AUTOMATIC LICENSE PLATE RECOGNITION:

Abstract: ALPR is the technique of extracting the information from an image or a sequence of images of Vehicle's Number Plate. The image will be captured from any camera. This captured image will be then sent to the System. System will detect car from the image and perform pre-processing on this on this image. System will perform the plate region extraction on the image and will convert it into grey scale. Then the number will be extracted character by character using Neural Network.

Introduction: Each License plate has a unique number assigned to it for vehicle identification. ALPR will work in small scale with the use of any camera device. For license plate detection purpose the concept of edge detection, contour determination and bounding box formation and elimination is used. Selection of license plate areas (LPA) and their elimination to obtain the actual license plate was based on various heuristics. This stage is important since improper detection of LPA can lead to misrecognized characters. Character Extraction or character segmentation is the important component of our ALPR system. It takes a properly segmented license plate as an input. Some preprocessing is done on the license plate image for the removal of noise. Various morphological operators are used on the image for this purpose and the noise free output image is sent for character segmentation. Image binarization and image projections are used for character extraction. image for this purpose and the noise free output image is sent for character segmentation. Image binarization and image projections are used for character extraction. In this project work we have developed a full-edged Automatic License Plate Recognition (ALPR) system with hardware implementation.

Phases of ALPR:

ALPR system work according to the given phases:

- Obtain image
- Detect car from the image
- License Plate Detection
- License Plate Separation
- License Plate Segmentation
- Number Identification

Working of ALPR:

1. **Input a RAW image:** Capture the image from camera. The resolution of the camera should be good so that the captured image can be further utilized for processing. Capture Image is given to Yolov5 to detect car.

2. **Detect Car:** Yolov5 detects car from the image and passes the cropped image having car to the ALPR as input.

3. **Detect License Plate:** ALPR first converts the image from BGR to HSV format and creates mask for yellow color (for SINDH License Plates). Then the mask and edge detection is used to detect the License Plate area. The License Plate portion is then cropped from the original image and then passed to the OCR for plate character recognition as input.

4. **OCR:** OCR first auto sets the brightness and contrast of the image using histogram clipping. Then it performs thresholding of the image using Histogram area to separate the characters from the image. Then this preprocessed image is used to draw contours on the characters of the license plate and sends it to the model for detection of character.

5. **Model:** Model passes the input image to the neural network to predict the character and then returns the predicted character with highest accuracy.

OCR Model: OCR is a custom CNN model which is used to detect the characters of number plates. It includes 35 classes i.e. A-Z and 0-9. It is trained on approximately 10000 images of characters from random license plates with an accuracy of 80.02%. It has three layers such that, an input layer of size 784 and an output layer of size equal to classes. In between there is a hidden layer of size 100. The model is trained for 150 epochs with a learning rate of 0.1. It takes images of characters and returns a string as label.

Flow of Operations:

- **ALPR:**

1. Importing Libraries:
   - 1.1 cv2 to read, view and apply filters on the image.
   - 1.2 Pytesseract to read text from the image
   - 1.3 Numpy to make matrices and do mathematical operations
   - 1.4 Os for dealing with images and paths
   - 1.5 Re for regular expression.

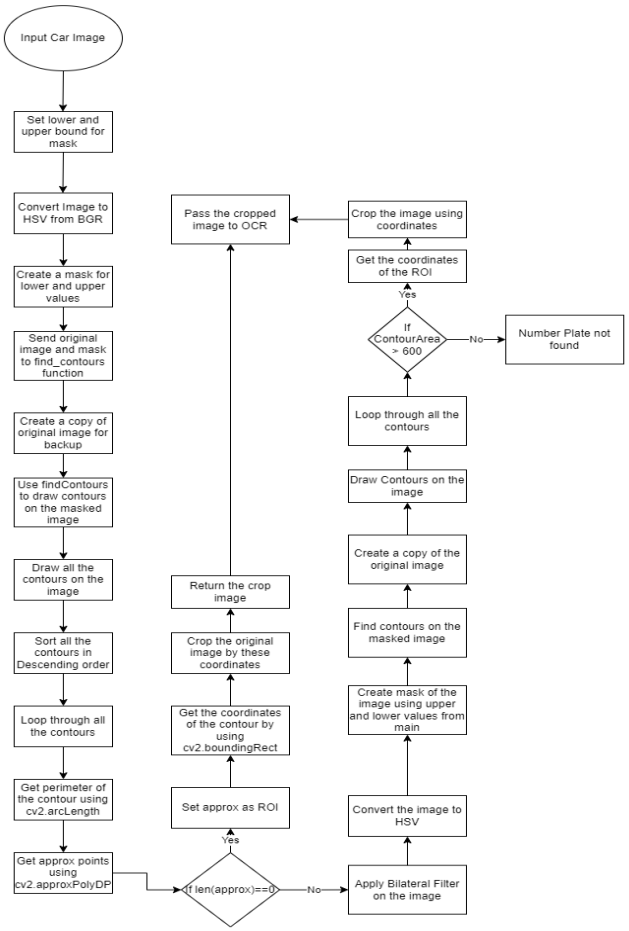
2. Defined a function find_contours with image,mask,lower and upper as paramters:
   - 2.1 Use cv2 find contours on the mask.
   - 2.2 Create a copy of original image.
   - 2.3 Use cv2 draw contours to draw the contours on the image.
   - 2.4 Sort the top 10 contours in descending order.
   - 2.5 Iterate through all the contours:
     - 2.5.1 Get perimeter of the contour using cv2 arc length.
     - 2.5.2 Get the contour with polygonal curve to get the approx. number plate contour.
     - 2.5.3 Check if approx contour has four points:
       - 2.5.3.1 Set approx. as number plate contour.

