International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 07 | July 2018 www.irjet.net p-ISSN: 2395-0072

Auto-detection and Masking of Vehicle License Plate Using Machine Learning

Burhanuddin Bharmal, Gopal Thakur, Kshama Tiwari

Abstract - *As per the European laws it is mandatory to mask the license plate of cars and face of humans processing on any images or videos for working on any projects outside the countries. The license plate and face of humans had to be masked manually. In this paper we are going to discuss various algorithms for detection of license plate in a car and automatically masking the plate using image processing. The similar procedures can be used for masking the faces. There are large numbers of dataset available on internet for detection of faces. But dataset for detection of license plate is not available. So here we also discuss how to create a dataset for image processing. For the whole system we have used python 2.7 on Linux operating system. The proposed methodology is designed without the use of GPU so the algorithm is compatible with any machine. In this system first the video is divided into number of images, the car is detected in image using another dataset, then license plate dataset is applied on the detected car, and at the end when the license plate is detected the part of image which is detected as license plate is masked. For masking the license plate the part of image is applied to Low pass filter for blurring the image. The images having tilted or half cut license plate are also masked using the design system.*

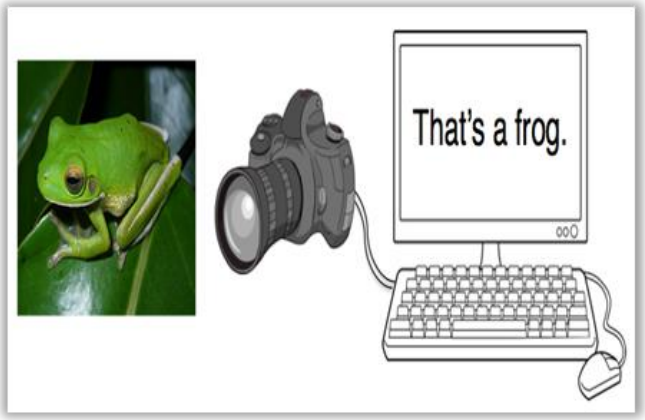
Key Words: Image processing, Histogram of oriented gradient, Car detection, license plate masking, Computer Vision, Python 2.7, OpneCV, dlib library, SVM files, caffe Framework, Imglab, xml file.

1. INTRODUCTION

Artificial intelligence is playing vital role in the automation of various things. We humans are trying hard to reduce the manual work required for any work and trying to bring automation in every field. And for this artificial intelligence and machine learning are playing very important role. This is not the general introduction to artificial intelligence, machine learning or deep learning. In this paper we will be discussing briefly about various algorithms used for image recognition and we will be focusing on histogram of oriented gradient for detection of license plate of car. We will also discuss the method used for masking the license plate. Here we are going to use image filtering for masking the license plate of car. Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

2. Machine learning

Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (i.e., progressively improve performance on a specific



task) with data, without being explicitly programmed. The studies reported here have been concerned with the programming of a digital computer to behave in a way which, if done by human beings or animals, would be described as involving the process of learning [1]. The goal of machine learning is to give computers the ability to do something without being explicitly told how to do it. We just provide some kind of general structure and give the computer the opportunity to learn from experience, similar to how we humans learn from experience too. Rather than writing a huge code for recognition of any object by machine, it's better to train a machine which learns from huge training data set.

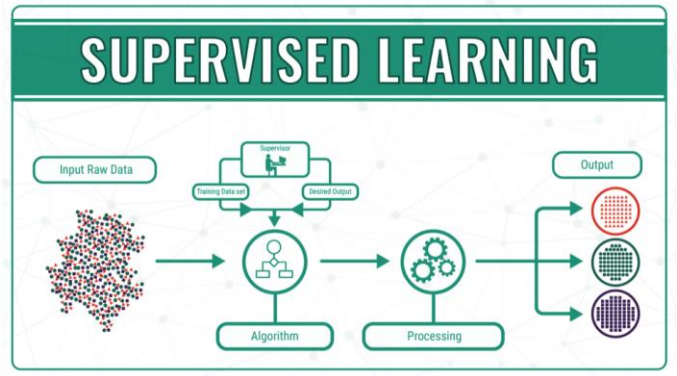
Machine learning comes in many different flavors, depending on the algorithm and its objectives. You can divide machine learning algorithms into three main groups based on their purpose. Machine learning is categorized into supervised, unsupervised and reinforcement learning.[2]

