Project Report

On

Vehicle Entry-Exit Management System Using Automatic Number Plate Recognition

Submitted

In partial fulfilment

For the award of the Degree of

PG Diploma in Artificial Intelligence

(C-DAC, ACTS (Pune))

Guided By:

Mr Manish Kumar Gupta

Join Director (R&D) CDAC-ACTS

Submitted By:

Basant Singh Bhaskar (220340128003)

Kapil Krishna Warke (22034012856)

Manish Nivrutti Deore (220340128006)

Vikas Pandey (22034012852)

Yogesh Gangaram Patil (22034012833)

Centre for Development of Advanced Computing(C-DAC), ACTS (Pune- 411008)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, **Mr Manish Kumar Gupta** C-DAC ACTS, Pune for his constant guidance and helpful suggestion for preparing this project **Vehicle Entry-Exit Management System Using Automatic Number Plate Recognition.**

We express our deep gratitude towards him for his inspiration, support, personal involvement, and constructive criticism that he provided us along with technical guidance during this project.

We take this opportunity to thank Program head **Mrs Risha P R** and the Head of the department **Mr Gaur Sunder** for providing us with such a great infrastructure and environment for our overall development.

We express sincere thanks to **Mrs Namrata Ailawar**, Process Owner, for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude toward **Dr Priyanka Ranade (**Course Coordinator, PG-DAI) for their valuable guidance and constant support throughout this work and helps to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS Pune**, which provided us with this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Abstract

Automatic Number Plate Recognition is an application of Machine Vision and a Database to detect and read vehicle License plates to Monitor the Entry-Exit of Authorized vehicles on private or public Premises.

The project aims to implement a Convoluted Neural Network (CNN) based Object detection model and LSTM-based Optical Character Recognition (OCR) model to build a solution by merging it with a database to manage an authorization-based controlled entry-exit system. The solution Proposed Detects and reads the number plate of a vehicle and checks for authorization based on the vehicle database of the organization. The model utilizes transfer learning using the YoloV5 Algorithm for Object detection followed by a series of Image Pre-Processing operations to prepare the output for OCR (Optical Character Recognition) Operation using Paddle OCR. The extracted text is then used for cross-referencing the vehicle authorization.

The proposed system is found to be successful and achieves results with a validation accuracy of _97.94_%.

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

1.1 Introduction

1.1.1 Automatic Number Plate Reader(s)

In recent times with the increase in technology, and the economy there is a significant increase in traffic as well, and with that, it has been more and more difficult for our authorities to keep a check on improper traffic activities which has led to a need of ANPR (Automatic Number Plate Recognition) systems. In India, there is one death every four minutes with most of them occurring due to overspeeding. ANPR is used to monitor the vehicles' average speed and can identify the vehicles that exceed the speed limit. The main use of ANPR is in highway monitoring, parking management, and neighbourhood law enforcement security.

At present times such a system is already in place in most of the Metro cities and on National Highway. We have tried to exploit one of the applications in Entry Exit Management on private premises. ANPR and such systems are the product of a field of Machine Vision. It is a process of enabling machines to have a human-like visual perspective. Some of the major applications of such systems are Absence/presence detection, Automated vision testing and measurement, Color verification, Defect detection, Optical character recognition and verification, Pattern matching, Vision guided robots and many more. All these are achieved in some basic concepts of Machine Learning by implementing Image Processing and feature mapping.

There are a lot of ways to achieve what’s needed, from using simple Machine Learning Algorithms to Complex Neural Network Frameworks and Models such as CNN, OpenCV, Yolo Model, Detectron etc.

The Basic Workflow of the Application is

1. Object Detection.
2. Image processing on the detected object (Number Plate).
3. Text Extraction using OCR.

1.1.2 Object Detection

Object detection is an application of computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, cars or anything that can be visually distinguished) in digital images and videos.

It refers to the process of discovering the location of objects in images. To do this, computer vision algorithms must first be able to recognize objects. Object recognition is a foundational subfield within computer vision that allows the software to identify specific features based on their visual attributes, such as proportion, position, and appearance. Once objects have been identified, object detection can find and localize these targets in images. This article will explore what object detection is and how it differs from image recognition.

The basic flow of process in any Object detection consists of the following steps:

1. Image preprocessing

Introduction

2. Thresholding
3. Feature identification by training
4. Feature mapping in the target image
5. Classification with confidence values

In Present times Usually, Object Detection models have 2 parts.

1. Pre-trained model on ImageNet
   For the detectors running on GPU, their backbone can be VGG, ResNet, ResNeXt, or DenseNet
   For the detectors running on the CPU, their backbone can be SqueezeNet, MobileNet, or ShuffleNet
2. A head is used to predict classes and bounding boxes. Which are of 2 types, and may also be anchor-free
   a. One-Stage object detectors
      YOLO, SSD and RetinaNet
   b. Two-Stage object detectors
      R-CNN series like fast R-CNN, faster R-CNN, R-FCN and Libra R-CNN

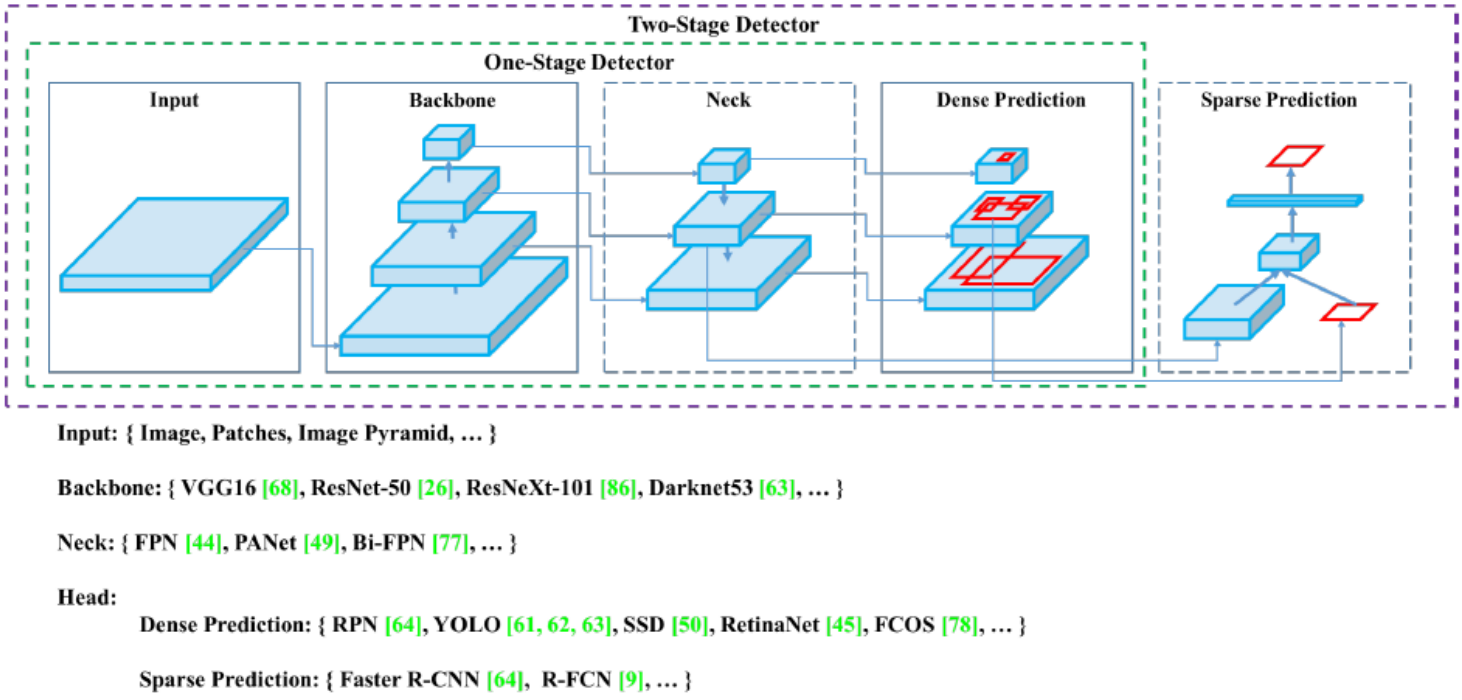

Figure 1. Modern-day, Object Detector Architecture

One of the good examples of such an Algorithm is Yolo by Ultralytics. [refer in detail]

1.1.3 OCR (Optical Character Recognition)

OCR is the use of technology to distinguish printed or handwritten text characters inside digital images of physical documents, such as scanned paper documents. The basic process of OCR involves examining the text of a document and translating the characters into code that can be used for data processing. OCR is sometimes also referred to as text recognition.

Introduction

The first step of OCR is to acquire digital images of the subject of concern. Once that is taken care of images are converted to two binary images also called black and white. The image or bitmap is analyzed for light and dark areas, where the dark areas are classified as characters that need to be recognized and light areas are classified as background.

The dark areas are then processed further to find alphabetic letters or numeric digits. OCR programs can vary in their techniques, but typically involve targeting one character, word or block of text at a time. Characters are then identified using one of two algorithms:

1. Pattern recognition- OCR program is fed examples of text in various fonts and formats which are then used to compare, and recognize, characters in the scanned document.
2. Feature detection- OCR program applies rules regarding the features of a specific letter or number to recognize characters in the scanned document. Features could include the number of angled lines, crossed lines or curves in a character for comparison. For example, the capital letter "A" may be stored as two diagonal lines that meet with a horizontal line across the middle.

When a character is identified, it is converted into an ASCII code that can be used by computer systems to handle further manipulations. Users should correct basic errors, proofread and make sure complex layouts were handled properly before saving the document for future use.

One of the good examples of such an Algorithm is Paddle OCR. [refer in detail]

1.1.4 Web App (Web Application)

A Web application (Web app) is an application program that is stored on a remote server and delivered over the Internet through a browser interface. Web applications can be designed for a wide variety of uses and can be used by anyone; from an organization to an individual for numerous reasons. Common examples of Web applications are webmail, online calculators, and e-commerce shops. Some Web apps can be only accessed by a specific browser; however, most are available no matter the browser.

Web applications do not need to be downloaded since they are accessed through a network. Users can access a Web application through a web browser. For a web app to operate, it has some dependency

1. Web server, → manages the requests that come from a client (Browser)
2. Application server, and → completes the requested task
3. A database. → used to store any needed information.

Some Major benefits that Web apps provide over old-school offline apps are:

1. Allow multiple users access to the latest version of an application at the same time.
2. Web apps don't need to be installed.
3. Web apps can be accessed through various platforms such as a desktop, laptop, or mobile.
4. Can be accessed through multiple browsers.

One Good Library in Python for linking the Backend of the web app to the Frontend is Flask.

1.1.5 SQL (Structured Query Language)

SQL is a language designed for interacting with Relational Database Management System (RDBMS). RDBMS is a software or service used to create and manage databases based on a relational model. There are a ton of RDBMS in

Introduction

the current market some of them are: MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server. Common to all these support Core SQL Commands to interact with the API.

A database is simply a collection of structured data. It is a place in which data is stored and organized. The word "relational" means that the data stored in the dataset is organized as tables. Every table relates in some way.

Client Server Model: Computers that install and run RDBMS software are called clients. Whenever they need to access data, they connect to the RDBMS server. That's the "client-server" part.

An RDBMS is built using Object which are:

1. User
2. Schema
    a. Table
        i. Rows
        ii. Columns
    b. View → Derived Tables or "Virtual Tables"

User
Users are a means of providing security at the schema level. Each schema has explicit user(s) associated with it, one of which must own the schema. The schema owner has full access to the schema and determines the access privileges of the other users.

Schema
A schema is a logical grouping of tables, indexes, triggers, routines, and other data objects under one qualifying name. Internationalization characteristics and user-level security can also be defined for schema objects.

Views
View provides an alternative way to look at the data of one or more tables. Essentially, a view is a set of a stored SELECT statements, of which you can retrieve the results at a later time by querying the view as though it were a table.

