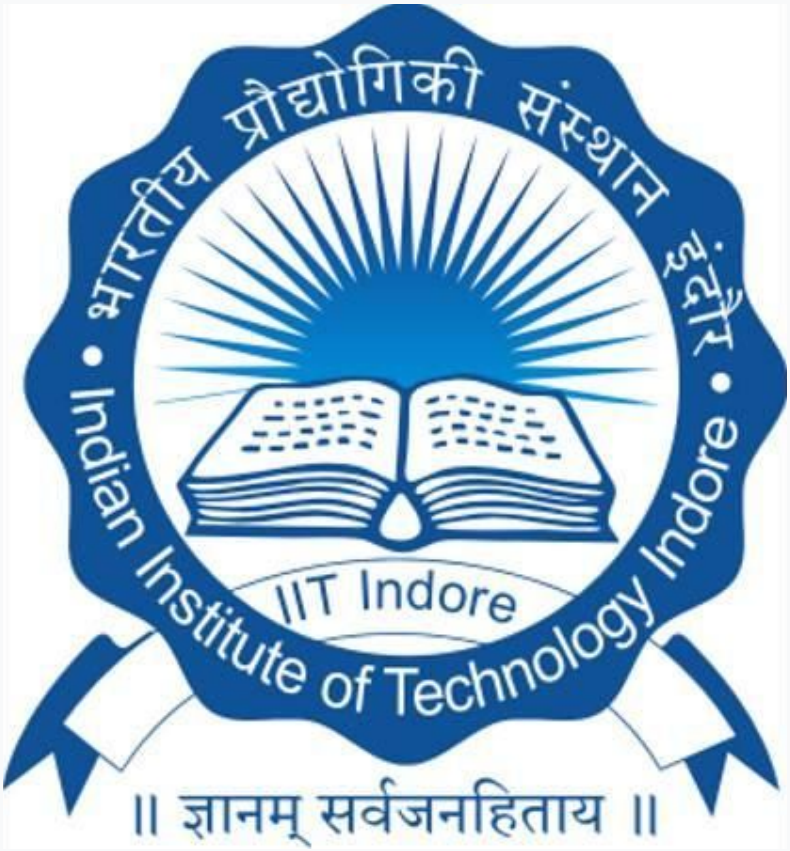


VEHICLE NUMBER PLATE RECOGNITION

CS 354N

COMPUTATIONAL INTELLIGENCE

FINAL PROJECT REPORT

Submitted by:

Anurag Verma 160001005
Aravind Ravikumar 160001006
Mohit Malviya 160001035

Under the Guidance of

Dr. Aruna Tiwari

Objective

To detect the vehicle number from vehicle number plate with high accuracy using image processing.

Problem statement

As Indian vehicle number plate do not follow a standard way, it is very difficult to accurately detect the vehicle number plate from vehicle image. The existing algorithms provide vehicle number plate detection but they are not that accurate and time efficient. Therefore an algorithm is needed to solve this problem which should be more time efficient, reliable(~100% ideally) and consistent(~100% ideally) with support for wide variety of number plates of India, UK, US etc.

Solution Considerations

A program consisting of deep learning and image processing based algorithms that takes following input, processes and gives desired output. This includes a detection and isolation algorithm, and a recognition algorithm. The detection algorithm detects and isolates the number plate from a regular image of a vehicle, which is used by a deep learning algorithm as uniform inputs to recognise the characters in the number plate.

The algorithm must be able to correctly and time efficiently detect vehicle numbers from image that includes one vehicle. This should be done

for various formats for the vehicle number plates that are being used world wide including India, UK, US, etc.

Input

Vehicle number plate images and country to which that vehicle belongs.

Output

Vehicle number of that vehicle.

Test Data

Consider at least 500+ images for test data with all possible day and night time snaps and different climate conditions like rains, fog, dusty, etc.

Deliverables

1. A practical Number plate detection and isolation program that works for above scenarios.
2. A Number plate character recognition program for the above isolated number plate.
3. Testing accuracy results for the collected data sample.

Approach

We have made two number plate detection/isolation programs and three learning models.

Detection 1

Our first number plate detection and isolation program. Doesn't work very well because stock rectangle detection function from found contours in cv2 python package wasn't good enough to identify the number plate edge from grayscale image . Not used in any model.

Detection 2

Better detection and isolation algorithm. Instead of finding and extracting rectangular contours first step, in this model we find possible characters directly from all the contours (doesn't extract number plate initially), and then find number plate contour from the region which contains the most possible character matches. (DetectChars.py, DetectPlates.py,preprocess.py).

