# A PROCESS FOR COMPLETE AUTONOMOUS SOFTWARE DISPLAY VALIDATION AND TESTING (USING A CAR-CLUSTER)

Malobika Roy Choudhury

Innovation and Technology, SAP Labs India Pvt Lmt.,
Bengaluru, Karnataka, India

***ABSTRACT***

*Every product industry goes through the process of product validation before its release. Validation could be effortless or laborious depending upon the process. Here in this paper, a process is defined that can make the task-independent of constant monitoring. This method will not only make the work of test engineers easier it will also help the company meet stringent release deadlines with ease. The method explores how to complete visual validation of the display screen using deep learning and image processing. In the example, a method is discussed wrt a car-cluster display screen. The method breaks down the components of the screen then validates each component against its design and outputs a result predicting whether the displayed content is correct or incorrect. The models like You-Only-Live-Once, Machine Learning, Convolution Neural Networks-Conv2D, and image processing techniques like Hough circle/Hough lines are used to predict the accuracy of each display component. These sets of algorithms compile to provide consistent results throughout and are being currently used to generate results for the validation process.*

***KEYWORDS***

*Convolution Neural Networks, You-Only-Live-Once, display-validation.*

## 1. INTRODUCTION

Before a product is released to the market it undergoes a lot of cycles. From development to release it goes through a massive amount of testing and validation. Most of the validation and testing is usually accomplished by test engineers who try and make sure that the final product is market-ready and follows Lean Software Principles. But to achieve this task they must spend hours of their time into visually validating the product. This paper aims to provide a method that can help reduce this time, help reduce last-minute defects (helps reduce cost, saves reputation), and make testing truly Automated. Several methods have been attempted towards this area but seldom discussed in the public forum as it advances to be propriety. In the behavior Driven testing field methods of hard coding are used to test a feature. Several tools are available in the market which are preferred like Test.ai, Testim.io, and other APIs that help in Automation testing. Although most companies have confidential data and tend to not use the available tools for validation, they instead have their tools. Another method deployed is where the validation is outsourced, and the client company provides expensive solutions. This paper wants fully automated visual testing accessible by all.

Dhinaharan Nagamalai et al. (Eds): CSEIT, WiMoNe, NCS, CIoT, CMLA, DMSE, NLPD - 2020
pp. 131-139, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.101112

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 248, 248, 32) | 896 |
| max_pooling2d_1 (MaxPooling2 | (None, 124, 124, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 122, 122, 64) | 18496 |
| max_pooling2d_2 (MaxPooling2) | (None, 61, 61, 64) | 0 |
| conv2d_3 (Conv2D) | (None, 59, 59, 128) | 73856 |
| max_pooling2d_3 (MaxPooling2) | (None, 29, 29, 128) | 0 |
| conv2d_4 (Conv2D) | (None, 27, 27, 128) | 147584 |
| max_pooling2d_4 (MaxPooling2) | (None, 13, 13, 128) | 0 |
| flatten_1 (Flatten) | (None, 21632) | 0 |
| dense_1 (Dense) | (None, 512) | 11076096 |
| dense_2 (Dense) | (None, 1) | 513 |

Total params: 11,317,441
Trainable params: 11,317,441
Non-trainable params: 0

Saved model to disk

```
With accuracy and loss as:

Epoch 1/5
20/20 [==============================] - 27s 1s/step - loss: 0.4907 - acc: 0.7325 - val_loss: 0.2082 - val_acc: 1.0000
Epoch 2/5
20/20 [==============================] - 26s 1s/step - loss: 0.1795 - acc: 0.9375 - val_loss: 0.0526 - val_acc: 1.0000
Epoch 3/5
20/20 [==============================] - 26s 1s/step - loss: 0.0576 - acc: 0.9850 - val_loss: 0.0056 - val_acc: 1.0000
Epoch 4/5
20/20 [==============================] - 27s 1s/step - loss: 0.0108 - acc: 1.0000 - val_loss: 0.0011 - val_acc: 1.0000
Epoch 5/5
20/20 [==============================] - 26s 1s/step - loss: 0.0011 - acc: 1.0000 - val_loss: 3.6085e-04 - val_acc: 1.0000

test acc: 1.0
```