Table
A table comprises several column objects and contains rows of data. A row is a nonempty sequence of values that correspond to the column objects in the table. Every row of the same table has the same number of columns and contains a value for every column of that table.

One good example of RDBMS is MySQL, it is an open-source relational database management system (RDBMS) with a client-server model.

MySQL and SQL are not the same, SQL is the domain-specific language used for interaction between Client and Server and MySQL is the Brand.

SQL contains the following four components in its process:

1. Query Dispatcher.
2. Optimization Engines.
3. Classic Query Engine.
4. SQL Query Engine.

1.2 Objective and Specification

1.2.1 Objective

The Objective of the proposed project is to build a system that implements a CNN-based Object Detection Algorithm in sync with an OCR to build a utility to Check, Manage and enable Control over Authorized and unauthorized vehicle entrance on-premises by utilizing Resources such as Database and deploy the application using a web app.

1.2.2 Specification

The Proposed Project uses YOLO V5 as an Object Detection algorithm along with a Paddle OCR to extract vehicle registration numbers from the image. The Models are integrated using the OpenCV library to handle the interaction between the two Algorithms and for intermediate Digital Image Processing. The Database of the authorized Vehicle is maintained using MySQL RDBMS Platform and the SQL Query Engine is interacted using SQL commands in a Python environment. The Front End of the web App is made using a combination of HTML, CSS and Java which is linked with the backend using the FLASK library in python.

Chapter 2: Literature Review and Techniques

2.1 Literature Survey

2.1.1 Object Detection (YOLO)

The Process of Classification and estimating precise the concept and location of the Object in a Digital Image is called Object Detection. The Object detection has 2 sub-tasks

1. Object Localization → locate coordinates of the Object of Interest
2. Object Classification → Classifying the Object of Interest and distinguishing it in case of a multi-class problem.

To accomplish this the pipeline of an Object detection model is divided into three stages

1. Informative Region Selection
2. Feature Extraction
3. Classification

Informative Region Selection

Identification of Region of Interest or potential object classifications in an image employing aspect ratio and edge detection by convolution with multi-shaped kernels.

Feature Extraction

Feature identification of an object by analyzing histogram, high-frequency region and other methods. Practically breaking down a potential Object classifier to a set of features for better accuracy to overcome challenges such as diversity of appearance, illumination condition and background.

Classification

Literature Review and Techniques

Distinguishing of Potential Object of interest for target interest. Based on ML or Deep learning models.

Some fields of Object detection are given below.

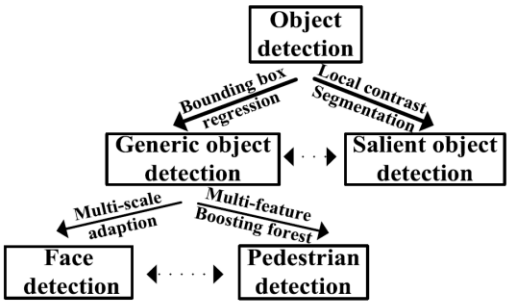

Figure 2. Application Domains of Object Detection

2.1.1.1 YOLO V1

YOLO stands for You Only Look Once; it is considered to be a revolutionary development in the field of object detection. When it was introduced, it proposed an execution speed rate of 45 fps for heavy models and as fast as 155 fps for a fast model. The Main cause behind its speed was to tackle the Object detection operation as a single regression problem. Secondarily it was to eliminate the need for multiple convolutions of the input image for individual class detection. The most dominant achievement of YOLO was to detect multiple Classes in the image in a single processing cycle.

Yolo V1 Working is a Three-stage process:

1. The model takes the image under operation and converts it into a 448x448 size
2. Runs a single convolution network on the image
3. Thresholds the resulting detection by models' confidence.

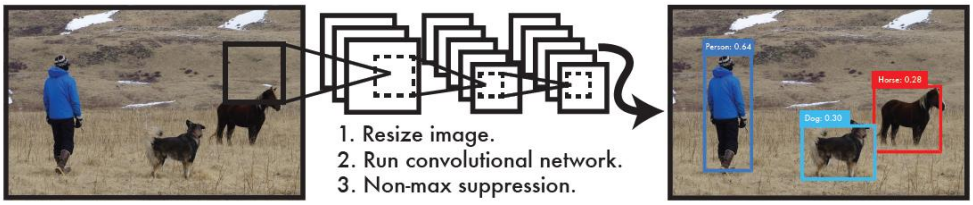

Figure 3. The YOLO Detection System (1) resizes the input image to 448 x 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Unified Detection:

The Image is divided into a grid of *S x S* squares, if the center of the Object matches a grid cell then that cell is responsible for the prediction of the Object.

Each grid cell predicts *B* bounding boxes and confidence score, if no object exists in that cell the confidence should be zero. Formally the equation of confidence is

Literature Review and Techniques

$$Pr(Object) \times IOU_{pred}^{Truth}$$

Where IOU: Integration over Union between Predicted box and Ground Truth

Each Bounding Box consists of 5 predictions: (x, y, w, h, Confidence) were

(x, y) →center of the bounding box

w → width of the bounding box

h → height of bounding box

Each grid cell predicts C Conditional class probability: $Pr(Class_i|Object)$ these are conditioned on grid cell containing Object. Only one set of class probability per grid cell is predicted due to the overlapping nature of objects on the image.

At test time conditional class probabilities and the individual box confidence prediction are multiplied to calculate a class-specific confidence score.

$$Pr(Class_i|Object) \times Pr(Object) \times IOU_{pred}^{Truth} = Pr(Class_i) \times IOU_{pred}^{Truth}$$

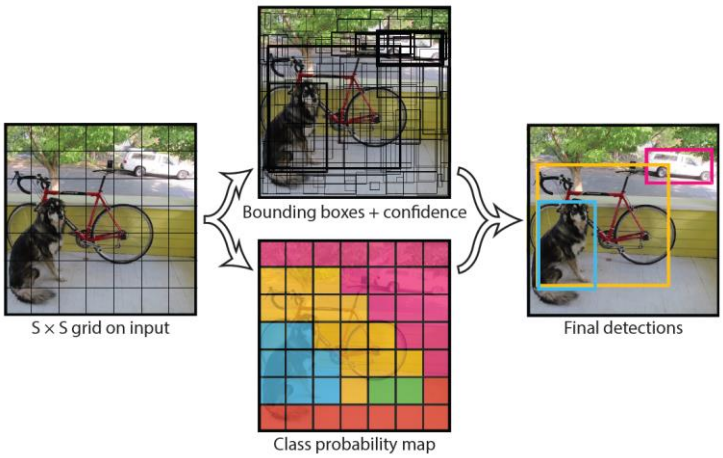

Figure 4. Unified Detection: It divides the image into an *S x S* grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an *S x S x (B * 5 + C)* tensor.

Neural Network Design

The Neural Network Used is a CNN with initial layers used for extracting the Features from the image and fully connected layers predicting the output probabilities and coordinates. The CNN of Yolo is inspired by the GoogleNet model for image classification.

Base YOLO CNN has 24 Convoluted Layers followed by 2 fully connected layers, then a 1 x 1 reduction layer followed by a 3x3 convoluted layer. It has leaky ReLU in all hidden layers and Linear activation at the output layer with SSE at metrics.

Literature Review and Techniques

Fast YOLO CNN is the same as base YOLO but has 9 Convoluted Layers instead of 24.

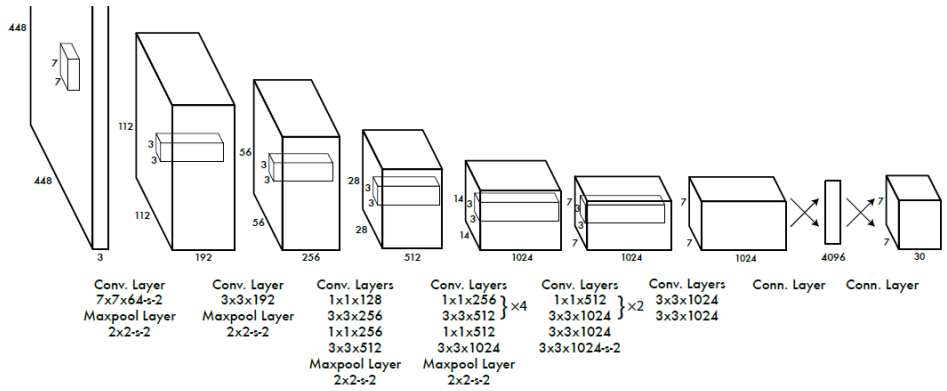

Figure 5. Base YOLO Architecture: 24 convolutional layers followed by 2 fully connected layers. Alternating the 1 x 1 convolutional layer reduces the feature space from the preceding layers.

Benefits of YOLO v1

1. Extremely Fast as YOLO takes the task as a regression problem which drops the need for complex pipelines which in return makes the processes with very low latency of 25 milliseconds with double precision compared to real-time systems of that time.
2. Yolo Process image in a global perspective when making predictions and while training as well which enables it to better distinguish background in the image which is a major challenge for other sliding window or region-based algorithms.
3. Learning in YOLO happens on a generalizable representation of Objects, which even enables YOLO to Outperform non-natural data such as paintings and artworks.

Limitation of YOLO v1

1. YOLO Lags in terms of Accuracy in comparison to the state-of-the-art detection system of that time. And struggle in precisely identifying small objects.
2. It has a spatial constraint on bounding boxes and can only predict two boxes and can only have one class. Which limits the number of nearby objects that can be predicted.
3. It also struggles to detect objects with unfamiliar aspect ratios.

2.1.1.2 YOLO V2

YOLO V2 also known as YOLO 9000 is a successor of YOLO v1 with a capacity of identifying 9000 Object Categories with a higher precision score mAP (Mean Average Precision) and a better trade-off between speed and accuracy while still being in real-time.

It proposes a joint training method for Object Detection and Classification. This method uses a hierarchical view of Object classification that allows for a combination of distinct datasets by use of a Labeled dataset.

Unlike v1 it also implements batch normalization eliminating the need for other normalization and dropout operations while increasing mAP by 2%.

Yolo v2 is trained with 224 x 224 resolution and then transfer learned on 448 x 448 resolution images for 10 epochs to adjust the network for High-resolution classification, which increases mAP by 4%.

Literature Review and Techniques

The image is then shrunk to 416 from 448 by downsampling by a factor of 32 to keep the image size in an odd number to have a single center cell for prediction instead of having 4 closely related cells in the center and Anchor boxes are created, these anchor boxes drop the mAP by 0.3 and increase the recall by 7%.

By Direct location prediction and using dimensional clustering with anchor boxes YOLO improves by 5% overall

In YOLO 2 the model is trained in multiple resolutions of images with a downsampling rate of 32 which provides a scope of 320 to 608 for its operations, with different processing requirements it enables implementation on small GPU and different frame rates as shown below.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Table 1. Detection frameworks on PASCAL VOC 2007. YOLO v2 comparison with different image resolutions

Neural Network parameters (Darknet19)

Literature Review and Techniques

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 2. YOLO v2 Neural Net Architecture

Similar to V1 V2 as well follow hierarchical Classification but is a step ahead by forming a Word Tree by fusing COCO and ImageNet dataset similar to the WorkNet approach.

The difference or progress from YOLO V1 to V2 can be summarized in the following table.

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table 3. YOLO v1 to v2 Progress

2.1.1.3 YOLO V3

This version came out as a minor update with some improvements in the architecture, loss function and metrics. For classification, softmax is dropped and an independent logistic classifier is used and binary cross-entropy for final

predictions. V3 is stronger and performs significantly better with smaller objects but struggles a little with medium and large-scale objects.

The Neural Net Architecture is tweaked by inspiring form Darknet 19 and as it has 53 layers its called Darknet-53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 6. YOLO v3 Darknet-53

2.1.1.4 YOLO v4

The main contribution of Yolo v4 Is that it brings the possibility of training fast and accurate models in conventionally commercial GPUs.