KNN Model

We have 5 sets of english alphabets and numbers as training sample (training_chars.png) and use the above detection 2 algorithm to isolate characters, store it in numpy array format for KNN training (using scikit-learn package). And accuracy from cross-validation with 5 sets was around 90%. For final detection, we use k=5. (GenData.py).

SVM Model 1

We took 10 sample 20x20 pixel images of each english alphabet and number as training example("train" folder). We converted the images into 1D binary array for SVM training. We used both poly and rbf variants and got similar accuracy of around 90%. (prediction.py, machine_train.py).

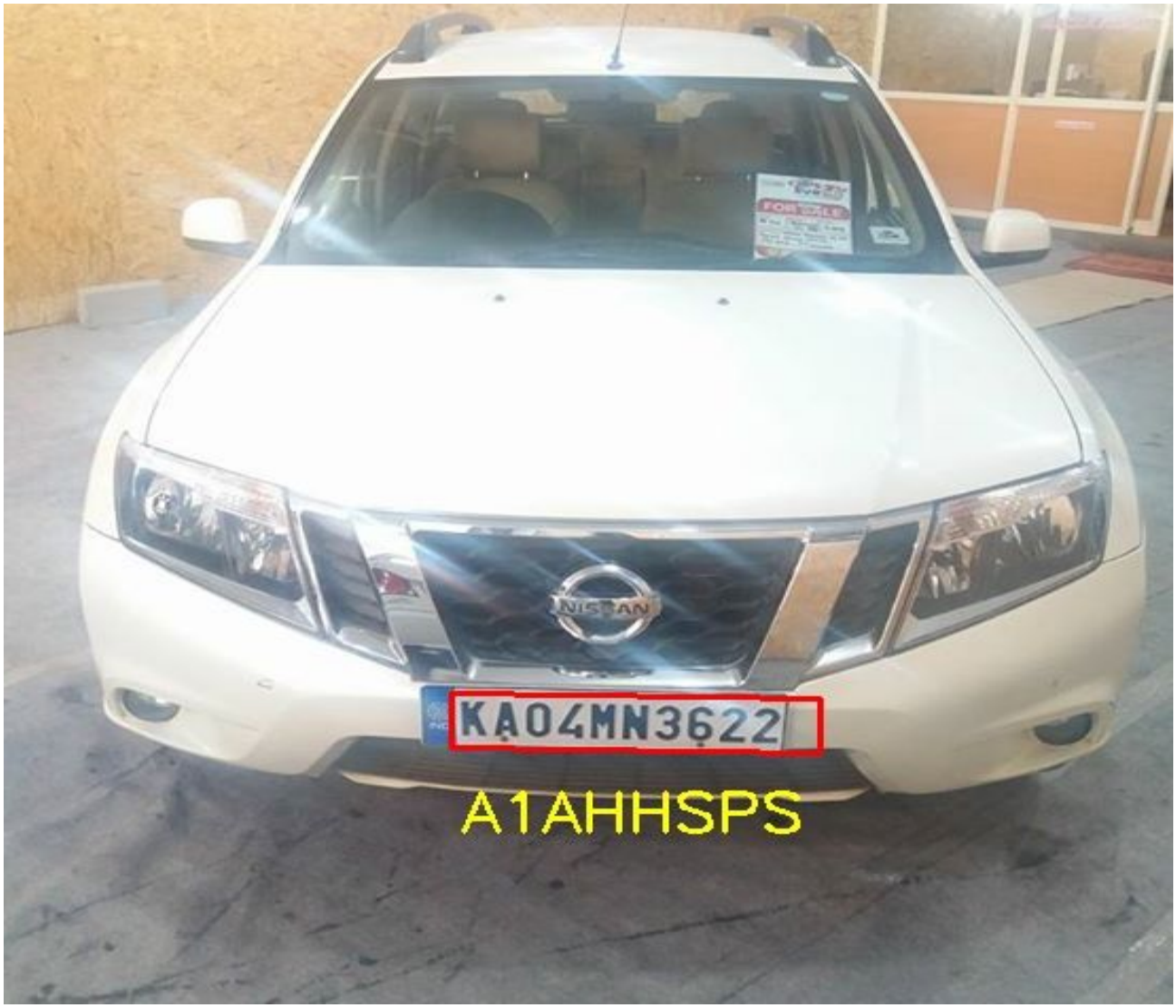
SVM Model 2

We took >30 images of every character and number ("Englishimg"). Repeated the same above algorithm. Ironically, the accuracy decreased to a mere 20%. We expect the reason is because of having too much diverse images, reducing the consistency by a lot. (prediction.py, machine_train.py).

