEORITICAL ANALYSIS

3.1 Block diagram

Diagrammatic overview of the project.

3.2 Hardware / Software designing

Hardware Design:

1. Camera: You'll need a camera to capture images or video footage for ID card detection. The camera should be capable

of capturing high-quality images with sufficient resolution and frame rate.

2. Memory: Sufficient RAM is necessary to store the model and process the captured frames efficiently. The memory capacity should be able to handle the size of the YOLOv7 model and the input data.
3. Storage: You may need storage space to save the captured images or video footage for future reference or analysis.

Software Design:

1. YOLOv7 Framework: Install the YOLOv7 framework on your chosen development environment. You can use popular deep learning frameworks such as TensorFlow or PyTorch to implement YOLOv7. Follow the installation instructions provided by the framework's documentation.
2. Dataset Preparation: Collect a dataset of ID card images with bounding box annotations. The dataset should include various samples of ID cards with different backgrounds, orientations, and lighting conditions. Annotate the ID cards in the images with bounding boxes to indicate their locations.
3. Model Training: Train the YOLOv7 model using the prepared dataset. This involves feeding the annotated images into the model and optimizing its parameters to learn how to detect ID cards accurately. Training can take a significant amount of time and may require a powerful GPU for faster processing.
4. Model Optimization: Fine-tune the trained model to improve its performance. Experiment with different hyperparameters, data augmentation techniques, and model architectures to achieve better accuracy and faster inference speed.
5. Integration and Deployment: Integrate the trained and optimized YOLOv7 model into your hardware setup. Develop a software application that captures frames from the camera, processes them through the YOLOv7 model for ID card detection, and displays the results in real-time. The

application should be able to handle the communication between the camera, processing unit, and display output.

6. User Interface: Design a user-friendly interface to interact with the application. The interface can include options to start or stop the ID card detection, adjust settings, display detection results, and save captured images or video footage.

4. EXPERIMENTAL INVESTIGATIONS

In this project we have done creating a customized dataset of around 300 images to detect a university id card and we preprocessed the images in robo flow software to create a new Class named id-card with 77% of training set and13% as a validation set 10% as a testing set.

Dataset We have created a customized dataset named idcard-detection-xtiwy/3 in roboflow which is a free open software used to created datasets for training object detecting models like YOLO,VGG, Fast RCNN, Mask R-CNN etc. We are able to get the mAP(mean Average pression) of 86.0% and pression of 87.3% and recall of 87.3%.After around a 400 epochs our data set acquired a least box loss of 0.014 and object loss of 0.007 and a class loss of 0 with 77% of training set and13% as a validation set 10% as a testing set. And then we moved to the next step of training an Object detecting model on this dataset. For our work we are going to take YOLO model to get a better accuracy and speed.

YOLO Yolo(You only look once) is a algorithm which uses neural networks detect real time objects. This is widely used in detecting people, animals, traffic signals etc. because of its speed and accuracy.

YOU ONLY LOOK ONCE(Yolo) is an algorithm which uses neural networks detect real time objects. This is widely used in detecting people, animals, traffic signals etc. because of its speed and accuracy. Yolo uses Convolution Neural networks to detect real time objects object detection is processes as a regression problem and the resultant output ,we will getting the class probabilities of the detected objects. Yolo uses Convolution Neural networks to detect real time objects object detection is processes as a regression problem and the resultant output ,we will getting the class probabilities of the detected objects. This algorithm can done its prediction in only single algorithmic run by taking only one forward propagation through a neural network. The CNN this algorithm uses will predict bounding boxes and various class probabilities at the same time. Yolo works on three techniques:

• Residual blocks: In this technique ,the image was divided into various grids of same dimension SxS . And then each grid cell will be detecting objects that appears within them.