Figures 1.1 ,1.2 System CNN-Conv2D Model parameters and training statistics

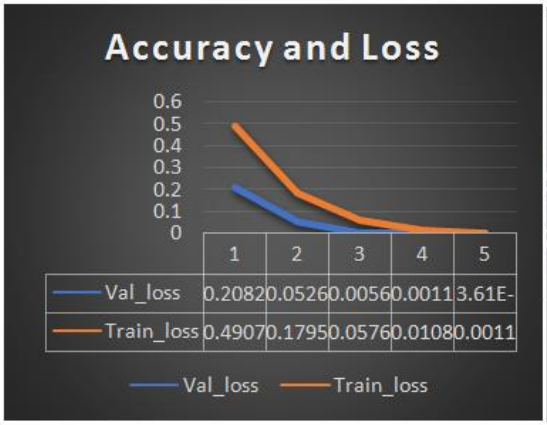



Figure 1.3 Data plotted using results from Figure 1.1 and 1.2.
Figure 1.4 representing an image of car cluster, Such images were used as starting point for YOLO(Image obtained from Reference [11])

During cluster visual validation the simulation of signals for various ECU's is accomplished using some Can-based tool. This tool stimulates RX/TX messages from other subsystems. Not only must messages be validated but also their effect must be studied. This process might involve a few indicators to turn on, speed to change in the speedometer, or some warning being displayed. Each of these is divided into components for regression testing. YOLO is used to divide these components into four major regions of interest. Depending upon the component either positional value approximation or further processing is done. These regions of interest are further passed through convolution [2] models to check for event accuracy. This helps satisfy all the principles of automated testing-helps reduce time, improves efficiency, saves money, and meets deadlines with ease.

The first section discusses the work being done in the respective fields, second describes the algorithm and how it works. The third section describes the dataset creation and accuracy of the model being used. The fourth section discusses the results obtained and explores the challenges and future work.

## 2. PREVIOUS WORK

The image grabber [2] is mentioned as a tool for validation testing for the Instrument panel cluster. The paper discusses in detail the method of capturing screenshots. It also focuses on the process of saving the screenshots using a frontend. This paper focuses on the validation of screens using video streaming and saving frames. YOLOv is applied over frames of video footage for continuous testing. Another work [3] have used frame by frame comparison, using MSE (Mean Squared Error) and SSIM