Results

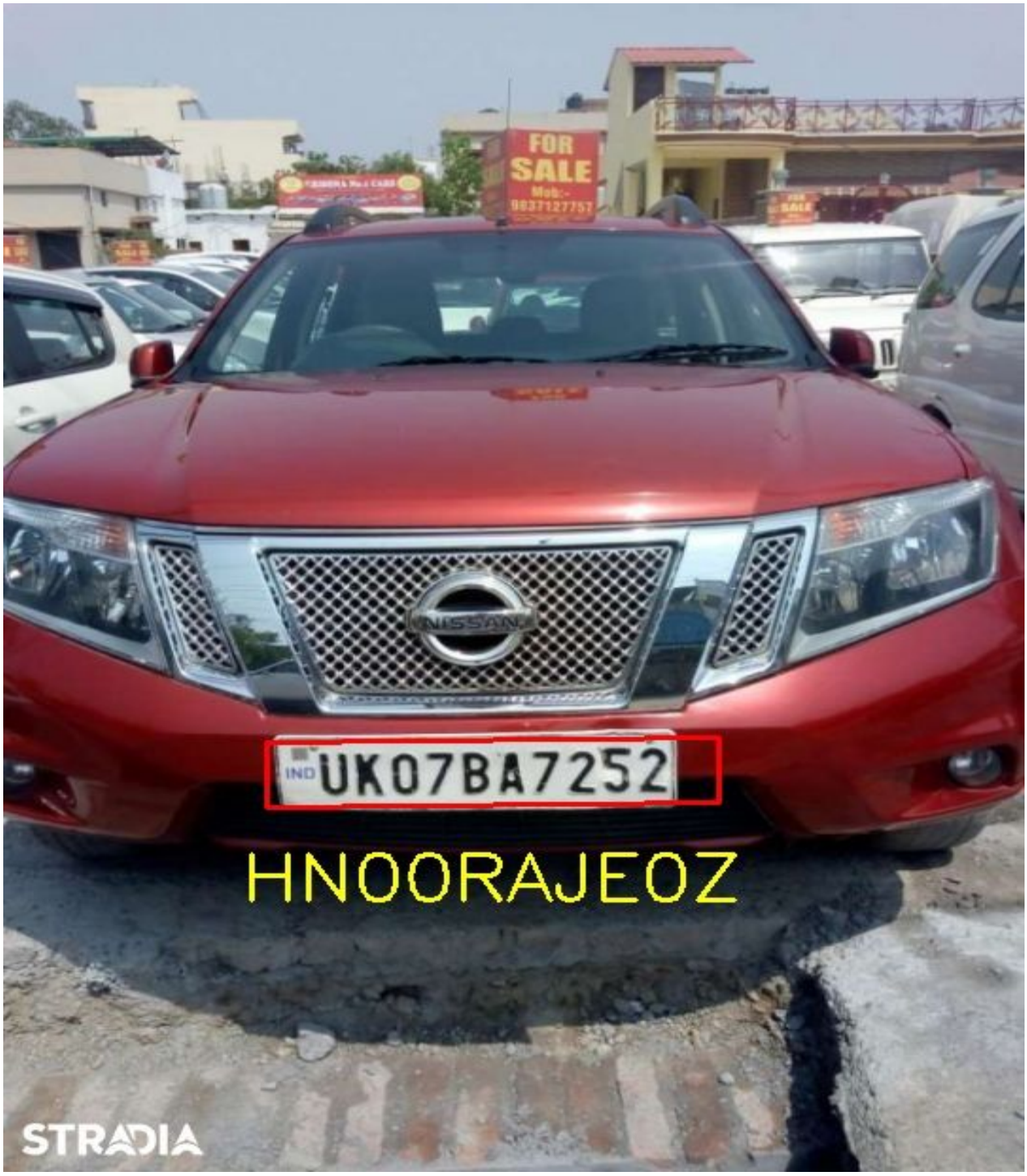
SVM Model 2 resulted in the worst results of 20% recognition of detected and isolated characters while KNN and SVM Model 1 have around 90% accuracy of recognition. All the detected images can be found in the folders of each of the models, in folder "Detected". The corresponding test images are under "Indian_images" folder.