2.1 Supervised learning

- In supervised learning, if we have the training example (**x***i*,*yi*), then it makes sense to construct a predictor *f*(**x**) which will output something close to *yi* when **x** is close to **x***i*.[3]
- Supervised learning is where you have input variables (x) and an output variable (Y) and you use

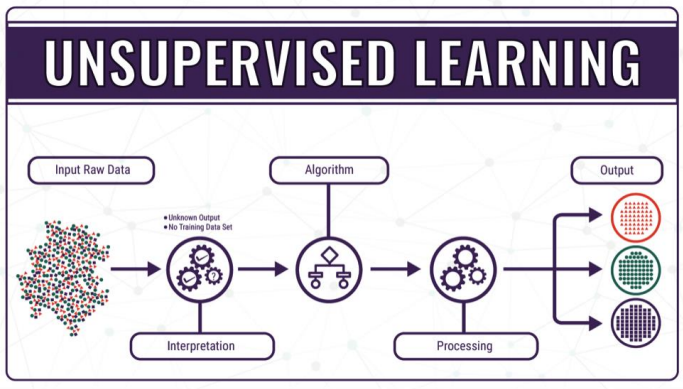
an algorithm to learn the mapping function from the input to the output.

- Y = f(X)
- The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.
- It is called supervised learning because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.



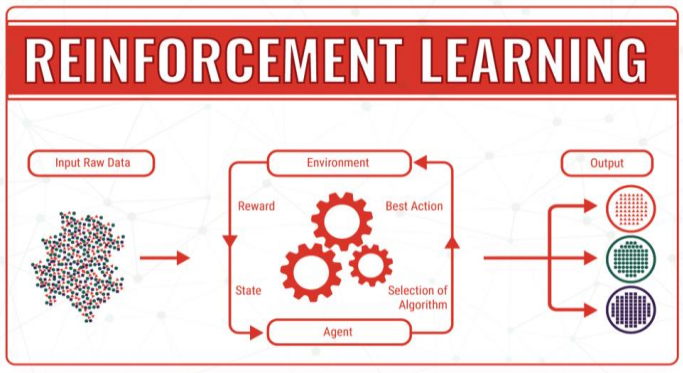
2.2 Unsupervised learning

- Unsupervised learning is where you only have input data (X) and no corresponding output variables.
- The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.
- These are called unsupervised learning because unlike supervised learning above there are no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.
- Unsupervised learning is further classified as clustering and association.



2.3 Reinforcement learning

Reinforcement learning refers to goal-oriented algorithms, which learn how to attain a complex objective (goal) or maximize along a particular dimension over many steps; for example, maximize the points won in a game over many moves. They can start from a blank slate, and under the right conditions they achieve superhuman performance. Like a child incentivized by spankings and candy, these algorithms are penalized when they make the wrong decisions and rewarded when they make the right ones – this is reinforcement.



3. FEATURE EXTRACTION

The input image has too much extra information that is not necessary for classification. Therefore, the first step in image classification is to simplify the image by extracting the important information contained in the image and leaving out the rest.

The main goal of feature extraction is to obtain the most relevant information from the original data and represent that information in a lower dimensionality space. [4]

For example, if you want to find shirt and coat buttons in images, you will notice a significant variation in RGB pixel values. However, by running an edge detector on an image we can simplify the image. You can still easily discern the circular shape of the buttons in these edge images and so we

can conclude that edge detection retains the essential information while throwing away non-essential information. The step is called **feature extraction**.

Turns out we can do much better than simple edge detection and find features that are much more reliable. In our example of shirt and coat buttons, a good feature detector will not only capture the circular shape of the buttons but also information about how buttons are different from other circular objects like car tires.[6]

Some well- known features extraction techniques are Scale invariant feature transform (SIFT), speeded up robust feature (SURF) and Histogram of oriented gradient.

In our project we are going to use Histogram of oriented gradient as it's very easy to implement and it extracts enough features from the image for auto detection of license plate.