2.5.3.2 Get x,y,w,h of the contour using bounding rect.
2.5.3.3 Draw a rectangle on these coordinates.
2.5.3.4 Create a crop_img of number plate using these coordinates.

2.5.4 If not 4 points:
2.5.4.1 Convert image to HSV
2.5.4.2 Create mask according to lower and upper parameters.
2.5.4.3 Dilate the mask to make ROI prominent.
2.5.4.4 Find contours on the dilated image.
2.5.4.5 Create a copy of original image.
2.5.4.6 Draw contours on the original image.
2.5.4.7 Sort the contours in descending order.
2.5.4.8 For Contours of area>500 we will get x,y,w,h.

3. Defined a write function which will image and count as parameters. It will save the image in the directory with name as count and will return count with an increment of 1

4. Main function:
4.1 Define the path of the files.
4.2 Loop through all the files in the folder and read them through opencv.
4.3 Set the upper and lower values of the kernel for mask.
4.4 Convert the image to hsv
4.5 Create the mask of the converted image.
4.6 Call the find_contours function for number plate processing, If any error occurs call Number Plate Not Found exception.
4.7 Write the processed image in the results folder.



- **Car Detection:**

1. Importing Libraries:
   1.1 os & glob for accessing operating system paths and performing operations
   1.2 matplotlib for viewing the images as they can't be viewed via opencv in jupyter notebook
   1.3 cv2 to read and perform operations on the image
   1.4 requests to download data from internet via url
   1.5 numpy for mathematical operations

2. Setting Constant Values:

   2.1 Setting seed to a random value for getting different results everytime the notebook is executed
   2.2 Setting TRAIN = True for the execution of train function
   2.3 Setting EPOCHS = 50 to train the model on the dataset for 50 epochs
   2.4 Downloading the dataset from roboflow using api

3. Defined a function to download files from the internet via url.

4. Defined a yolo2bbox function to convert bounding boxes in YOLO format to a normal opencv format.

5. Defined a plot_box function:

   5.1 It will have image, bboxed and labels as parameters.
   5.2 Iterating through the bboxes and normalize the coordinates.
   4.3 Draw rectangles on the image using opencv.

6. Defined a plot function to extract the images, labels from the image and label path.

   6.1 Store the images and labels from their path.
   6.2 Sort the images and labels.
   6.3 Get the number of images.
   6.4 Iterate through the images and read the image.
   6.5 open the label file of the image and extract the bounding box and coordinates.
   6.6 pass the image coordinates, labels to plot_box to draw the rectangle.