SVM Model 2



SVM Model 2 example

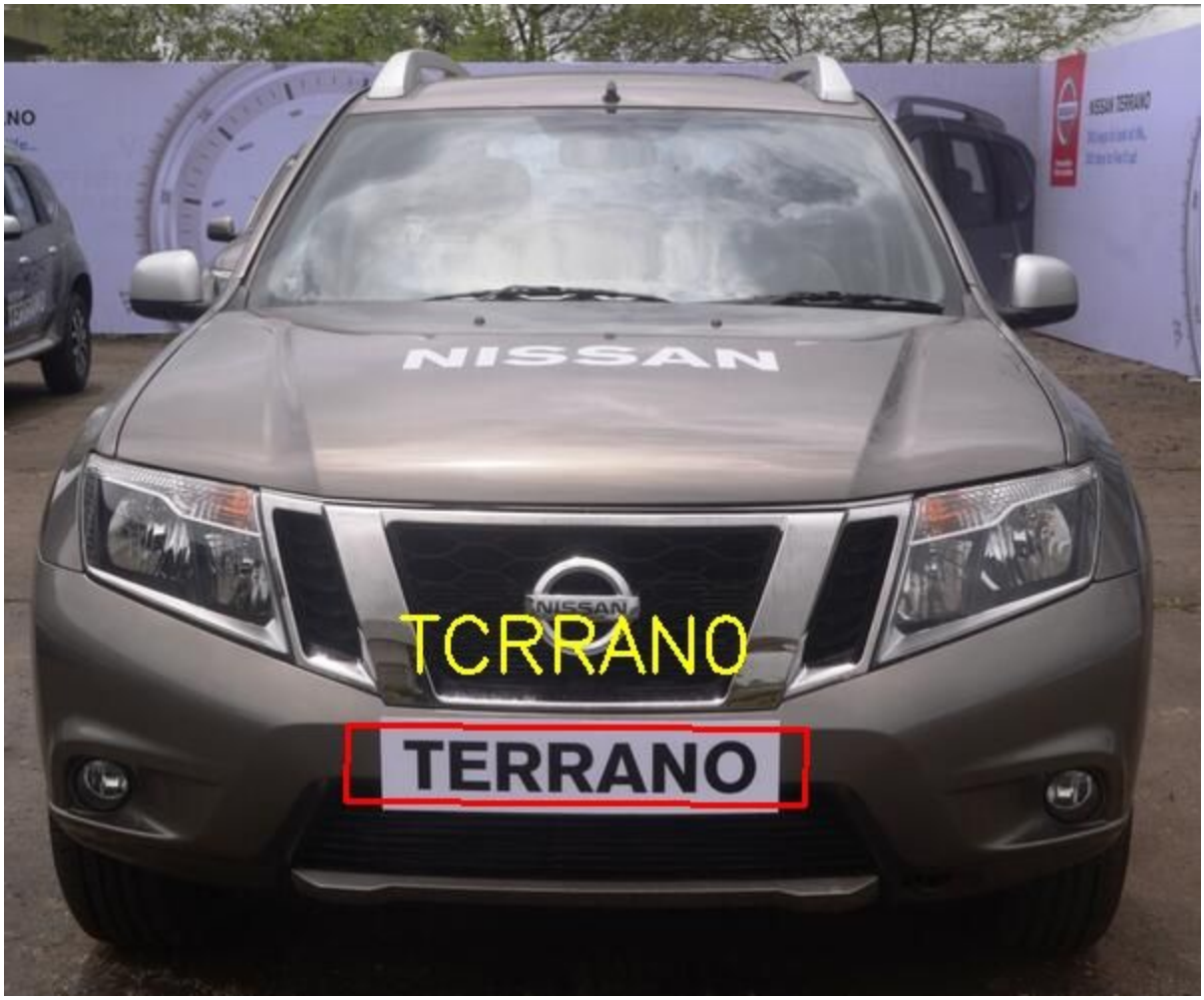
It matched 4->A,M->H, N->H, 6->S. It gets close in predictions, but ultimately accuracy is very low and this model is not very useful.