3.1 Histogram of Oriented Gradient

A feature extraction technique converts an image of fixed size to a feature vector. HOG is based on the idea that local object appearance can be effectively described by the distribution (histogram) of edge directions (oriented gradients). Histogram of Oriented Gradient (HOG) descriptors is proven to be effective at object detection. [5]

Steps for calculating HOG for an image is:

1. **Gradient calculation**: Calculate the x and the y gradient images, g_x and g_y, from the original image. This can be done by filtering the original image with the following kernels.

-1	0	1

-1
0
1

Using the gradient images g_x and g_y, we can calculate the magnitude and orientation of the gradient using the following equations.

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

The calculated gradients are "unsigned" and therefore θ are in the range 0 to 180 degrees.

2. **Cells**: Divide the image into 8×8 cells.

3. **Calculate histogram of gradients in these 8×8 cells:** At each pixel in an 8×8 cell we know the gradient (magnitude and direction), and therefore we have 64 magnitudes and 64 directions — i.e. 128 numbers. Histogram of these gradients will provide a more useful and compact representation. We will next convert these 128 numbers into a 9-bin histogram i.e. (9 numbers). The bins of the histogram correspond to gradients directions 0, 20, 40 … 160 degrees. Every pixel votes for either one or two bins in the histogram. If the direction of the gradient at a pixel is exactly 0, 20, 40 … or 160 degrees, a vote equal to the magnitude of the gradient is cast by the pixel into the bin. A pixel where the direction of the gradient is not exactly 0, 20, 40 … 160 degrees splits its vote among the two nearest bins based on the distance from the bin. E.g. A pixel where the magnitude of the gradient is 2 and the angle is 20 degrees will vote for the second bin with value 2. On the other hand, a pixel with gradient 2 and angle 30 will vote 1 for both the second bin (corresponding to angle 20) and the third bin (corresponding to angle 40).

4. **Block normalization:** The histogram calculated in the previous step is not very robust to lighting changes. Multiplying image intensities by a constant factor scales the histogram bin values as well. To counter these effects we can normalize the histogram — i.e. think of the histogram as a vector of 9 elements and divide each element by the magnitude of this vector. In the original HOG paper, this normalization is not done over the 8×8 cell that produced the histogram, but over 16×16 blocks. The idea is the same, but now instead of a 9 element vector you have a 36 element vector.

5. **Feature Vector:** In the previous steps we figured out how to calculate histogram over an 8×8 cell and then normalize it over a 16×16 block. To calculate the final feature vector for the entire image, the 16×16 block is moved in steps of 8 (i.e. 50% overlap with the previous block) and the 36 numbers (corresponding to 4 histograms in a 16×16 block) calculated at each step are concatenated to produce the final feature vector. **What is the length of the final vector?**

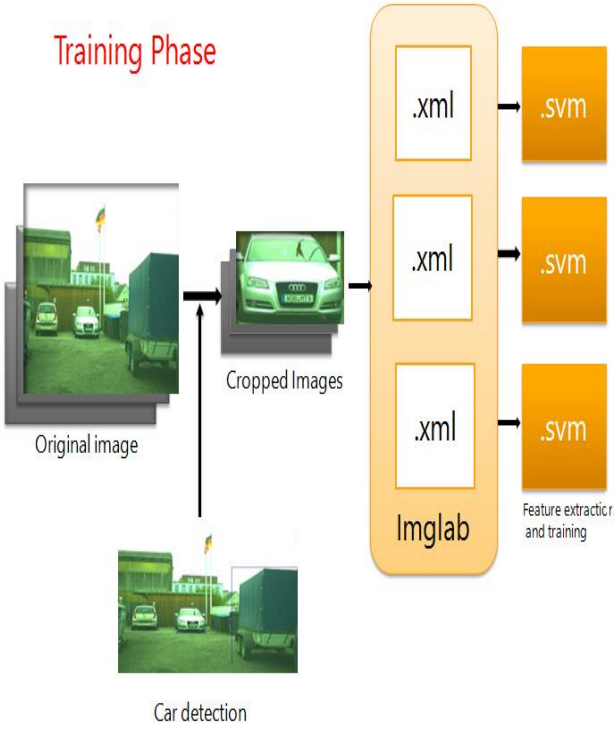
The input image is 64×128 pixels in size, and we are moving 8 pixels at a time. Therefore, we can make 7 steps in the horizontal direction and 15 steps in the vertical direction which adds up to 7 x 15 = 105 steps. At each step we calculated 36 numbers, which makes the length of the final vector 105 x 36 = 3780.