To understand how YOLO v4 makes it possible, we need to understand the meaning of the two phrases:

1. Bag of Freebies
   For conventional GPUs, the training strategy needs to be modified to receive better accuracy without increasing inference costs. The only change in the training strategies or training cost is termed as Bag of Freebies.
2. Bag of Specials
   The method or strategy that increases the inference cost by little but impacts greatly on the performance is called Bag of Special.

Bag of Freebies

1. Data Augmentation, → The purpose of data augmentation is to increase the variability of the input images so that the designed object detection model has higher robustness to the images obtained from different environments.
   a. Pixel-wise
      i. Photometric distortion→ brightness, contrast, hue, saturation, and noise of an image is randomly altered in the training dataset.

ii. Geometric distortion→ random scaling, cropping, flipping, and rotation is performed in the training dataset.

b. Image Level
   i. Random erase & Cutout → making a section of pixel values to zero
   ii. Hide-and-Seek & Grid mask → making pixel values of multiple rectangles to zero

c. Network Level / Feature Map
   i. DropOut
   ii. DropConnect
   iii. DropBlock

d. Recreation of Data from an existing dataset
   i. MixUp→ Multiplying 2 images and superimposing with different coefficients
   ii. CutMix→ cover the cropped image to the rectangle region of other images, and adjusts the label according to the size of the mixed area.
   iii. Style transfer GAN→ To reduce texture-based bias learning by CNN

2. To Solve semantic distribution in the dataset → To correct data imbalance between different classes.
   a. Hard negative example mining or online hard example mining. → Only for 2 staged Object Detectors.
   b. Focal loss
3. One Hot hard representation→ to express the degree of Association between categories.
4. Objective Function of Bounding Box Regression → Traditionally MSE is used to perform regression on center points.

Bag of Special

Bag of Specials can be considered as an add-on for any object detectors present right now to make them more accurate on benchmark datasets.

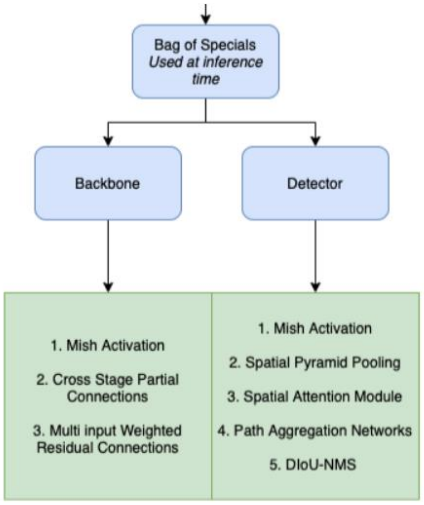

Figure 7. Different Methods in Bag of Specials

Mish Activation

Mish[2] is a novel activation function similar to Swish[3] and is defined as

$$f(x) = x \cdot tanh\big(softplus(x)\big) = x \cdot tanh(\ln(1 + e^{x}))$$

Literature Review and Techniques

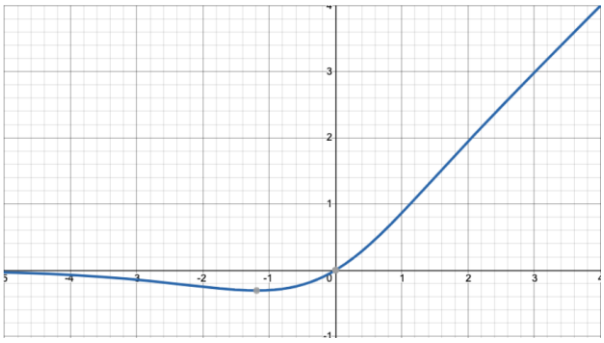

Figure 8. Mish Activation Function

Properties of Mish Activation Function are:-

- Non-monotonic function: Preserves negative values, that stabilize the network gradient flow and unlike ReLU, almost solves the Dying ReLU problem and helps to learn more expressive features.
- Unboundedness and Bounded Below: The former helps to remove the saturation problem of the output neurons and the latter helps in better regularization of networks.
- Infinite Order of Continuity: Unbiased towards initialization of weights and learning rate due to the smoothness of a function helping for better generalizations.
- High compute function but increases accuracy: Although being a high-cost function, it has proven itself better in deep layers in comparison to ReLU.
- Scalar Gating: Scalar Gating is an important property of this function and so it becomes logical and can easily replace the pointwise functions like ReLU.

CSP (Cross Stage Partial Connections)

Input to a dense block is divided into two parts. One is used directly in the concatenation with the final output of the DB+TB chain and the other is used as an input in the dense block

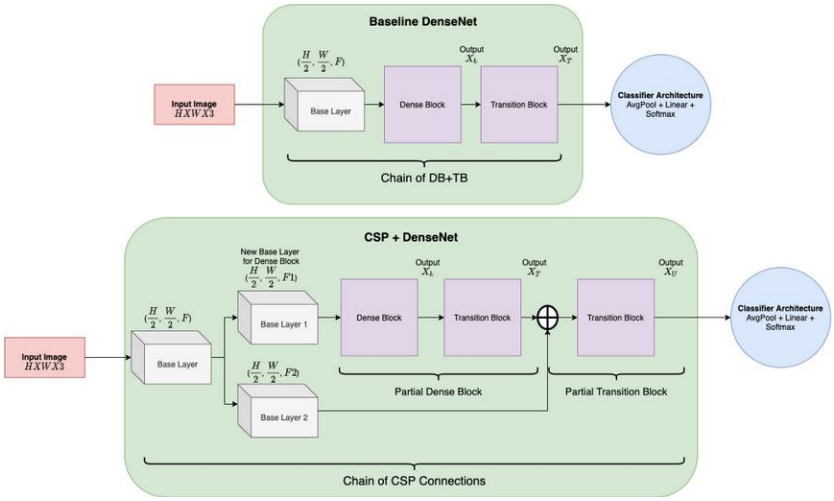

Figure 9. Baseline and CSP + DenseNet Architecture

Advantages of CSP Connections are: -

1. Increasing the gradient flow paths that help in removing the replicated updates of the weights in DenseBlock.

2. Breaking a base layer into two parts helps to decrease the number of multiplications in Dense Block, which further helps in increasing inference speed.

Selection of BoF and BoS for YOLOv4

For improving the object detection training, a CNN usually uses the following:

- **Activations:** ReLU6
- **Bounding box regression loss:** MSE, IoU, GIoU, CIoU, DIoU
- **Data augmentation:** Mosaic[1] and Self-Adversarial Training (SAT)[2]
- **Regularization method:** DropBlock
- **Normalization of the network activations by their mean and variance:** Batch Normalization (BN), Filter Response Normalization (FRN), modified SAM[4], modified PAN[5], and Cross mini-Batch Normalization (CmBN)[3]
- **Skip-connections:** Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)

[1] Mosaic Data Augmentation:

A new data augmentation technique that mixes 4 raining images to make a single image. Thus having 4 different contexts mixed, while CutMix mixes only 2 input images. This allows the detection of objects outside their normal context. Batch normalization calculates activation statistics from 4 different images on each layer eliminating the need for a larger mini-batch size.

[2] Self-Adversarial Training:

It represents a new data augmentation technique that operates in 2 forward and backwards stages.

In the 1st stage, the neural network alters the original image instead of the network weights. In this way, the neural network executes an adversarial attach on itself altering the original image to create the deception that there is no desired object in the image.

In the 2nd stage, the neural network is trained to detect an object on this modified image in the normal way.

[3] Cross mini-Batch Normalization (CmBN):

It represents a CBN-modified version, as shown below, defined as Cross mini-Batch Normalization (CmBN). This collects statistics only between mini-batches within a single batch.

Literature Review and Techniques

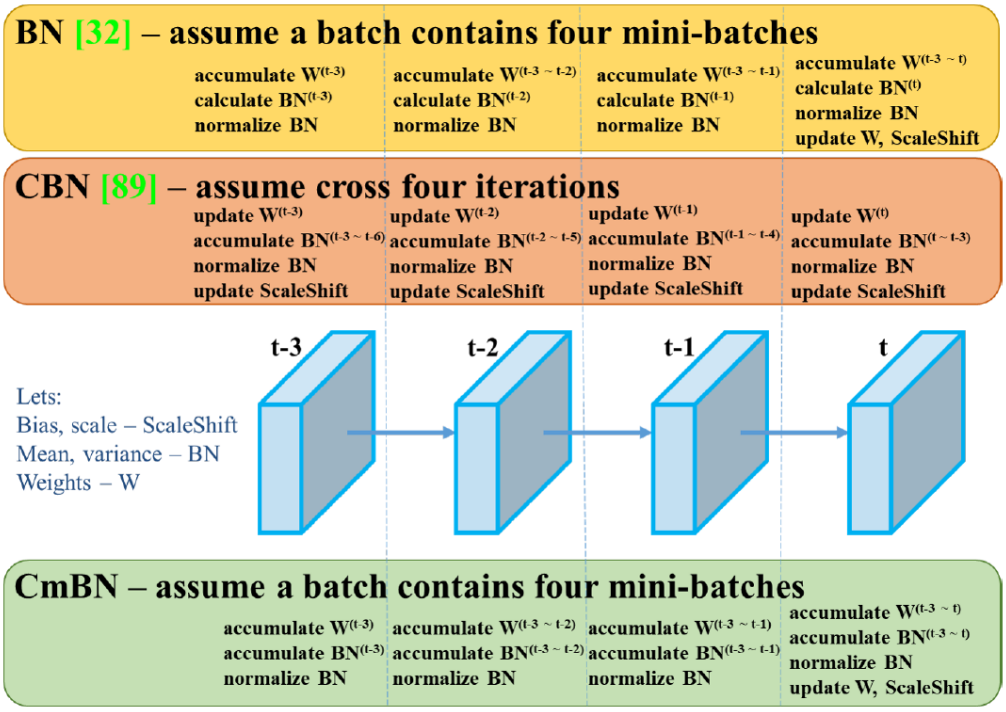

Figure 10. Cross mini-Batch Normalization(CmBN)

[4] Modified SAM and PAN:

In SAM from spatial-wise attention to pointwise attention and replace shortcut connection of PAN to concatenation as shown below.

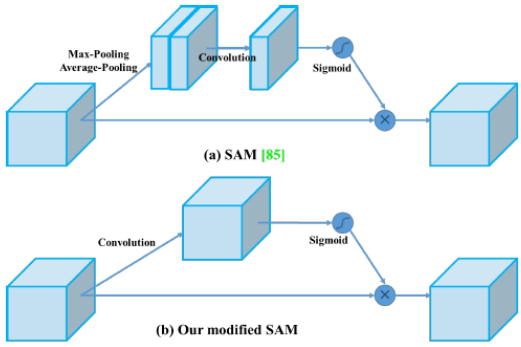

Figure 11. Modified SAM

Literature Review and Techniques

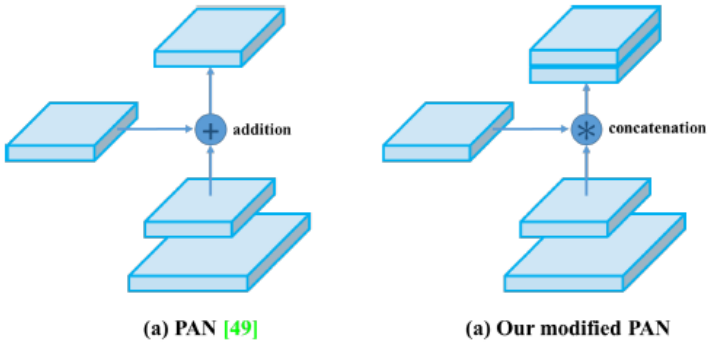

Figure 12. Modified PAN

YOLOv4 Architectural Overview

- **Backbone**: CSPDarknet53
- **Neck**: SPP [25], PAN
- **Head**: YOLOv3
- **Bag of Freebies (BoF) for backbone**: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing
- **Bag of Specials (BoS) for backbone**: Mish activation, Cross-stage partial connections (CSP), Multi input weighted residual connections (MiWRC)
- **Bag of Freebies (BoF) for detector**: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, eliminate grid sensitivity, using multiple anchors for single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes
- **Bag of Specials (BoS) for detector**: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

2.1.1.5 YOLOv5

The YOLOv5 Was released just after 2 months of release from YOLOv4

As not being published by the same author but is inspired by the original researcher's Last release YOLOv3. The Major Difference between the two is:

1. It's implemented in PyTorch.
2. The Mosaic data Argumentation and auto-learning bounding box anchor.

It has 4 different Models in the repository

YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x

The first is the smallest being 's' with the fastest but least accurate among the four. And 'x' is the slowest but most accurate among the four.