SVM Model 2 example

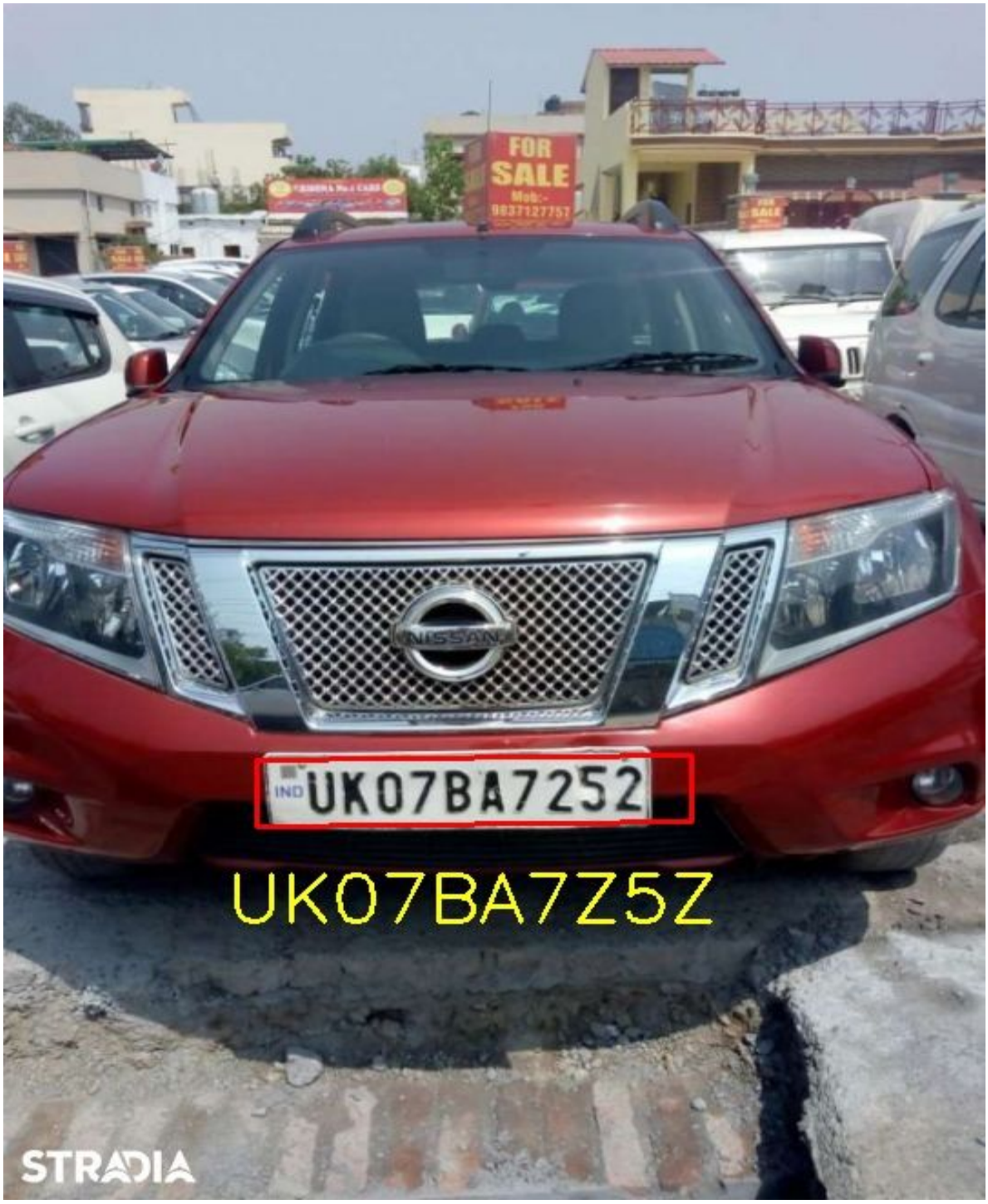
Matched K->N, B->R, 2->Z. Similar result as previous example. Close predictions but low accuracy.