Input Car Number-Plate Image

After applying HOG algorithm

4. License plate Blurring

- In our project the detection of license plate is divided into two phases
- First the system is build to which the image is passed the output of the system is image on which the license plate is detected and blurs the license plate.

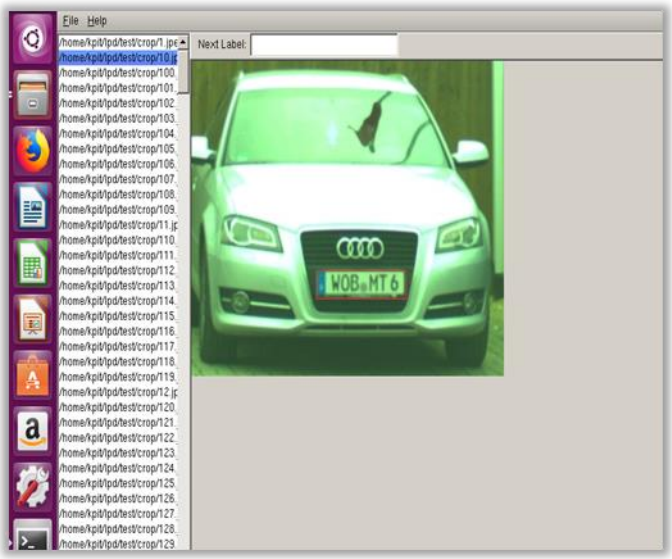


- For building the system, the system has to be trained with number of images of license plate.
- First the original images are cropped to car images before training.

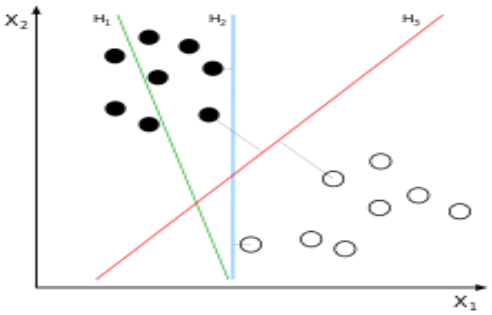
Original Image

Cropped Image

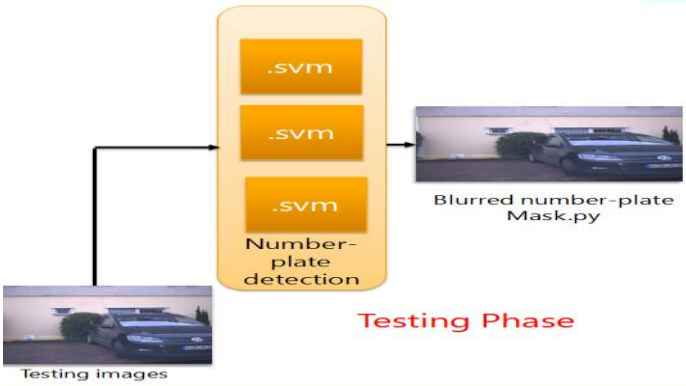
- Then using imglab tool in dlib library of python the training images are converted into xml file.
- The xml files contain the feature data and background data.
- The data which is annotated is the feature data which we have to recognize in the original image.
-