The largest version of the YOLOv5x model and the smallest YOLOv5s do not differ in layers. The difference between them as well as between other versions is in the scaling multipliers of the width and depth of the network.

Literature Review and Techniques

2.1.2 OCR (Paddle OCR)

As previously introduced, OCR is a key component of the proposed Solution.

PaddleOCR is a state-of-the-art Optical Character Recognition (OCR) model published in September 2020 and developed by Chinese company Baidu using the PaddlePaddle (Parallel Distributed Deep Learning) deep learning framework. It gives a lot of consideration to the balance between recognition accuracy and computational load for practical use. In addition to these considerations.

Paddle OCR provides multi-Language, as of now the supported language are:

Traditional Chinese, English, French, Arabic, Spanish, Portuguese, Russia, German, Korean, Japanese, Italian, Hindi, Uyghur, Persian, Urdu, Occitan, Marathi, Nepali, Serbian (Cyrillic), Serbian (Latin), Bulgarian, Ukrainian, Belarusian, Telugu, Kannada, Tamil, Mongolian, Bangla, Vietnamese, Burmese, Turkish, polish.

In terms of Applications, the following are an area of possible implementation.

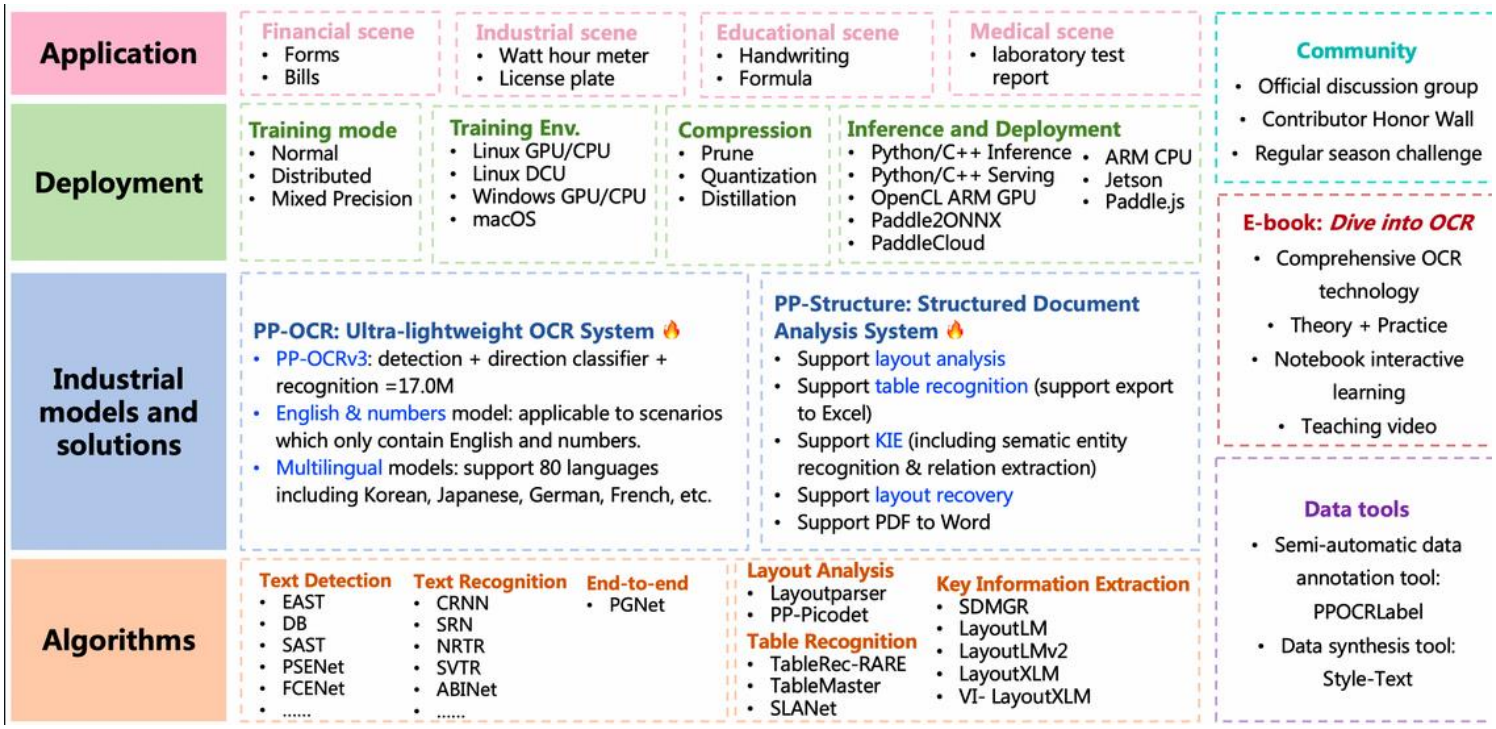

Figure 13. Paddle OCR, Application, Deployment, Solutions and Algorithms'

The Pipeline used in the Paddle OCR:

Literature Review and Techniques

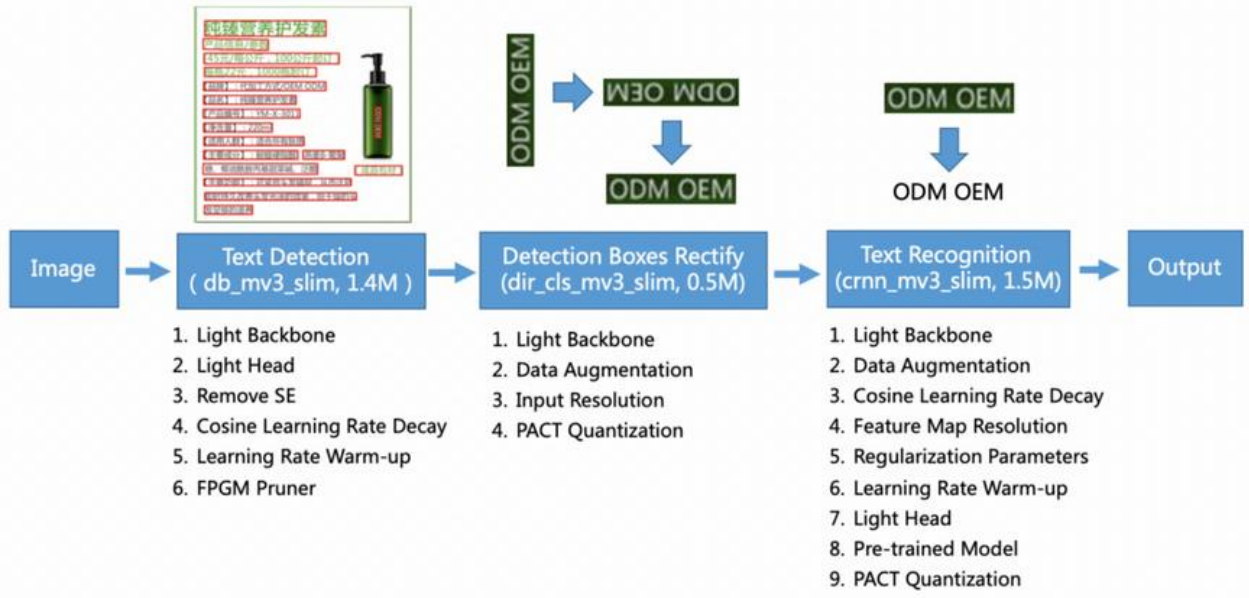

Figure 14. Paddle OCR Pipeline.

Source : https://github.com/PaddlePaddle/PaddleOCR/blob/release/2.0/doc/ppocr_framework.png

There are three major processing steps in the Paddle OCR pipeline:

1. Character position detection | Text Detection
2. Recognition of character orientation and correction | Direction Classification
3. Character content recognition | Text Recognition

It contains one model each for the above-mentioned Steps. For all Languages, the first 2 model remains Common, but for the last step, each supported language has a separate Model.

2.1.2.1 Text Detection

The purpose of text detection is to locate the text area in the image. In PP-OCR, It uses Differentiable Binarization (DB) as a text Detector which is based on a simple Segmentation network. This simple postprocessing of DB makes it very efficient.

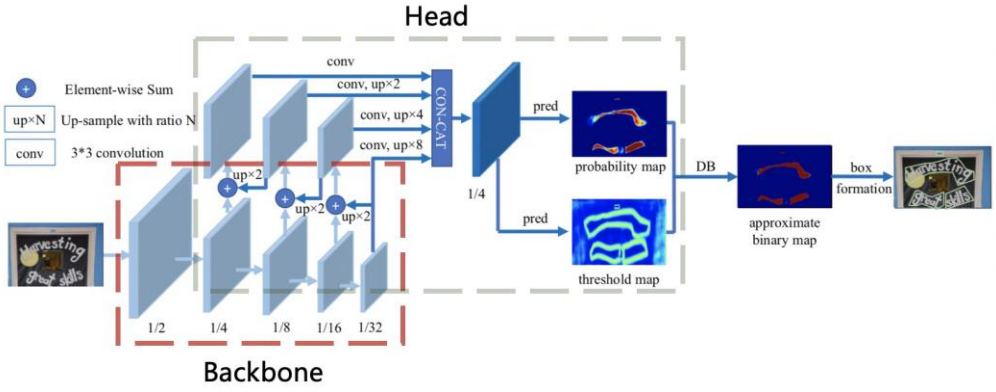

Figure 15. Paddle OCR Text Detector Architecture

To further improve its effectiveness and efficiency, the following six strategies are used:

Literature Review and Techniques

1. light backbone,
   The size of the backbone has a dominant effect on the model size of a text detector. Therefore, light backbones should be selected for building ultra-lightweight models.
   Paddle OCR uses MobileNetV3 large x0.5 to balance accuracy and efficiency empirically.

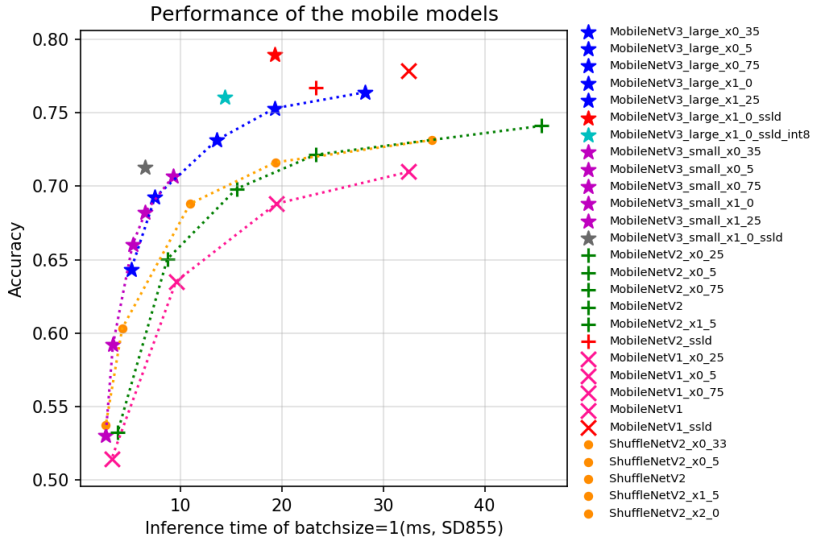

Figure 16. Paddle OCR, light backbone Performance on ImageNet 1000, MobileNetV1, V2, V3 and ShuffleNetV2 Series

2. light head,
3. remove SE (squeeze-and-excitation) module,
4. cosine learning rate decay,
5. learning rate warm-up, and
6. FPGM pruner.