SVM Model 1



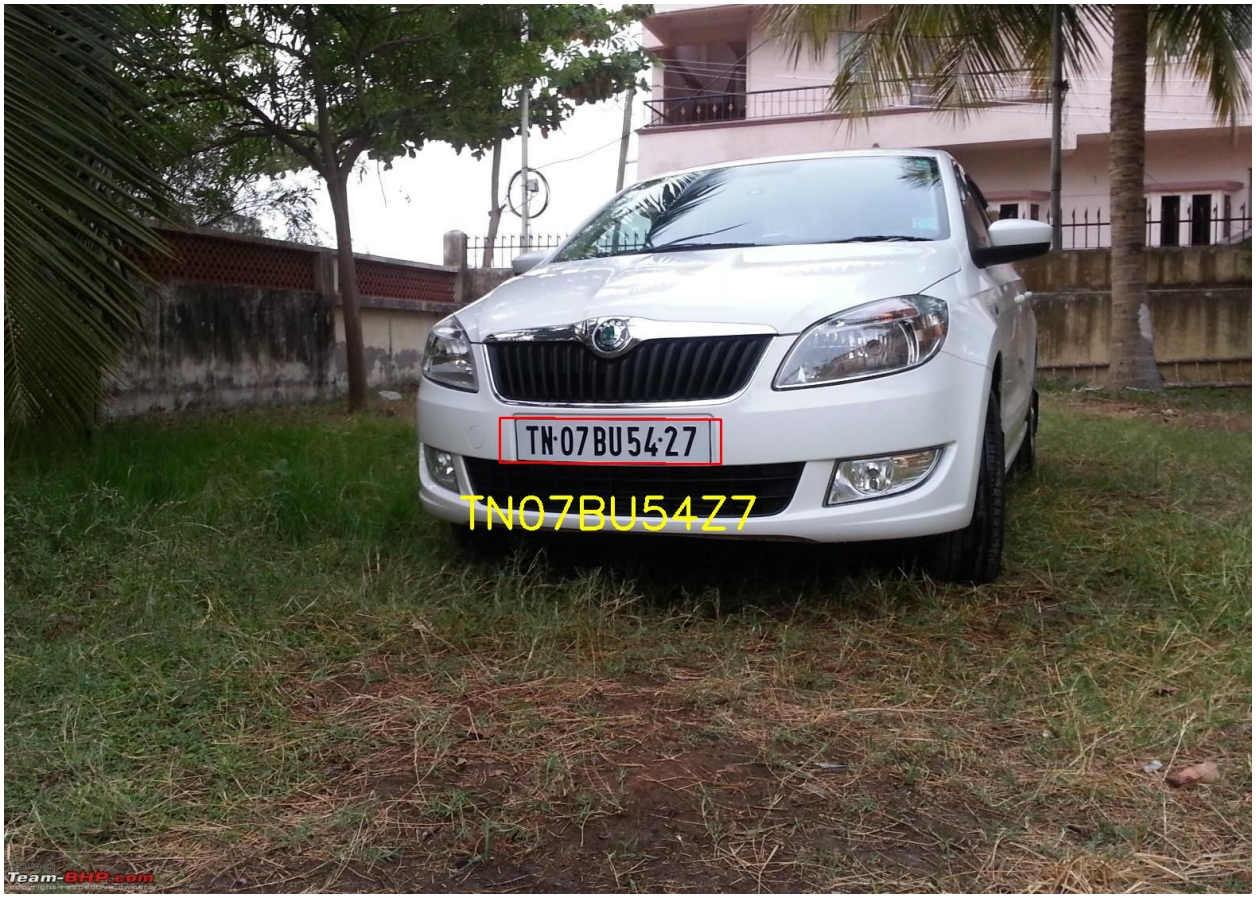
SVM Model 1 example

Completely accurate apart from E->C match. Much more useful model for application.



SVM Model 1 example

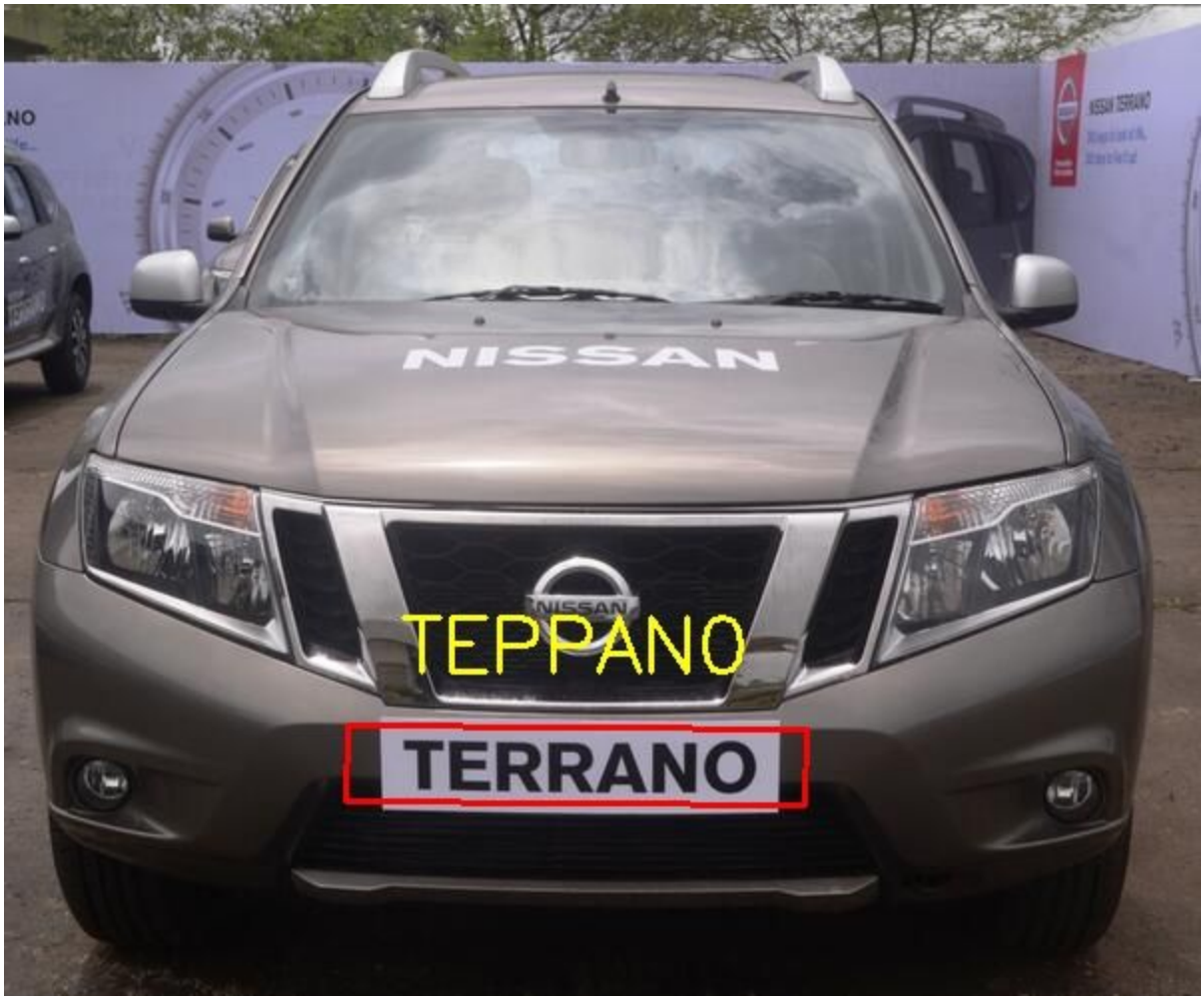
Apart from 2->Z match, it correctly recognizes all the other characters.