7. Defined a monitor_tensorboard function to load the tensorboard for viewing the results.

8. Check if yolov5 repo already exists in the file folder. If not then clone the repository:

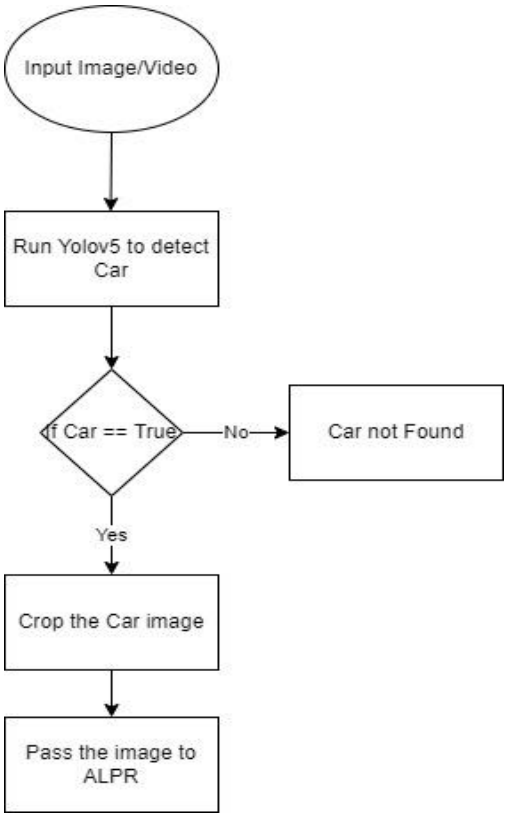
9. Install the requirements.txt to make the system ready for yolo

10. Train the model on custom dataset having the following parameters.

    --data => path of training images
    --wieghts => which yolov5 model to be used
    --img => image size
    --epochs => no of iterations
    --batch_size => how much data should be passed in one batch

--name => directory name where results will be stored

11. Function to set_resdir to create directory for storing results.
12. Function show_valid_results to show validation set predictions by the model saved during training.
13. We take snaps from the video frames i.e 2 frames per second instead of 60 to make data short and save them in directory
14. We run detect.py and pass the above saved results of snaps and detec.py will be modified to target only those who have 80% confidence and their bounding box area is greater than 144600 and smaller than 200000. Detect.py will save in the RES_DIR the image with bounding box as well as the cropped image of only the object inside the bbox.



- **OCR:**

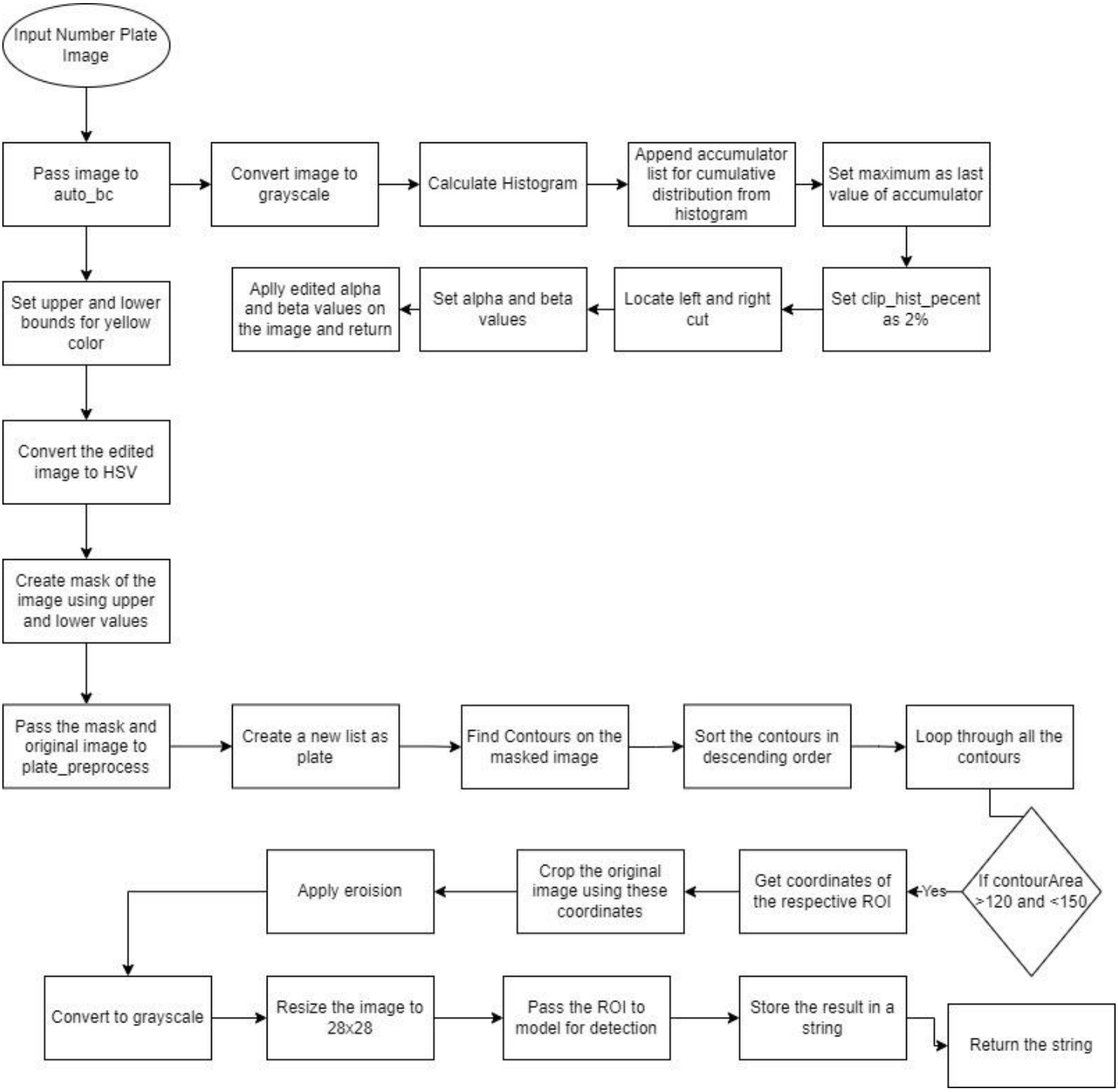
1. Import Libraries
   1.1 cv2 to read and perform image operations
   1.2 numpy for mathematical operations
   1.3 model_test for passing the characters to CNN for detection