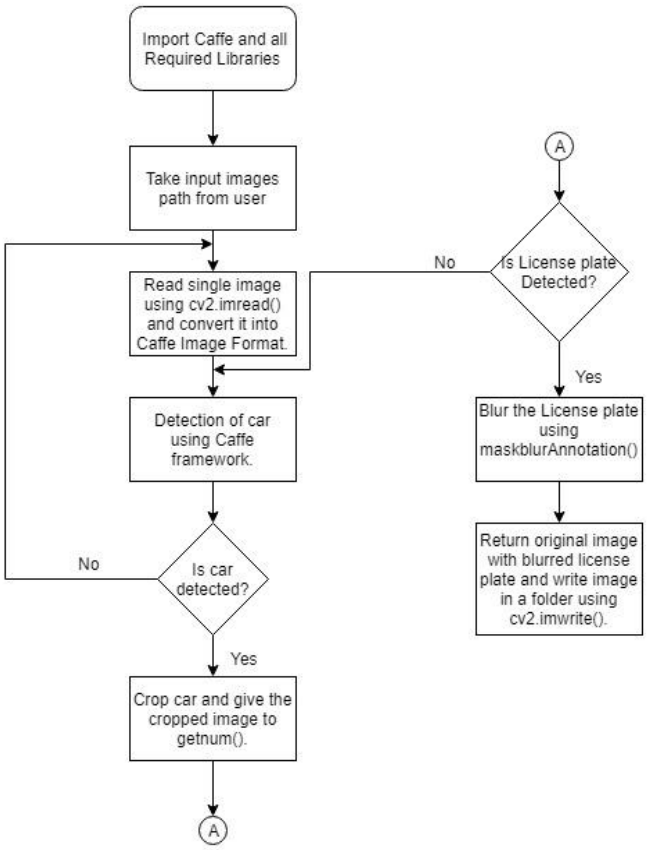
In machine learning, support vector machines (**SVMs**, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.



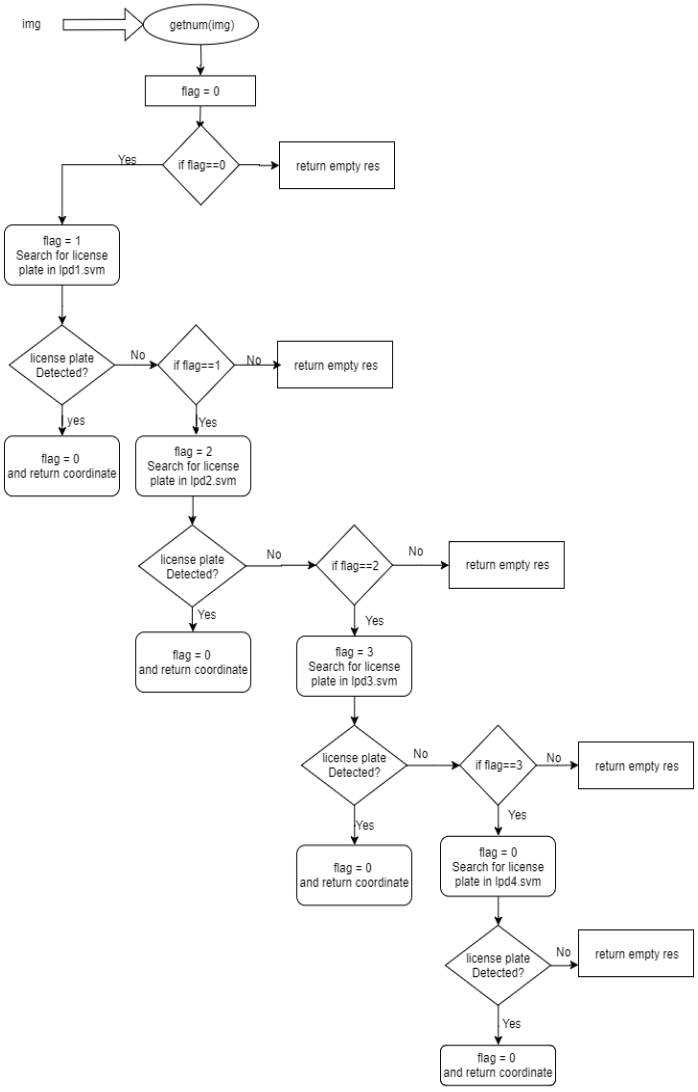
- Then the xml file is converted to svm file using below procedure
- For converting xml file to SVM classifier we are using dlib function.
- **dlib.train_simple_object_detector(*(*str)dataset_filename, *(*str)detector_output_filename, *(*simple_object_detector_training_options)options)**
- **Dataset :-** input file is the xml file in which we have labelled the license plate in car.
- **Detector output file** :- is the output SVM classifier which we are going to use for detection of license plate in car.
- **dlib.simple_object_detector_training_options**
- This object is a container for the options to the train_simple_object_detector() routine.
- It has options:
- **C** :– It is regularization parameter, It should be greater than zero.
- **add_left_right_image_flips:**-if true , train_simple_object_detector() will assume the objects are left/right symmetric and add in left right flips of the training images. This doubles the size of the training dataset.
- **be_verbose**:- If true, train_simple_object_detector() will print out a lot of information to the screen while training
- In our project we have created four SVM files, to increase the accuracy of detection of license plate. Each SVM file is created using different set of training images covering the maximum angles for license plate.