SVM Model 1 example

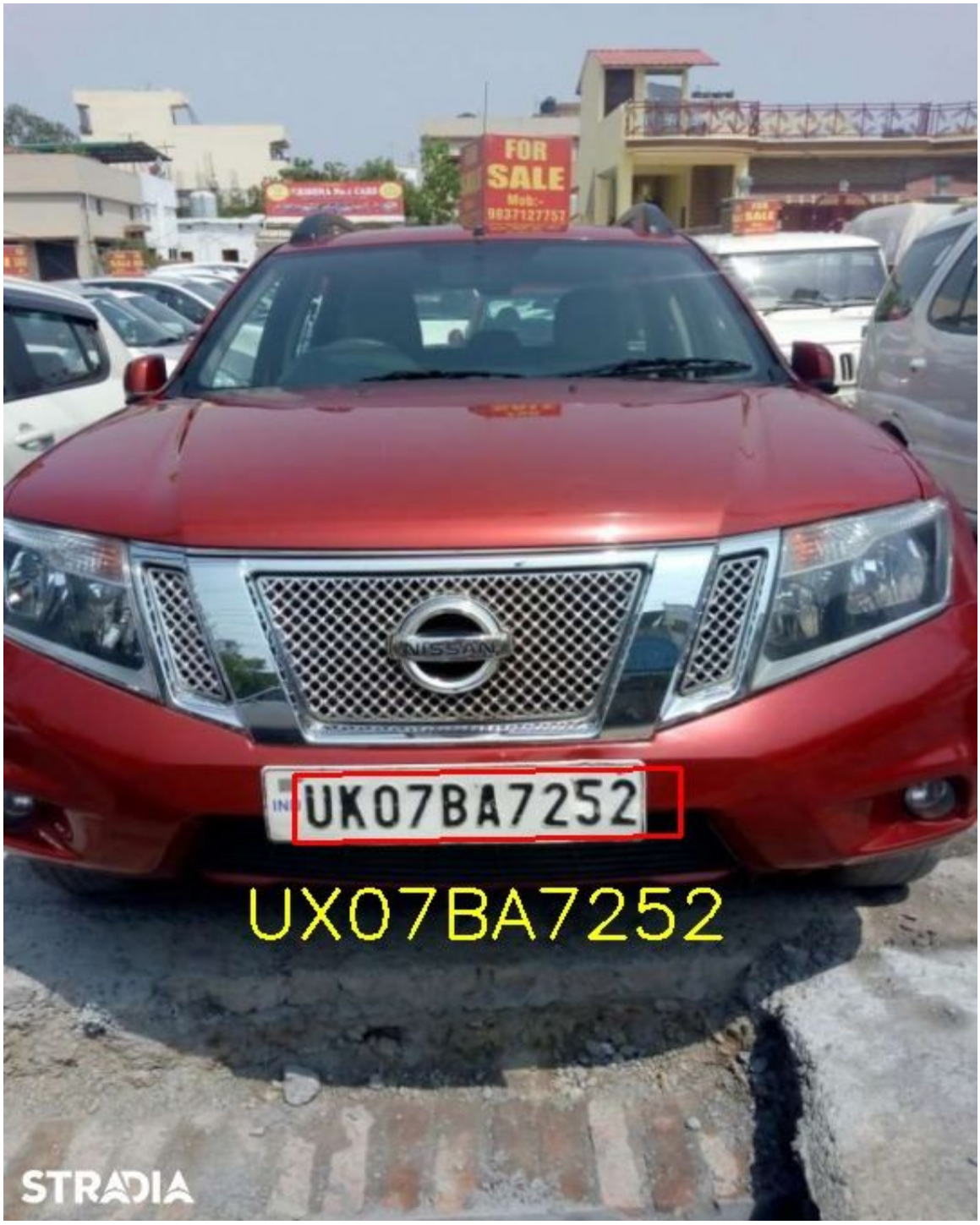
Almost perfect. 2->Z was the only mistake.

