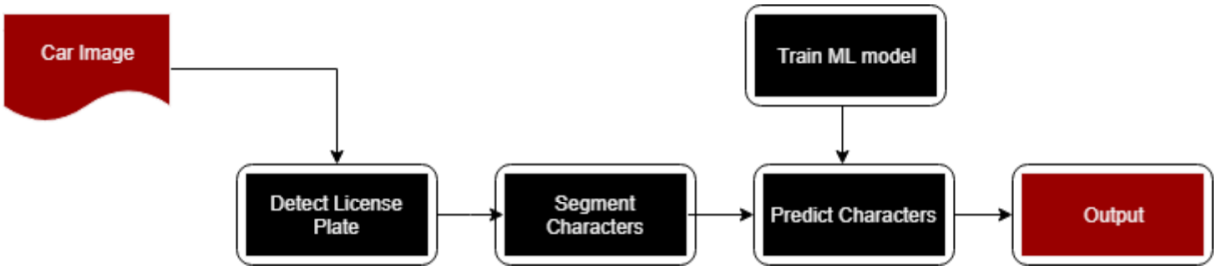
Kamaneh Akhavan
Machine learning project report

This project was with challenges such as collecting data sets, I could not find a free database of Iranian license plates, I tried to generate it myself, which was very difficult due to the uncertainty of the font and finding the appropriate photos because in many of them the font was unclear or the license plate was not clear, which was due to the wrong viewing angle or light, there was a change in the background, in which case it was difficult to detect. I also needed a more sophisticated model to identify the license plate number at night in low light, which was not possible due to the short time.
The capabilities of the project are as follows:

- Detect a car license plate in the image
- Detection and identification of external plaque
- Recognition and identification of Iranian license plates
- Ability to teach new data for both foreign and Iranian license plates
- Extract the desired attribute for each new data
- Fast and accurate for flat images

Implementation steps

- ❖ Capture input image or frame
- ❖ Pre-processing
- ❖ license plate detection in the image(CCA-Connected Component Analysis algorithm)
- ❖ Extract the characters in the license plate
- ❖ License plate identification




For Iranian license plate I had a small data set which I made it so KNN algorithm could help a lot here with its classification and give us high accuracy with a small data set.
I have used image processing techniques and algorithms to detect license plates because they are faster and require less detail.
In the license plates extraction characters step in the image, we should be able to extract the characters after the license plates is detected and give it to the KNN classifier to identify it.
There are different methods for extracting characters, the most famous of which is segmentation, and using the contours inside the image, we can perform segmentation on the license plate and extract the characters.After extracting the characters, we can identify the characters with a pre-trained classifier.
Each of the above steps affects the accuracy of the algorithm in some way and all of them produce a final accuracy for us. Choosing the right algorithm is very important.

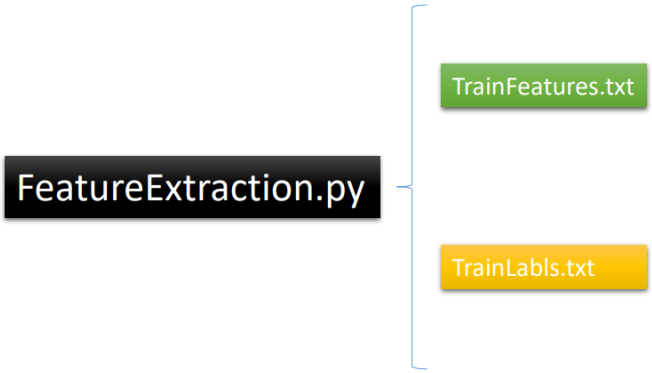
I used image processing algorithms to detect license plate because they take shorter runtime and less memory and do not require much detail, but I used machine learning algorithm to identify license plate.