- The second phase of the project is about testing the system
- In second phase, the images are given as input to the system.
- Computer vision is used for read , writing of image.
- The image of car is read using cv2.imread function of open cv.
- The image is converted into caffe framework image format.
- Using caffe framework the car is detected in the image.
- The cropped image of car is then passed to another function for detection license plate.



- The cropped image is first checked with first SVM, if the license plate is found then it is given to another function for blurring.
- If the license plate is not found using first SVM then the system continues to search license plate in different SVM.
- Four SVM is use to cover maximum angles of license plate images and increase the accuracy of detection of license plate.



- When the license plate is detected the coordinates are passed to another function for blurring the part of image.
- We are using cv2 2d filter for blurring the image. The part of image to be blurred is passed through the filter number of times for blurring the image
- The image with blurred license plate is given as the output of the system.

- The output image is saved in the output folder using cv2.imwrite function.

Output image with license plate blur

5. CONCLUSION

This image recognition technique can successfully detect license plate in the image. The false detection of the license plate is also avoided as first car is detected in the image using caffe framework. The detection of license plate using this technique has achieved 86.95% accuracy. This accuracy can be achieved with just for 2 hours of learning phase. The system is requires simple 16Gb ram machine with linux os. The system can also be used for detection of different objects such as car, faces. For detection of different objects we have to train the system with images accordingly.

6. REFERENCES

[1] A. L. Samuel, "SOME STUDIES IN MACHINE LEARNING USING THE GAME OF CHECKERS", IBM journal.

[2] Sunpreet Kaur, Sonika Jindal, "A SURVEY ON MACHINE LEARNING ALGORITHM",International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2763 Issue 11, Volume 3 (November 2016)

[3] Yoshua Bengio, "LEARNING DEEP ARCHITECTURE FOR AI", Foundations and Trends in Machine Learning Vol. 2, No. 1 (2009) 1–127_c 2009 Y. Bengio DOI: 10.1561/2200000006

[4] Gaurav Kumar, Pradeep Kumar Bhatia, "A DETAILED REVIEW OF FEATURE EXTRACTION IN IMAGE PROCESSING SYSTEM", 2014 Fourth International Conference on Advanced Computing & Communication Technologies

[5] Maesen Churchill, Adela Fedor, "HISTOGRAM OF OREINETED FOR DETECTION OF MULTIPLE SCENE PROPERTIES".

[6] https://www.learnopencv.com/image-recognition-and-object-detection-part1/

[7] http://deeplearning.net/tutorial/

[8] John wright, Allen Yang, Arvind Ganesh, S. Shankar Sastry, Yi Ma, "ROBUST FACE RECOGNITION VIA SPARE REPRESEBTATION", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 31, NO. 2, FEBRUARY 2009.

[9] http://yeephycho.github.io/2016/08/15/image-data-in-tensorflow/

10] http://www.wolfib.com/Image-Recognition-Intro-Part-1/

[11] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva, "LEARNING DEEP FEATURES FOR SCENE RECOGNITION USING PLACES DATABASE".

[12] Fabrizio Sebastiani, "MACHINE LEARNING IN AUTOMATED TEXT CATEGORIZATION", ACM Computing Surveys, Vol. 34, No. 1, March 2002, pp. 1–47.

[13] Pedro Domingos, "A FEW USEFUL THINGS TO KNOW ABOUT MACHINE LEARNING".

[14] Yuqian Li, Guangda Su, "SIMPLIFIED HISTOGRAM OF ORIENTED GRADIENT FEATURES EXTRACTION ALGORITHM FOR THE HARDWARE IMPLEMNETATION.", 2015 International Conference on Computers, Communications, and Systems (ICCCS)

[15] https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[16]https://docs.opencv.org/3.0beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[17]http://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/