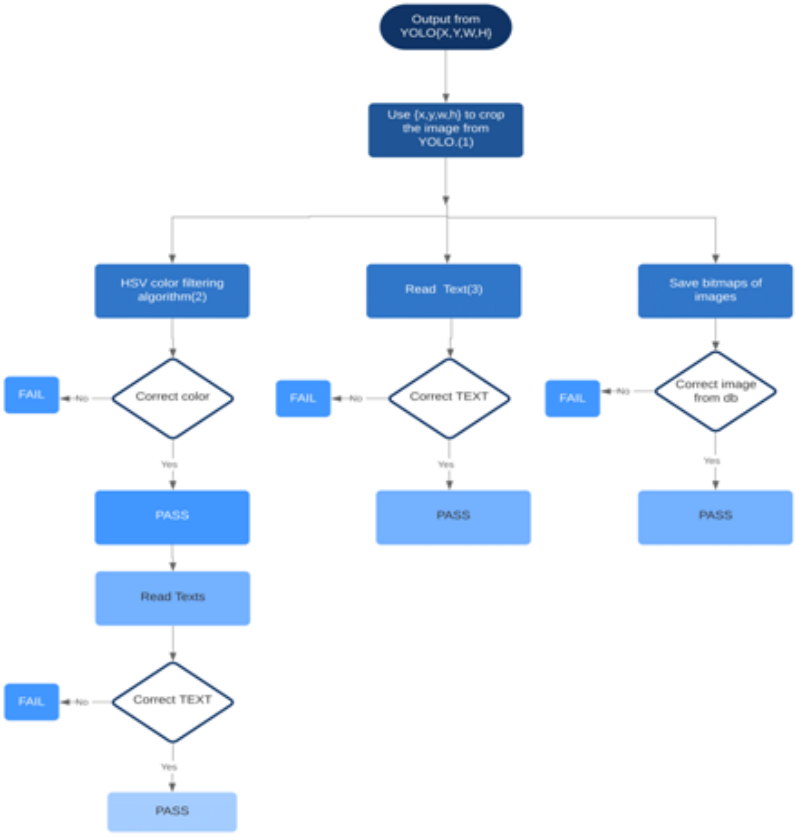


Figure 2.1 Representing the flowchart or system flow

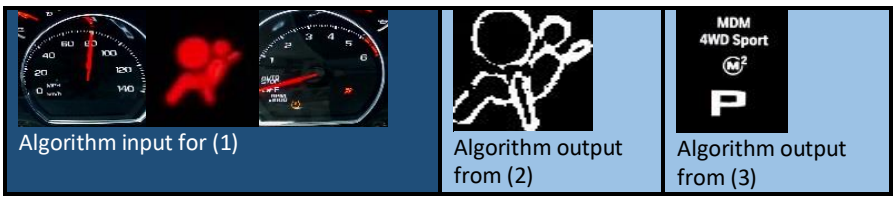


Figure 2.2 Results from algorithm before decision stage

the parameters to extract the similarities between two images. The reference image is compared to the live-stream image, if it fails and exceeds the required MSE, the system will output the test case as failed. Yet, the most seeming drawback with this method is if the brightness or the contrast of room changes the MSE can vary drastically as it is based upon pixel values. Visteon

has come close with [4] SIFT in segregating and identifying its text-based regions from images. Before applying YOLO several methods experimented like SIFT or M-SURF, but it resulted in using extreme amounts of computation power. That is why it was discarded. These algorithms can provide good outputs with a high-powered GPU. Most research papers talk about automating the manual signals to automated signals. The major player