• Bounding Box regression: YOLO uses a single bounding box regression method to predict height, width, centre and class of object y=(pc,bx,by,bh,bw,c)

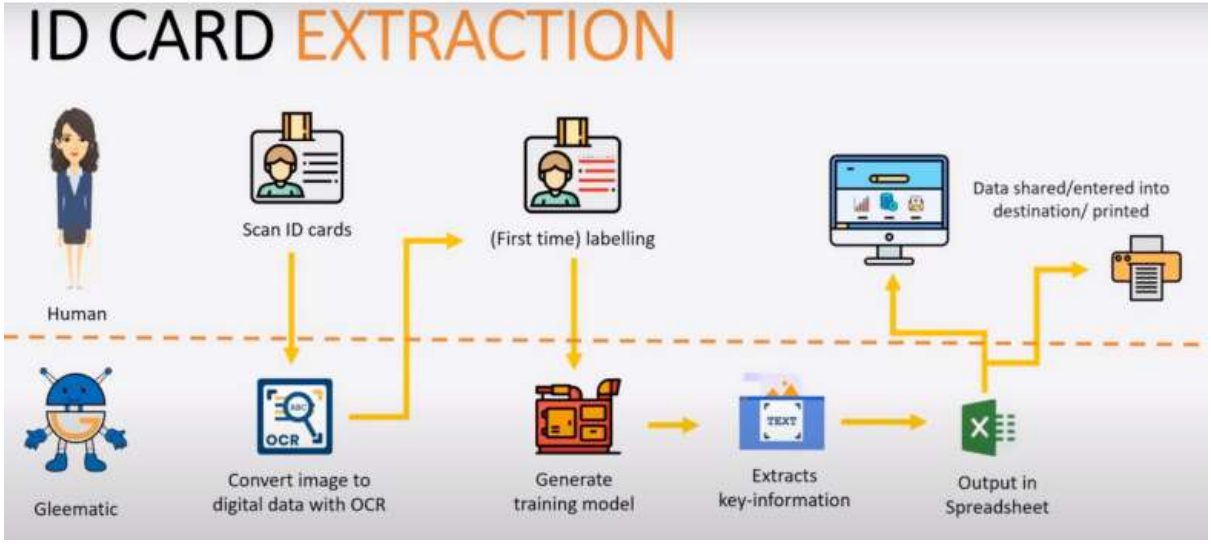
• Intersection over union (IOU): In object detection, the IOU tells us about how two boxes overlap or intersect each other.

Our algorithm(yolo) also uses this technique to detect the boxes that surround the object perfectly. This technique will resulting 1 if the predicted box and real box are exactly equal. The yolo algorithm uses the combination of 3 techniques to predict the final output.

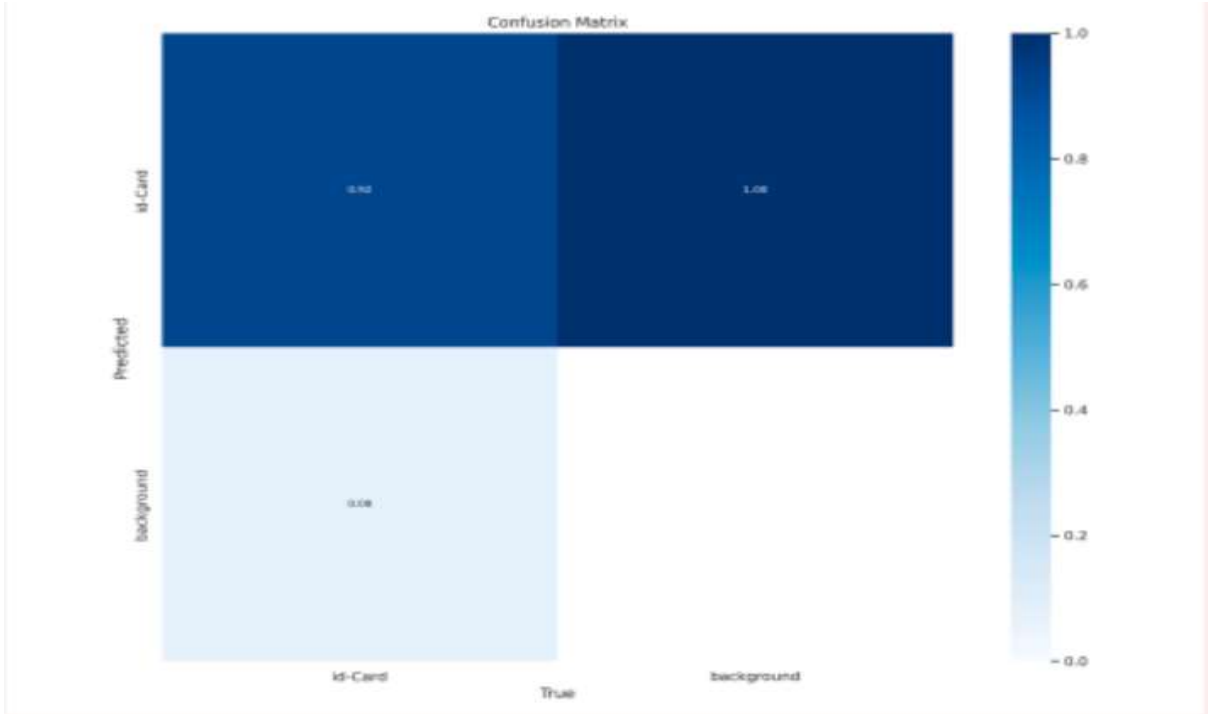
IMAGE TRANSFORMATION In our work we are going to detecting the ID card and we are going to separate it and send it to the next step of OCR of text extraction. For this we are using the YOLO model, the yolo algorithm as a result of detecting an object it will give us the bounding box details of the ID card. We are going to use those ads crop the input image and preprocess the cropped image for text extraction.