2.1.2.2 Direction Classification

Before recognizing the detected text, the text box needs to be transformed into a horizontal rectangle box for subsequent text recognition, which is easy to be achieved by geometric transformation as the detection frame is composed of four points. However, the rectified boxes may be reversed. Thus, a classifier is needed to determine the text direction. If a box is determined reversed, further flipping is required. Training a text direction classifier is a simple image classification task.

Literature Review and Techniques

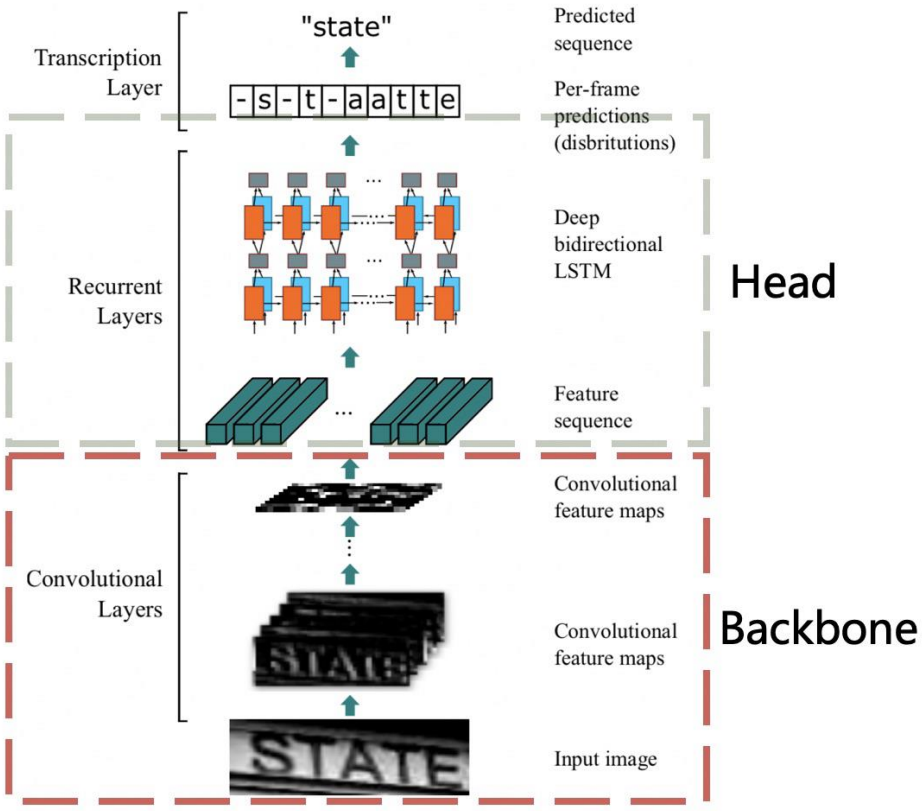

Figure 17. Paddle OCR, Text Direction Classification Model

It Adopts the following four strategies to enhance the model ability and reduce the model size.

1. Light backbone
   It also uses MobileNetV3 as the backbone for the direction classifier which is the same as the text detector. Because this task is relatively simple, it uses MobileNetV3 small x0.35 to balance accuracy and efficiency empirically.
2. Data augmentation
3. Input resolution
4. PACT quantization

2.1.2.3 Text Recognition

PP-OCR uses CRNN as a text recognizer which is widely used and practical for text recognition. CRNN integrates feature extraction and sequence modelling. It adopts the Connectionist Temporal Classification (CTC) loss to avoid the inconsistency between prediction and label. To enhance the model ability and reduce the model size of a text recognizer,

The following nine strategies are used:

1. light backbone,
   It also uses MobileNetV3 as the backbone of the text recognizer which is the same as text detection. MobileNetV3 small x0.5 is selected to balance accuracy and efficiency empirically.

Literature Review and Techniques

2. data augmentation,
3. cosine learning rate decay,
4. feature map resolution,
5. regularization parameters,
6. learning rate warm-up,
7. light head,
8. pre-trained model and
9. PACT quantization

The paddle OCR generates Two Outputs.

1. Detection Bounding Box (dt_boxes)
   It stores a list-of-list-of-list of size (No_of_detection,4,2) which has a value of the four corners of the bounding box of the detected characters, expressed in XY coordinates.
   [[675., 66.], [844., 66.], [844., 104.], [675., 104.]]
2. Recognition result (rec_res)
   It stores a list-of-list of size (No_of_detection,2) which has text recognized and confidence on a scale of 0 to 1.

2.2 Model Selection

Initially, for Model and Algorithm selection we shortlisted 3 popular Algorithms and then made a comparison among them and then finalised on grounds of being most suitable for our requirement.

2.2.1 Object Detection

For Object Detection We considered the Following Options:

1. R-CNN
2. Fast R-CNN
3. Faster R-CNN
4. R- FCN (Fully connected Neural Network)
5. Mask R-CNN
6. SSD (Single Shot Detector)
7. YOLO (You Only Look Once)

But we finalized YOLO as the algorithm to work upon as:

1. It is Fast. Almost Realtime.
   Because we regard it as a regression problem, without the need for complex pipelines, when testing, we only need to put the new image into the neural network and run it. It can achieve real-time effects and the mean average precision (mAP) is higher.
2. Have a global understanding and context of the image.
   Unlike the technology based on sliding window and region proposal, YOLO implicitly encodes the context information of the target's category and its appearance, while Fast RCNN cannot see the larger context

information. YOLO uses the characteristics of the entire image to predict each bounding box and simultaneously predicts all categories of bounding boxes in the image, that is, it has a global understanding of the image.

3. The learning of goals is more general.
4. High accuracy and can quickly locate the target and position in the image, even small targets.
5. The design of the grid unit can alleviate the problem of detecting the same object multiple times.
6. Provides fewer bounding boxes, only 98 images per image, and about 2000 selective searches.
7. Combine these separate components into a single joint optimization model.
8. YOLO is a more versatile detector that can detect multiple objects simultaneously

2.2.2 OCR

We have used Paddle OCR for our Project.

Paddle ocr pros over tesseract

1. If the text is rotated in non-90-degree rotations, PaddleOCR can still detect some text correctly, but Tesseract cannot do this even if OSD is used.
2. You can use the detection results to fix the rotation, but Tesseract is likely to retrieve non-sense results.
3. PaddleOCR works better than Tesseract when images are in RGB/BGR if you can't binarize your image.
4. Paddle ocr works faster on GPU by 46% than tesseract on standard GPU.
5. With preprocessing of an input image paddleOCR gives higher accuracy compare to tesseract
6. Paddle ocr is built on python language.

Tesseract ocr pros over paddle

1. Tesseract is better in terms of processing scanned documents.
2. Image segmentation modes are to the rescue and help a lot with improving the results.
3. Tesseract results on binarized images with long text are usually better than PaddleOCR.
4. Tesseract is far better at detecting symbols.
5. Tesseract is faster on CPU
6. Tesseract is built in c++ language.

Literature Review and Techniques

2.3 Process Flow

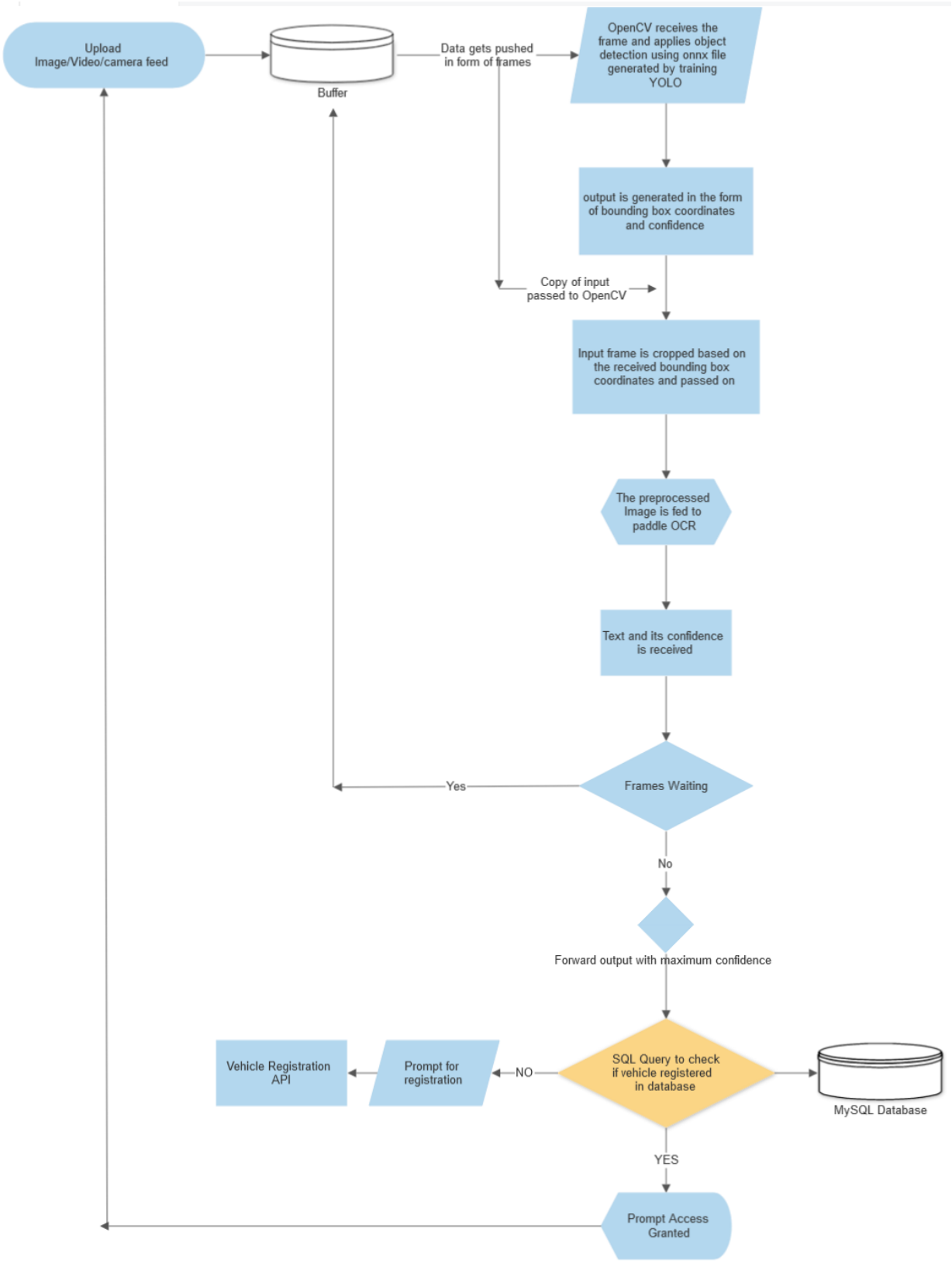

Figure 18. Process Flow

Chapter 3: Implementation

3.1 Dataset Preparation

3.1.1 Data Definition.

For Yolo Dataset needs to be in a specific format. And this preparation is one of the most important steps.

Yolo Takes the dataset separate for the train and test labelled folder. Example

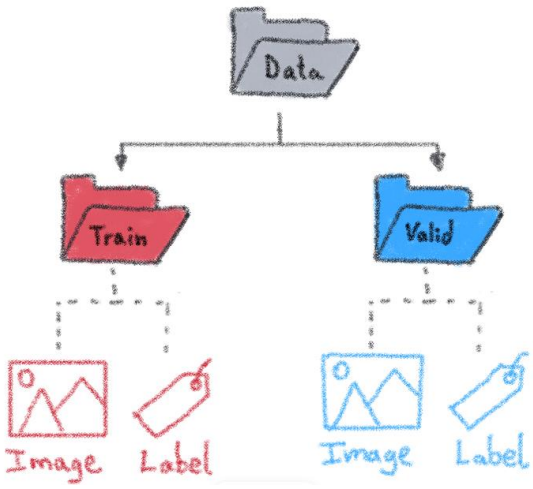

Figure 19. Dataset segregation

3.1.2 Label list File

YOLO requires every image with a txt file sharing the same file name but differentiated by file type/ extension.