Kamaneh Akhavan
Machine learning project report

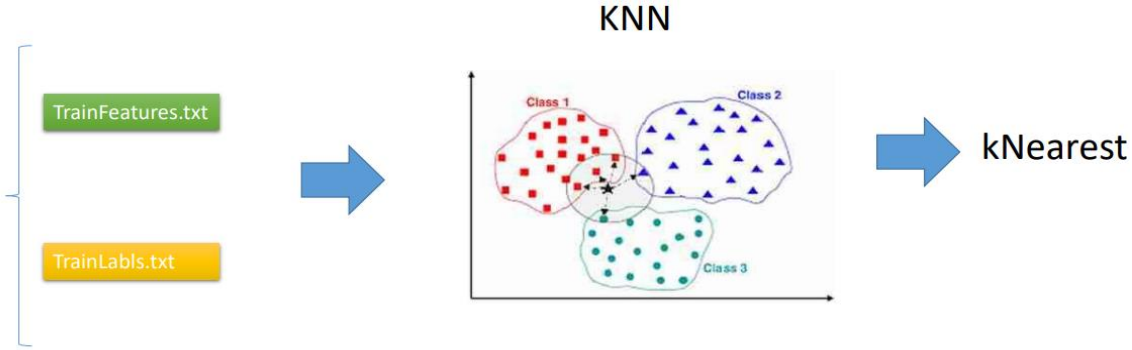
## The code:


- Input image
- Turn gray
- Become a binary
- Apply CCA to capture connected areas
- Plaque detection based on all connected points

FeatureExtraction.py, Here we extract the features from the images which is the first step to prepare the data, I have prepared a series of images of letters and numbers that are important to me in recognizing the license plate which are both English and Persian, which are We give this algorithm and this function gives us two files called TrainFeatures and TrainLables which are the same properties extracted from the image. The inside of the TrainFeatures gives us the properties extracted from that image and the inside of the TrainLables, gives us the label of the input image. As the label puts the same ascii code of that character.




This model is taught using KNN on datasets with 20x 20x characters. After obtaining the scene characters and training the model, the model is loaded to identify each character. When KNN learns it well,it can make a good model(KNearest) for me so that I can do the prediction. KNearest is going to Use to get the characters that will be obtained in the next steps.



Kamaneh Akhavan
Machine learning project report

In the pre-processing we do a series of processing tasks to eliminate noise and segmentation of the image and edge finding and contours. These help us to be able to process the image better.
As an input we will have an image and put it in **imgOriginalScene**(main.py).

```
imgGrayscaleScene, imgThreshScene = Preprocess.preprocess(imgOriginalScene)
# preprocess to get grayscale and threshold images

if Main.showSteps == True: # show steps
#######################################################
    cv2.imshow("1a", imgGrayscaleScene)
    cv2.imshow("1b", imgThreshScene)
```

Grayscale is very common in image processing projects. We do this by reducing the green, red and blue color channels . Because if we do not do this, we have to process three matrices in each step, and this is very laborious and time consuming in both image processing algorithms and machine learning algorithms. This allows us to prevent color confusion and make detection easier. In preprocess.py with imgGrayscale I did grayscale and each image will have useful and useless information, here the useful information is just a plaque and the rest of the image is useless. This useless information is called noise. Unwanted details are typically removed from an image using a two-way filter in this step(imgBlurred).After this step, our image looks like this: “1a”


The next interesting step is where the edge detection is performed. There are many ways to do this, the easiest and most popular way is to use the canny edge method in OpenCV. Then only edges with a slope greater than the minimum threshold and less than the maximum threshold are displayed.( imgThreshScene)Then the image goes as follows: “1b”

And then in Detectplates.py “findPossibleCharsInScene(imgThresh)” function , given a list of all possible chars, find groups of matching chars and in the next steps each group of matching chars will attempt to be recognized as a plate and it finds all the contours, Contours are the boundaries of the objects that are inside the image. With the contour, we can fragment the image and we can see the plate better.


In the final stage, we do a segmentation and a background subtraction. Background subtraction can actually remove the background for us and leave us what has the most connection in the image(The place of the image that has the highest pixel density), which is shown in the pic:

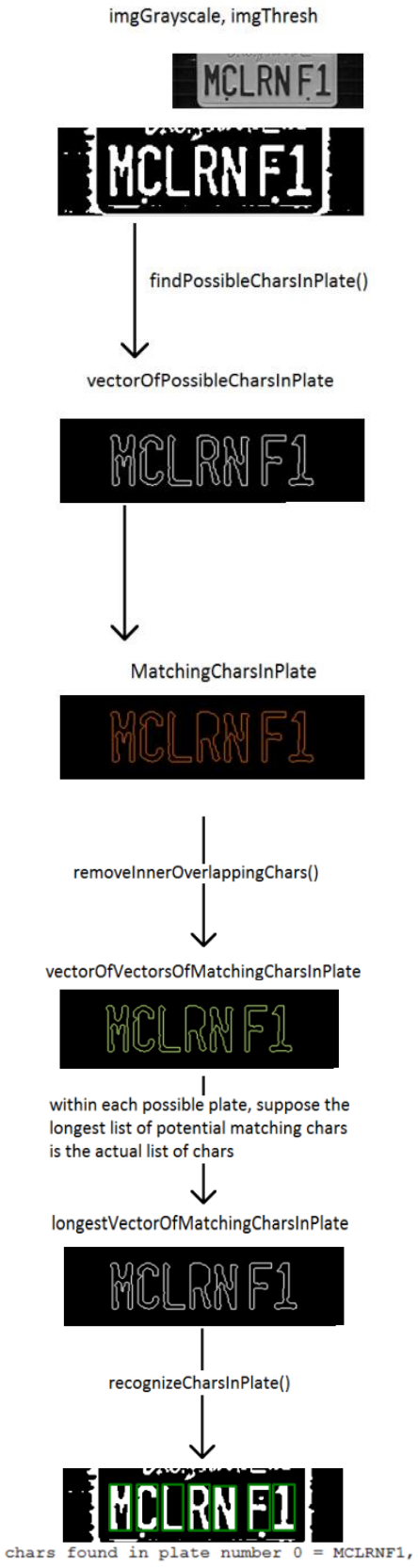

In the license plate detection step, it finds the license plate inside the image, Which uses image processing techniques that are easier and faster.Here the extractions parts ,get bold and Noise relief And vector of vector of matching character:

And then it is found the possible plates by extract plate:


And here it extracted the plate and delete the rest of the picture,


In extracting the characters from the license plate, we use the techniques of reconstruction and reduction of noises, and based on the connectivity that exists in the license plate, we can extract the characters. Here, too, sometimes in some photos, the presence of light or stains on the license plate is very challenging and confuses the system. So noise is a huge challenge, and it is very important to use an algorithm that can detect and reconstruct the image as much as possible, which means that even though it is part of a noisy character, it can reconstruct and detect it.
After detecting the plate, we strengthen the edges of the plate we have identified and reduce the noise as much as possible and fill in the places that have been deleted and remove the extra points. VectorOfvectorsofMactchingcharsinplate and longestVectorOfMatchingcharsplate do those steps for us.Then do the segmentation with recognizeCharinplate function, After extracting the characters, we give them to KNN to identify them.

Kamaneh Akhavan
Machine learning project report



The CCA is used again on the license plate to classify the characters. Similar assumptions are made here for the size of the characters. After splitting the characters, they are resized to 20x 20 and moved to a list. The character extraction step is done by DetectChars.py.

In the plate identification phase, machine learning algorithms do this very well, provided they are given a good data set.
At this stage, the algorithms used in the articles are usually: KNN-SVM-NN-BL-DL
The DL and Bl algorithms should have a good and enough data set and you cannot work them with a small data set.
Most articles use KNN and SVM because of the generalization as well as the good accuracy they have on different data. My goal is to use KNN which is a simple and accurate classifier. KNN uses neighborhoods with a neighborhood radius to decide which characters to classify into which class.

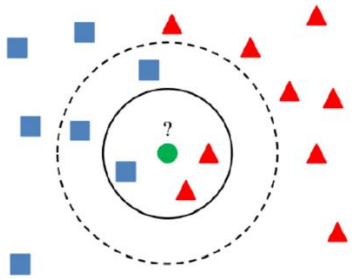
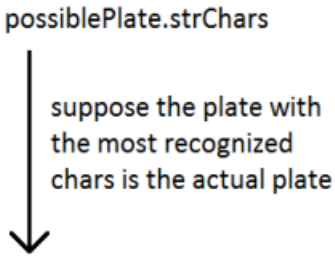



*Figure 1KNN classifier uses neighborhood radius for classifying*




Kamaneh Akhavan
Machine learning project report