KNN Model



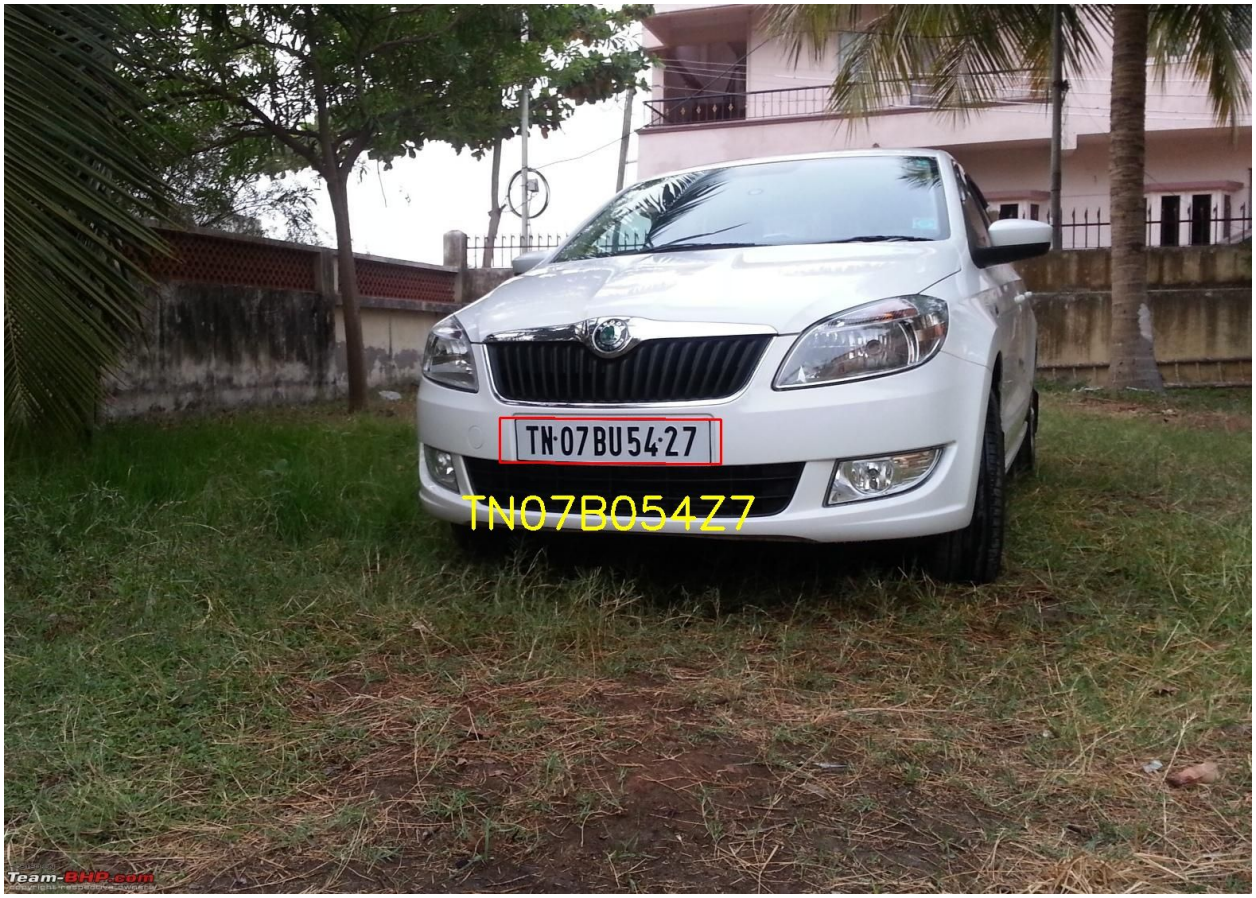
KNN Model example

This model has an accuracy similar to SVM model 1. Only error is R->P.



KNN Model example

K->X is the only error.



KNN Model Example

We find very good accuracy in prediction.

References

1. https://github.com/MicrocontrollersAndMore/OpenCV_3_KNN_Character_Recognition_Python
2. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
3. https://stackabuse.com/pytesseract-simple-python-optical-character-recognition/
4. https://dataturks.com/projects/devika.mishra/Indian_Number_plates
5. https://github.com/femioladeji/License-Plate-Recognition-Nigerian-vehicles
6. https://github.com/MicrocontrollersAndMore/OpenCV_3_License_Plate_Recognition_Python