For example, if there is an image with the file name “sample_1.jpg” then it requires “sample_1.txt”

The label file is a “txt” file.

Which has the following data in normalized values wrt

1. Coordinate the centre of the Bounding box.
2. Height of Bounding box.
3. Width of the Bounding box.
4. Class id of the object in the bounding box.

Example:

```
(class_id, x_centre,  y_centre,  width,  height)
```

3.1.3 Tools for Data

There are multiple tools available for label creation some are:

1. MakeSense.AI

Implementation

2. LabelImg
3. VGG image annotator
4. LabelMe
5. Scalable
6. RectLabel

We have used LabelImg for our project, The installation and Use can be learned using the below link

https://github.com/heartexlabs/labelImg

3.2 Environment Setup

3.2.1 Base Environment

For the complete Project environment, we are using python version 3.9

3.2.1.1 Train and Runtime environment setup

Ensure that Python 3.9 with virtualenv is installed in your Windows / Linux system. Procedures can be found online on multiple forums any of them will suffice.

We recommend making different environments for training and runtime as both of them have independent requirements. For that use the following command

1. *"python -m venv training_env runtime_env"* → to create virtual env
2. *"dir"* → to see the new contents of the working directory.

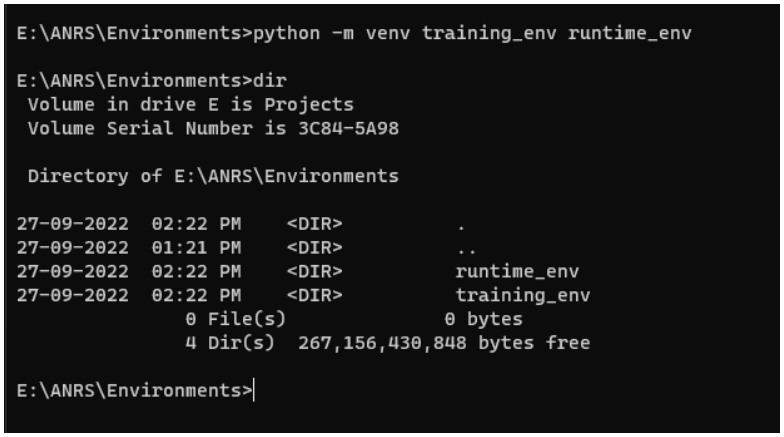

Figure 20. creating python virtual environment for training and runtime

3.2.2 For Darknet

YOLO V5 is Built on PyTorch and thus requires a specific setup for its training. And thus we need to set up python with or without Cuda as per system configuration.

3.2.2.1 Activate Training environment

We need to activate the newly created training environment,

For that run the activate.bat file in the Scrips directory of training_env

Implementation

1. "*training_env\Scripts\activate.bat*" → To run activate the training virtual env, the output can be seen below figure

```
E:\ANRS\Environments>training_env\Scripts\activate.bat

(training_env) E:\ANRS\Environments>
```

Figure 21. activate training environment

3.2.2.2 Install PyTorch

The command to install PyTorch can be generated at the PyTorch website by selecting the configurations we have uses below configuration

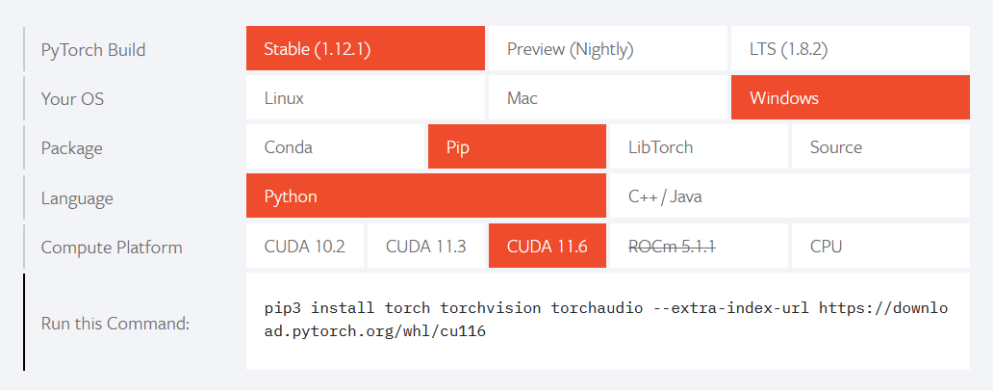

Figure 22. The PyTorch installation command generation

Run the following command

"*pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu116*"

```
E:\ANRS\Environments>training_env\Scripts\activate.bat

(training_env) E:\ANRS\Environments>pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu116
Looking in indexes: https://pypi.org/simple, https://download.pytorch.org/whl/cu116
Collecting torch
  Using cached https://download.pytorch.org/whl/cu116/torch-1.12.1%2Bcu116-cp39-cp39-win_amd64.whl (2388.0 MB)
Collecting torchvision
  Using cached https://download.pytorch.org/whl/cu116/torchvision-0.13.1%2Bcu116-cp39-cp39-win_amd64.whl (2.6 MB)
Collecting torchaudio
  Using cached https://download.pytorch.org/whl/cu116/torchaudio-0.12.1%2Bcu116-cp39-cp39-win_amd64.whl (1.2 MB)
Collecting typing-extensions
  Using cached typing_extensions-4.3.0-py3-none-any.whl (25 kB)
Collecting pillow!=8.3.*,>=5.3.0
  Using cached Pillow-9.2.0-cp39-cp39-win_amd64.whl (3.3 MB)
Collecting numpy
  Using cached numpy-1.23.3-cp39-cp39-win_amd64.whl (14.7 MB)
Collecting requests
  Using cached requests-2.28.1-py3-none-any.whl (62 kB)
Collecting charset-normalizer<3,>=2
  Using cached charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting idna<4,>=2.5
  Using cached idna-3.4-py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2022.9.24-py3-none-any.whl (161 kB)
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
Installing collected packages: urllib3, typing-extensions, pillow, numpy, idna, charset-normalizer, certifi, torch, requests, torchvision, torchaudio
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 idna-3.4 numpy-1.23.3 pillow-9.2.0 requests-2.28.1 torch-1.12.1+cu116 torchaudio-0.12.1+cu116 torchvision-0.13.1+cu116 typing-extensions-4.3.0
urllib3-1.26.12
WARNING: You are using pip version 22.0.4; however, version 22.2.2 is available.
You should consider upgrading via the 'E:\ANRS\Environments\training_env\Scripts\python.exe -m pip install --upgrade pip' command.

(training_env) E:\ANRS\Environments>
```

Figure 23. PyTorch Installation with CUDA

Implementation

3.2.2.3 Clone Git Repo of YOLOv5

This can be in two ways

1. If you have Git installed in your system use the following command
   "*git clone* https://github.com/ultralytics/yolov5.git"
   Run the "dir" command to check if the repo is cloned to the working directory.

```
(training_env) E:\ANRS\Environments>git clone https://github.com/ultralytics/yolov5.git
Cloning into 'yolov5'...
remote: Enumerating objects: 12844, done.
remote: Total 12844 (delta 0), reused 0 (delta 0), pack-reused 12844
Receiving objects: 100% (12844/12844), 13.02 MiB | 1.32 MiB/s, done.
Resolving deltas: 100% (8830/8830), done.

(training_env) E:\ANRS\Environments>dir
 Volume in drive E is Projects
 Volume Serial Number is 3C84-5A98

 Directory of E:\ANRS\Environments

27-09-2022  02:52 PM    <DIR>          .
27-09-2022  01:21 PM    <DIR>          ..
27-09-2022  02:22 PM    <DIR>          runtime_env
27-09-2022  02:22 PM    <DIR>          training_env
27-09-2022  02:52 PM    <DIR>          yolov5
               0 File(s)              0 bytes
               5 Dir(s)  262,513,266,688 bytes free

(training_env) E:\ANRS\Environments>
```

Figure 24. Cloning YOLOv5 using the git clone command

2. Or you can directly download the repo from git-hub at https://github.com/ultralytics/yolov5
   After opening the link click on the green button with code written in it, and you will see a Download option in the newly opened window. Save and extracted that folder in your working directory.

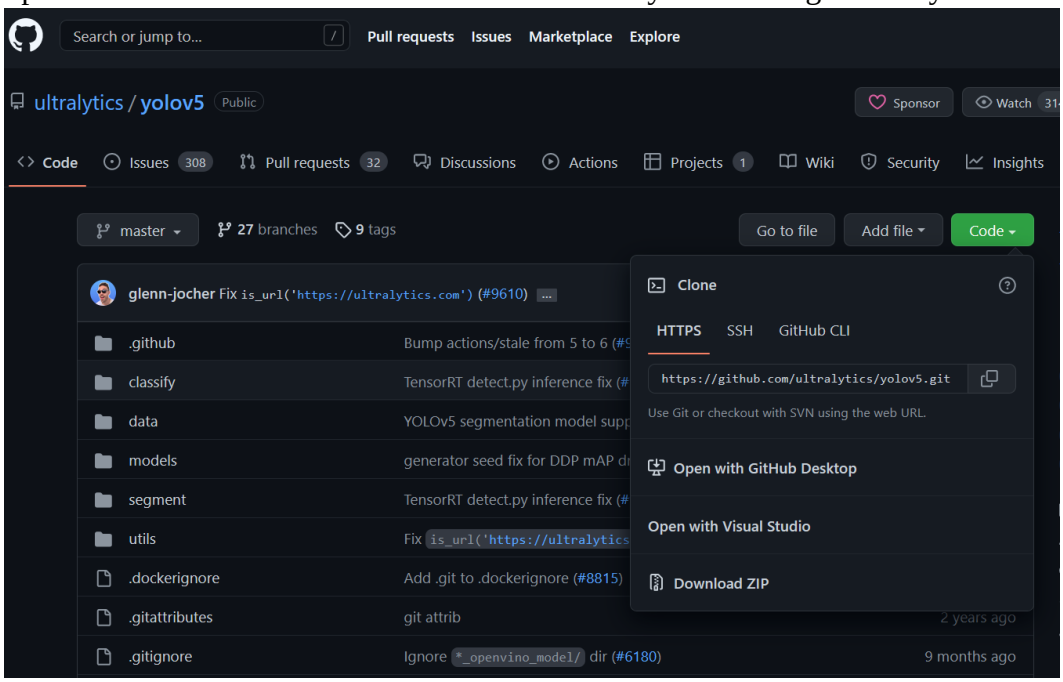

Figure 25. download the yolov5 repo from the browser

Implementation

3.2.2.4 Install darknet

To install the required dependencies run the following command

"*pip install -r yolov5\requirements.txt onnx onnx-simplifier*"

```
(training_env) E:\ANRS\Environments>pip install -r yolov5\requirements.txt onnx onnx-simplifier
Collecting onnx
  Using cached onnx-1.12.0-cp39-cp39-win_amd64.whl (11.5 MB)
Collecting onnx-simplifier
  Using cached onnx_simplifier-0.4.8-cp39-cp39-win_amd64.whl (1.2 MB)
Collecting matplotlib>=3.2.2
  Using cached matplotlib-3.6.0-cp39-cp39-win_amd64.whl (7.2 MB)
Requirement already satisfied: numpy>=1.18.5 in e:\anrs\environments\training_env\lib\site-packages (from -r yolov5\requ
irements.txt (line 6)) (1.23.3)
Collecting opencv-python>=4.1.1
  Using cached opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
```