TEXT EXTRACTION Tesseract is an open source optical character recognition platform or software. For this we are going to install some modules and importing the requires submodules. After the step of extracting the id card from the input image , we are running the tesseract on it to extract the text from the ID-card image. As of now the text was extracted using the tesseract module, but also have stored the extracted information such as name , registration number , department, day border type etc.. into an excel sheet.

5. FLOWCHART



6. RESULT




Mettu Sai Sanjana Reddy
20BCI7057
BCI SCOPE
DAY SCHOLAR

```
print(text)


Mettu Sai Sanjana Reddy

20BCI7057
BCI SCOPE

DAY SCHOLAR
```

7. ADVANTAGES & DISADVANTAGES

Advantages:

1. Real-time Processing: YOLOv7 is optimized for real-time object detection. It can process images or video frames quickly, allowing for efficient and fast detection of project ID cards.

2. Object Classification: YOLOv7 not only detects project ID cards but can also classify them based on pre-defined classes. This can be useful in scenarios where different types of ID cards need to be identified and processed differently.

3. Open-Source and Community Support: YOLOv7 is an open-source algorithm, which means it has a vibrant community of developers and researchers actively contributing to its improvement. This provides access to a wealth of resources, tutorials, and community support for implementing and fine-tuning the project ID card detection system**.**

Disadvantages:

1. Training Complexity: Training a YOLOv7 model for project ID card detection requires a large amount of labeled training data and significant computational resources. Collecting and annotating a sufficient amount of data can be time-consuming and expensive.

2. Hardware Requirements: Running a YOLOv7 model efficiently often requires high-end hardware, such as a powerful GPU, to achieve real-time processing speeds. This can be a limitation for users with limited computational resources.

3. Fine-tuning and Optimization: While YOLOv7 provides excellent performance out of the box, fine-tuning and optimizing the model for

specific project ID card detection tasks may require expertise and experimentation. Tuning hyperparameters, adjusting the network architecture, or employing data augmentation techniques might be necessary to achieve the desired performance.

8. APPLICATIONS

Automatic License Plate Recognition using YOLOv2 : This study proposes an approach to recognize license plates usingYOLOv2 algorithm, which can also be applied to ID card detection and information extraction.

Passport MRZ Recognition using Deep Learning Techniques : This study proposes an approach to extract information from passport MRZ (Machine Readable Zone) using deep learning techniques, which can also be applied to ID card information extraction.

9. CONCLUSION

With OCR techniques and image preprocessing we can detect id card. Segmentation, text area extraction and preprocessing are included in image processing techniques of this research. OCR is used to detect the characters. Text area of NIK and name of the citizen in the I'd card is recognised by text area extraction consists of kernel. This experiment has created an accuracy between 90 to 100%. After all these efforts we are excited to share this ID card detection and data extraction automated system. We can use this to detect driver’s license and passports and also relevant information from id cards. Our system utilizes advanced computer vision techniques to identify and isolate the ID card from within an image or video feed. Once the card is detected, OCR (Optical Character Recognition) technology is used by our software to extract the relevant data, such as name, address, and date of birth. We have tested our system extensively under various lighting conditions, angles, and ID card types, and have achieved impressive accuracy rates. Our system is capable of processing large batches of ID cards quickly and efficiently, making it ideal for use in industries such as banking, security, and law enforcement. In conclusion, we believe that our ID Card Detection and Data Extraction system represents a significant step forward in the field of computer vision and OCR technology. We are confident that it will prove to be a valuable tool for organizations seeking to streamline their ID card processing workflows and improve overall efficiency.