2. main Function:
   2.1 Set the upper and lower values for yellow color
   2.2 Pass the image to auto_bc function to enhance the brightness and contrast of the image
   2.3 Convert the enhanced image to HSV format
   2.4 Create a mask of the image using cv2.inRange for the upper and lower values
   2.5 Pass the mask to plate_preprocess for processing the image and text detection
   2.6 Pass the plate string and number plate to write function to write the image with it's predicted text

3. auto_bc Function:
   - 3.1 Convert the image to gray scale.
   - 3.2 Calculate histogram using cv2.hist
   - 3.3 Get size of the histogram
   - 3.4 Set the accumulator to value for all values of hist to calculate cumulative distribution from the histogram.
   - 3.5 Set maximum as the last value of accumulator
   - 3.6 Set clip_hist_percent to 2%
   - 3.7 Set min_gray to zero and while accumulator is less than 2% increment min_gray to locate left cut
   - 3.8 Set max_gray to zero and while accumulator is greater than or equal to 2% increment max_gray to locate right cut
   - 3.9 Calculate alpha and beta values for new brightness and contrast
   - 3.10 Pass the image, alpha and beta to convertScale Function to get enhanced image
   - 3.11 Return enhanced image

4. convertScale Function:
   - 4.1 Create a new image using previous image and new alpha and beta values
   - 4.2 Set all values of new_img less than zero to zero
   - 4.3 Set all values of new_img greater than 255 to 255
   - 4.4 Set the datatype of new image to uint8
   - 4.5 Return new image

5. plate_preprocess Function:
   - 5.1 Create a new list as plate
   - 5.2 Find all the contours on the image
   - 5.3 If len(contours) == 2:
     - 5.3.1 contours = contours[0]
   - 5.4 else contours = contours[1]
   - 5.5 Loop through all the contours:
     - 5.5.1 if contour area > 120 and < 500:
       - 5.5.1.1 Get coordinates of the contour by bounding rect
       - 5.5.1.2 Crop the original image with these coordinates to get the ROI
       - 5.5.1.3 Erode the ROI with kernel (3,5)
       - 5.5.1.4 Convert the ROI to grayscale
       - 5.5.1.5 Resize the ROI to 28x28
       - 5.5.1.6 Pass the ROI to the model for prediction
       - 5.5.1.7 Append the plate list with the predicted value
   - 5.6 Return the plate list


Input Number Plate Image
Pass image to auto_bc
Convert image to grayscale
Calculate Histogram
Append accumulator list for cumulative distribution from histogram
Set maximum as last value of accumulator
Set clip_hist_pecent as 2%
Locate left and right cut
Set alpha and beta values
Aplly edited alpha and beta values on the image and return
Set upper and lower bounds for yellow color
Convert the edited image to HSV
Create mask of the image using upper and lower values
Pass the mask and original image to plate_preprocess
Create a new list as plate
Find Contours on the masked image
Sort the contours in descending order
Loop through all the contours
If contourArea >120 and <150
Yes
Get coordinates of the respective ROI
Crop the original image using these coordinates
Apply eroision
Convert to grayscale
Resize the image to 28x28
Pass the ROI to model for detection
Store the result in a string
Return the string