Figure 26. YOLOv5 dependency installation

```
Installing collected packages: wcwidth, tensorboard-plugin-wit, pytz, pyasn1, pure-eval, pickleshare, executing, commonm
ark, backcall, zipp, wheel, traitlets, tensorboard-data-server, six, scipy, rsa, PyYAML, pyparsing, pygments, pyasn1-mod
ules, psutil, protobuf, prompt-toolkit, parso, opencv-python, oauthlib, MarkupSafe, kiwisolver, fonttools, decorator, cy
cler, contourpy, colorama, cachetools, absl-py, werkzeug, tqdm, thop, rich, requests-oauthlib, python-dateutil, packagin
g, onnx, matplotlib-inline, jedi, importlib-metadata, grpcio, google-auth, asttokens, stack-data, pandas, onnx-simplifie
r, matplotlib, markdown, google-auth-oauthlib, tensorboard, seaborn, ipython
Successfully installed MarkupSafe-2.1.1 PyYAML-6.0 absl-py-1.2.0 asttokens-2.0.8 backcall-0.2.0 cachetools-5.2.0 coloram
a-0.4.5 commonmark-0.9.1 contourpy-1.0.5 cycler-0.11.0 decorator-5.1.1 executing-1.1.0 fonttools-4.37.3 google-auth-2.11
.1 google-auth-oauthlib-0.4.6 grpcio-1.49.1 importlib-metadata-4.12.0 ipython-8.5.0 jedi-0.18.1 kiwisolver-1.4.4 markdow
n-3.4.1 matplotlib-3.6.0 matplotlib-inline-0.1.6 oauthlib-3.2.1 onnx-1.12.0 onnx-simplifier-0.4.8 opencv-python-4.6.0.66
 packaging-21.3 pandas-1.5.0 parso-0.8.3 pickleshare-0.7.5 prompt-toolkit-3.0.31 protobuf-3.19.5 psutil-5.9.2 pure-eval-
0.2.2 pyasn1-0.4.8 pyasn1-modules-0.2.8 pygments-2.13.0 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.2.1 requests-oau
thlib-1.3.1 rich-12.5.1 rsa-4.9 scipy-1.9.1 seaborn-0.12.0 six-1.16.0 stack-data-0.5.1 tensorboard-2.10.1 tensorboard-da
ta-server-0.6.1 tensorboard-plugin-wit-1.8.1 thop-0.1.1.post2209072238 tqdm-4.64.1 traitlets-5.4.0 wcwidth-0.2.5 werkzeu
g-2.2.2 wheel-0.37.1 zipp-3.8.1

(training_env) E:\ANRS\Environments>
```

Figure 27. YOLOv5 Dependency installation confirmation

We have completed setting up the darknet for training.

3.2.3 For paddle OCR

3.2.3.1 Activate Runtime environment

We need to activate the newly created training environment,

For that run the activate.bat file in the Scrips directory of training_env

1. "*runtime_env\Scripts\activate.bat*" → To run activate the training virtual env, the output can be seen below figure

```
E:\ANRS\Environments>runtime_env\Scripts\activate.bat

(runtime_env) E:\ANRS\Environments>
```

Figure 28. Activate runtime environment

Implementation

3.2.3.2 Install paddlepaddle and paddleocr

To install paddlepaddle and paddleocr run "*pip install paddlepaddle paddleocr"*

```
(runtime_env) E:\ANRS\Environments>pip install paddlepaddle paddleocr
Collecting paddlepaddle
  Using cached paddlepaddle-2.3.2-cp39-cp39-win_amd64.whl (64.3 MB)
Collecting paddleocr
  Using cached paddleocr-2.6.0.1-py3-none-any.whl (389 kB)
Collecting requests>=2.20.0
  Using cached requests-2.28.1-py3-none-any.whl (62 kB)
Collecting astor
  Using cached astor-0.8.1-py2.py3-none-any.whl (27 kB)
Collecting Pillow
```

Figure 29. paddle OCR installation

```
=3.6.0->flask>=1.1.1->visualdl->paddleocr) (3.8.1)
Collecting MarkupSafe>=2.0
  Using cached MarkupSafe-2.1.1-cp39-cp39-win_amd64.whl (17 kB)
Using legacy 'setup.py install' for future, since package 'wheel' is not installed.
Installing collected packages: python-dateutil, pyparsing, pycryptodome, protobuf, Pillow, numpy, networkx, MarkupSafe,
lxml, kiwisolver, jarowinkler, itsdangerous, importlib-metadata, idna, future, fonttools, et-xmlfile, dill, cython, cycl
er, cssutils, cssselect, colorama, charset-normalizer, certifi, cachetools, Babel, attrdict, Werkzeug, tqdm, tifffile, s
cipy, requests, rapidfuzz, PyWavelets, pandas, packaging, openpyxl, opencv-python, opencv-contrib-python, multiprocess,
Jinja2, imageio, contourpy, click, bce-python-sdk, scikit-image, premailer, matplotlib, flask, imgaug, Flask-Babel, visu
aldl, paddleocr
  Running setup.py install for future ... done
Successfully installed Babel-2.10.3 Flask-Babel-2.0.0 Jinja2-3.1.2 MarkupSafe-2.1.1 Pillow-9.2.0 PyWavelets-1.4.1 Werkze
ug-2.2.2 attrdict-2.0.1 bce-python-sdk-0.8.74 cachetools-5.2.0 certifi-2022.9.24 charset-normalizer-2.1.1 click-8.1.3 co
lorama-0.4.5 contourpy-1.0.5 cssselect-1.1.0 cssutils-2.6.0 cycler-0.11.0 cython-0.29.32 dill-0.3.5.1 et-xmlfile-1.1.0 f
lask-2.2.2 fonttools-4.37.3 future-0.18.2 idna-3.4 imageio-2.22.0 imgaug-0.4.0 importlib-metadata-4.12.0 itsdangerous-2.
1.2 jarowinkler-1.2.2 kiwisolver-1.4.4 lxml-4.9.1 matplotlib-3.6.0 multiprocess-0.70.13 networkx-2.8.6 numpy-1.23.3 open
cv-contrib-python-4.6.0.66 opencv-python-4.6.0.66 openpyxl-3.0.10 packaging-21.3 paddleocr-2.6.0.1 pandas-1.5.0 premaile
r-3.10.0 protobuf-4.21.6 pycryptodome-3.15.0 pyparsing-3.0.9 python-dateutil-2.8.2 rapidfuzz-2.10.1 requests-2.28.1 scik
it-image-0.19.3 scipy-1.9.1 tifffile-2022.8.12 tqdm-4.64.1 visualdl-2.4.1

(runtime_env) E:\ANRS\Environments>
```

Figure 30. paddle OCR installation confirmation

With this, our Paddle Ocr is ready to use.

3.2.4 For MySQL

3.2.4.1 Download MySQL software

MySQL software to create a local host server for MySQL. It is an open-source local host server providing several functionalities through the package of software it contains.

go to https://dev.mysql.com/downloads/installer/ and download as per the platform

Implementation

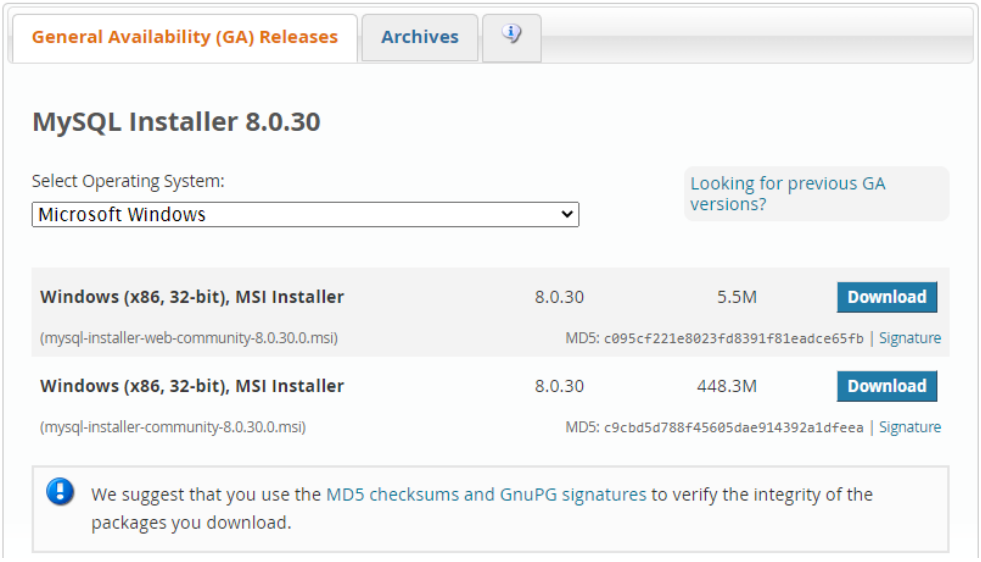

Figure 31. MySQL download page

At the installation time, you will need to create your Password, which needs to be entered in the code in "my_db_other.py". Once installation is done the below screen will be visible.

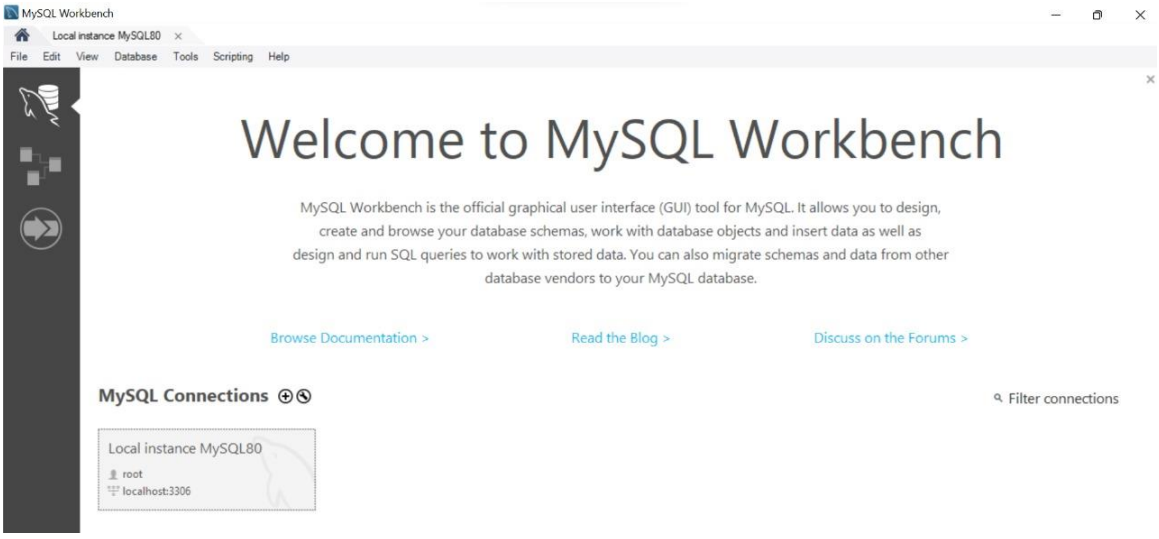

Figure 32. MySQL runtime

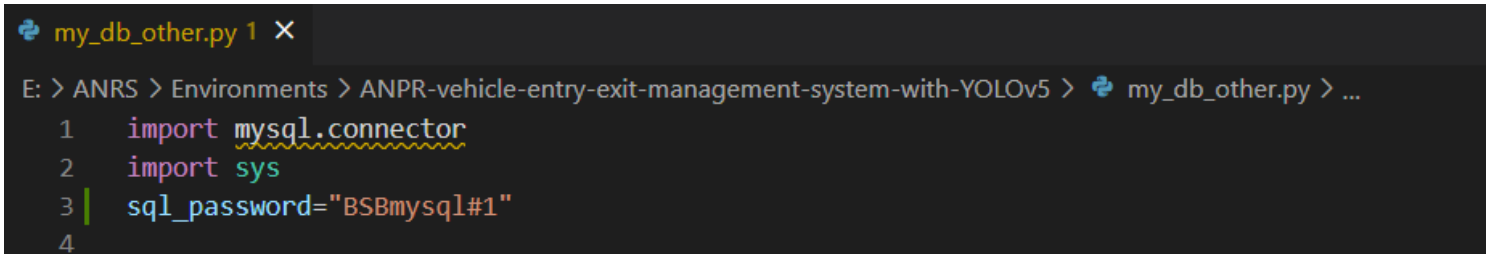

Figure 33. Password for MySQL root user

Implementation

3.2.4.2 Install MySQL-related dependency

Before we get started first ensure that we are in runtime environments

To Install MySQL and related dependencies Run the following command

"*pip install MySQL-connector-python Flask-MySQLdb*"

```
(runtime_env) E:\ANRS\Environments>pip install mysql-connector-python Flask-MySQLdb
Collecting mysql-connector-python
  Using cached mysql_connector_python-8.0.30-cp39-cp39-win_amd64.whl (7.8 MB)
Collecting Flask-MySQLdb
  Using cached Flask-MySQLdb-1.0.1.tar.gz (4.3 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: protobuf<=3.20.1,>=3.11.0 in e:\anrs\environments\runtime_env\lib\site-packages (from mys
ql-connector-python) (3.20.0)
Requirement already satisfied: Flask>=0.12.4 in e:\anrs\environments\runtime_env\lib\site-packages (from Flask-MySQLdb)
(2.2.2)
```