10. FUTURE SCOPE

Dataset Expansion: To improve the performance and generalization of the ID card detection model, you can collect and annotate a larger and more diverse dataset of ID cards. This can include ID cards from different regions, with varying designs, fonts, and layouts. The expanded dataset will help the model learn and detect a wider range of ID card variations.

Performance Optimization: Explore ways to optimize the performance of your ID card detection system. This can involve model quantization techniques to reduce the model size and inference time, leveraging hardware accelerators (such as GPUs or specialized inference chips), or deploying the model on edge devices for real-time and low-latency processing.

Fine-tuning and Transfer Learning: Consider fine-tuning the pre-trained YOLOv7 model using your specific ID card dataset. This process involves training the model further on your data to adapt it to the specific characteristics and variations of ID cards. Transfer learning can help improve the model's accuracy and reduce the training time.

11. BIBILOGRAPHY

[1]https://guides.nyu.edu/tesseract#:textTesseract%20is%20an%20open%20source,or%20most%20other%20popu

[2] https://nanonets.com/blog/id-carddigitization-deep-learning/

[3] https://www.section.io/engineeringeducation/introduction-to-yolo-algorithm-forobjectdetection/#:~:text=YOLO%20is%20an%20algo

rithm%20that,%2C%20parking%20meters%2C%20and%20animals.

[4] Mithe R, Indalkar S and Divekar N 2013 Optical Character Recognition Int. J. Recent Technol.

[5] Wang X, Ding X and Liu C 2005 Gabor filters-based feature extraction for character recognition

[6] DONGRE V J AND MANKAR V H 2011 DEVNAGARI DOCUMENT

SEGMENTATION USING HISTOGRAM APPROACH Int. J. Comput. Sci. Eng. Inf. Technol

APPENDIX

A. Source Code

```
!pip install ultralytics==8.0.28
from IPython import display
display.clear_output()
import ultralytics
ultralytics.checks()
from ultralytics import YOLO
from IPython.display import display, Image
!mkdir {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="eEuewAprXEIsFbptBZTC")
project = rf.workspace("myproject-vm4hg").project("id-card-detection-xtiwy")
dataset = project.version(3).download("yolov8")
%cd {HOME}
!yolo task=detect mode=train model=yolov8m.pt
data=/content/datasets/id-Card-detection-3/data.yaml epochs=20
imgsz=640
!ls {HOME}/runs/detect/train/
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/confusion_matrix.png'
,
width=600)
%cd {HOME}
```

```
Image(filename=f'{HOME}/runs/detect/train/results.png'
, width=600)
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/val_batch0_pred.jpg'
,
width=600)
%cd {HOME}
!yolo task=detect mode=val model=
{HOME}/runs/detect/train/weights/best.pt
data=/content/datasets/idCard-detection-3/data.yaml
%cd {HOME}
!yolo task=detect mode=predict model=
{HOME}/runs/detect/train/weights/best.pt conf=0.25
source=/content/test2.jpg save=true
import glob
from IPython.display import Image, display
for image_path in
glob.glob(f'{HOME}/runs/detect/predict/*.jpg')[:3]:
display(Image(filename=image_path, height=600))
print("\n")
model = YOLO(f'{HOME}/yolov8s.pt')
model.train(data='/content/datasets/id-Card-detection3/data.yaml'
, epochs=10,conf=0.25)results =
model.predict(source='/content/test_img2.png'
, conf=0.25)
results[0].boxes.xyxy
box = results[0].boxes.xyxy
import cv2
```

```
# Load the image
image = cv2.imread("/content/test_img2.png")
roi = image[y1:y2, x1:x2]
roi.size
cv2.imwrite('output_image.jpg'
, roi)
!sudo apt install tesseract-ocr
!pip install pytesseract
from numpy.lib.type_check import imag
from PIL import Image
import pytesseract
# Load the image
img = cv2.imread('/content/output_image.jpg')
height, width, channels = img.shape
# image=img
image = img[height//2:, :]
# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Perform thresholding to obtain a binary image
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV
+ cv2.THRESH_OTSU)
[1]
# Apply some image processing to remove noise
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel,
iterations=1)
# Perform OCR using PyTesseract
text = pytesseract.image_to_string(thresh)
```

```
# Display the extracted text
print(text)
```