Figure 34. MySQL-related Dependencies Installation

```
=3.6.0->Flask->Flask-MySQL) (3.8.1)
Requirement already satisfied: MarkupSafe>=2.0 in e:\anrs\environments\runtime_env\lib\site-packages (from Jinja2>=3.0->
Flask->Flask-MySQL) (2.1.1)
Using legacy 'setup.py install' for Flask-MySQLdb, since package 'wheel' is not installed.
Installing collected packages: PyMySQL, protobuf, mysqlclient, mysql-connector-python, Flask-MySQLdb, Flask-MySQL
  Attempting uninstall: protobuf
    Found existing installation: protobuf 4.21.6
    Uninstalling protobuf-4.21.6:
      Successfully uninstalled protobuf-4.21.6
  Running setup.py install for Flask-MySQLdb ... done
Successfully installed Flask-MySQL-1.5.2 Flask-MySQLdb-1.0.1 PyMySQL-1.0.2 mysql-connector-python-8.0.30 mysqlclient-2.1
.1 protobuf-3.20.1

(runtime_env) E:\ANRS\Environments>
```

Figure 35. MySQL-related Dependencies confirmation

With this, Our Environment setup is finished.

3.3 Model Implementation (YoloV5)

Training in YOLO is pretty straightforward.

Since we already have prepared our dataset and set up the environment. All we need to do is train our model.

The process is the same for all the model configurations.

3.3.1 Model Training

For model training, the following steps need to be performed.

3.3.1.1 Divide data into training and validation sets.

We have kept our data in two separate folders in a 7:1 ratio, that is

1. For_Training
2. For_Valid

3.3.1.2 Create a "data.yaml" file

This Data.yaml file defines the 4 things for the model

Implementation

1. Training data
2. Validation data
3. Number of classes
4. List of Names of the classes

For us, the File content is as follows

```
train: "E:\ANRS\Environments\Dataset\For_Training"
val: " E:\ANRS\Environments\Dataset\Train_valid"
nc: 1
names: [
    'license_plate'
]
```

To create this open a text editor and paste the above text and then save it as .yaml with the name “data.yaml” in Dataset Folder.

Once that is done we are ready to train our model.

3.3.1.3 Training YOLO model

Now open the cloned YOLO repository and open the cmd on that location, or “cd” to that location

Now run this command

```
python train.py --
data E:\ANRS\Environments\Dataset\data.yaml --
cfg yolov5x.yaml --batch-size -1 --name Model --epochs 100 --
device 0
```

here we are running the “train.py” file with 6 attributes which are:

1. Data → need to specify the yaml file here
2. Config → need to specify model configuration here, where yolov5*.yaml where * = x/l/m/n/s
3. Batch → need to mention the batch size for training or “-1” for auto batch size selection
4. Epochs → number of epochs to be run
5. Device → 0 for GPU cpu for CPU
6. Name → name of the Model

Once the training is over it will generate weights named “best.pt” at the following location

".\runs\train\Model\weights\best.pt"

3.3.1.4 Export the weights to onnx

For this, we need to run another command in the same environment and working directory

Implementation

```
python export.py --weights runs/train/Model1/weights/best.pt --include onnx --simplify
```

Once the exporting is over it will generate weights named “best.onnx” at the following location

".\runs\train\Model\weights\best.onnx"

Now it is ready to be used in OpenCV to import and perform detection tasks.

3.4 Model Implementation → OCR

From the Object detection section, we receive bounding box coordinates and confidences which needs to be fed to the OCR section with some pre-processing.

3.4.1 Pre-Processing

Some basic preprocessing is done with the received bounding box on the copy of the input frame.

1. Grayscale conversion→ needed for faster and simpler operation.
2. Resizing→ as per required size for OCR.
3. Blurring → to dull down noises.
4. Thresholding→ to make image binary for better performance by OCR.
5. Dilation → to grow text by volume for finely printed numberplate text.

```
roi_bgr = cv2.cvtColor(roi, cv2.COLOR_RGB2BGR)
gray = cv2.cvtColor(roi_bgr, cv2.COLOR_BGR2GRAY)

resize = cv2.resize(gray, None, fx = 3, fy = 3, interpolation = cv2.INTER_CUBIC)
blur = cv2.GaussianBlur(resize, (5,5), 0)
mblur = cv2.medianBlur(blur, 3)

ret, thresh = cv2.threshold(mblur, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
rect_kern = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
dilation = cv2.dilate(thresh, rect_kern, iterations = 1)
```

Figure 36. Pre-Processing for OCR

3.4.2 Feeding preprocessed image to Paddle OCR

After preprocessing we feed the result pre-processed image to Paddle OCR and it gives us output as a list of tuples consisting of elements as extracted string and confidence.

```
ocr = PaddleOCR(lang='en',rec_algorithm='CRNN')
text =ocr.ocr(dilation, cls=False, det=False)
a=text[0]
text=list(a)
```

Figure 37. Feeding Pre-Processed Image to OCR

3.5 Database Program Building → MySQL

To start with the MySQL and related services we need to start the XAMPP software, as we did while setup.

3.5.1 Create the MySQL Database

The Below code will create the MySQL database

```
import mysql.connector
import sys
sql_password="BSBmysql#1"

db= mysql.connector.connect(host="localhost", user="root", passwd=sql_password, database="my_test")
mycourser=db.cursor()

creating database #Note remove the databse from db before running below code if already exist
mycourser.execute("CREATE DATABASE  my_test")
```

Figure 38. MySQL Database Creation

3.5.2 Create a table in the Database

Below code will create a table in the database “my_test”

```
import mysql.connector
import sys
sql_password="BSBmysql#1"

db= mysql.connector.connect(host="localhost", user="root", passwd=sql_password, database="my_test")
mycourser=db.cursor()

#to create a table in database
mycourser.execute("""CREATE TABLE anpr (
                    EMP_ID INT PRIMARY KEY,
                    NAME  VARCHAR(250) NOT NULL,
                    PHONE_NO INT NOT NULL,
                    VECHICLE_NO VARCHAR(250)
                    )""")
```

Figure 39. Table Creation

3.5.3 Initiate MySQL Engine

The below code will start the MySQL engine to connect the server through python.

```
import mysql.connector
import sys
sql_password="BSBmysql#1"

db= mysql.connector.connect(host="localhost", user="root", passwd=sql_password, database="my_test")
mycourser=db.cursor()

# this will show availble databases
mycourser.execute("show databases;")
for i in mycourser:
    print(i)
```

Figure 40. Initiating MySQL engine

Implementation

3.5.4 Inserting data to the database table

The below code will insert data to the database's table.

```
def menu(self):
    user_input=input("""
                1.Add the vehical in data base:
                2.Check if vehicle is already in our database:
                """)
    if user_input=="1":
        self.register()

    elif user_input =="2":
        self.numberplate()

    else:
        sys.exit(1000)


def register(self):
    EMP_ID = int(input("Enter the employ id"))
    name = input("Enter the owner name")
    phone_no= int(input("Enter the mobile number"))
    vehicle_no = str(input("Enter the vehicle number"))

    response = self.db.register(EMP_ID,name,phone_no,vehicle_no)

    if response:
        print("Registration successful")

    else:
        print("Registration faild")
```

Figure 41. Inserting data to the database's table (a)

```
def register(self,EMP_ID,NAME,PHONE_NO,VECHICLE_NO):
    try:
        self.mycursor.execute("INSERT INTO anpr VALUES(%s,%s, %s,%s)",(EMP_ID,NAME,PHONE_NO,VECHICLE_NO))
        self.conn.commit() ## to enter the data into table from ram

    except:
        return -1
    else:
        return 1
```

Figure 42. Inserting data to the database's table (b)

3.5.5 Retrieve Data from the database

The below code will retrieve data from the table

Implementation

```
def numberplate(self):
    reg_plate=input("Enter the vehicale number plate")

    self.db.search(reg_plate)
```

Figure 43. Data Retrieval from the database (a)

```
def search(self,reg_plate):  ## to read the record from DB

    try:
        self.mycursor.execute("SELECT * FROM anpr WHERE VECHICLE_NO = (%s)",(reg_plate,))

        data=self.mycursor.fetchall() # basically we are getting all stored vale from mycursor
        for x in data:
            print(x)

    finally:
        self.conn.close() # disconnecting from server
```

Figure 44. Data Retrieval from the database (b)

Chapter 4: Results

Thus we have built and acquired an accuracy of 97.94%.

For OCR accuracy we use the Levenshtein Distance method in which we find the Levenshtein distance between actual text and ocr extracted text. With the help of this, we found the accuracy of OCR used. In this, We feed a set of images to function above and find accuracy image by image and after this, we find overall accuracy.

The formula for Levenshtein Distance:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Figure 45. Levenshtein Distance formula

```
def Levenshtein(ground_value,ocr_value):

  error=distance(ground_value, ocr_value)
  actual=len(ground_value)
  per=(actual-error)/actual*100

  return per
```

Figure 46. Levenshtein distance function

```
acc_paddleocr=[]
for i,j in zip(ground_value,ocr_value_paddleocr):
  accuracy=Levenshtein(i,j)
  acc_paddleocr.append(accuracy)

print(acc_paddleocr)

total_acc_paddleocr=np.mean(acc_paddleocr)
total_acc_paddleocr
```

Figure 47. Levenshtein distance implementation

Chapter 5: Conclusion

Thus, we have completed our Project with satisfactory results abiding by our timeline. We have trained Our Object detection Model using YOLOv5 which is one of the fasters and real-time algorithms known to detect Number Plates in vehicles. We implemented Paddle OCR to read the Number Plate text from the number plate image detected. We set up a MySQL database to maintain an Employee database with vehicle registration. We also integrated the complete models into a GUI using the Flask library in Python. With this, we have made a solution to manage Vehicle Entry Exit Management System Using Automatic Number Plate Recognition utilizing AI Technologies such as CNN, Machine Vision, OCR and SQL.

Chapter 6: References

Zhao, Z.Q., Zheng, P., Xu, S.T. and Wu, X., 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), pp.3212-3232.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Redmon, J. and Farhadi, A., 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Bochkovskiy, A., Wang, C.Y. and Liao, H.Y.M., 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, tkianai, yxNONG, Adam Hogan, lorenzomammana, AlexWang1900, Jan Hajek, Laurentiu Diaconu, Marc, Yonghye Kwon, oleg, … Francisco Ingham. (2021). ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations (v5.0). Zenodo. https://doi.org/10.5281/zenodo.4679653

Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q. and Wang, H., 2020. Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*.