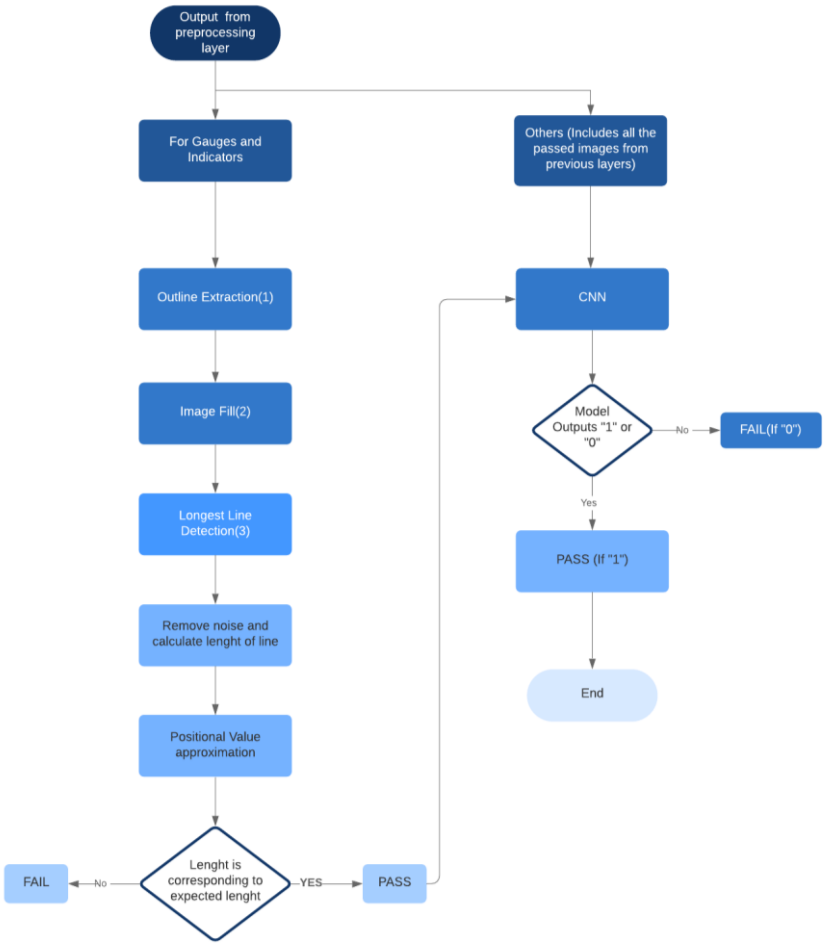


Figure 3.1 Flowchart for Image Processing layer

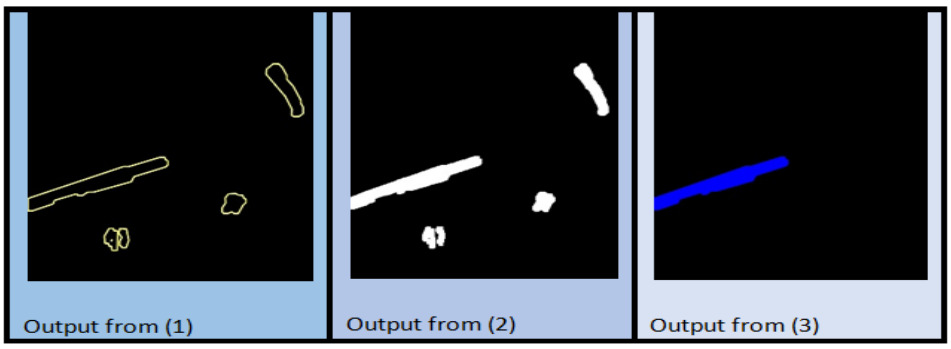


Figure 3.2 Outputs from the image processing layer

in the field is selenium [5], using selenium one can automate the test cases and test for each component. It provides extensive and rich user Interfaces and comfortable experience for testers. But most Fortune 500 companies already have automated test cases, only problem persistence is visual validation and while most companies try to patent their method and have unique ways, challenges remain with how a few algorithms are guaranteed to give results generically.

# 3. A PROPOSED SOLUTION

## 3.1. System Model

The Automation testing can be divided into three major divisions which can be used as three threads interacting with each other based upon inputs and outputs. 1. The Image processing component: This component involves cropping the image, finding contours, extracting bounding boxes, the longest line, or circles from an image. In the method explained OpenCV is used to contour out the regions of interest. These regions of interest are then further used for processing. YOLO is used to extract a box of the desired size from Figure 1.4. Masking techniques are used to get the contour of the needle of the speedometer [Figure 3.1]. As one can see from Figure 2.2 that the needle is red so after masking out red color using HSV [6] filter a mask is applied to extract the image [ Figure 3.2]. Similarly, in a generic car-cluster, there are circular regions to extract, open cv libraries are used to extract Hough circles from image along with Hough lines. This component varies drastically from cluster to cluster. 2. The CNN-Conv2D component. The cluster contains some constant indicator images, telltales, and warnings. These can be verified by using a convolution neural network model either specific to the image or a classification model to indicate the category. CNN-Conv2D comes at the bottom of the architecture because the final prediction of the image or final validation is completed by CNN-Conv2D. Results are interpreted as one or zero. 3. The YOLO [7] component is used to separate the regions in the cluster, this, in turn, helps provide inputs for CNN-Conv2D models.

## 3.2. Algorithm

### 3.2.1. CNN-Conv2D Algorithm

As Referenced in [1] we define a convolution process by assuming a 4-D kernel tensor K with element $K_{i,j,k,l}$ giving the connection strength between a unit in channel i of the output and a unit in channel j of the input, with an offset of k rows and l columns between the output unit and the input unit. Assume the input consists of observed data V with element $V_{i,j,k}$ giving the value of the input unit within channel i at row j and column k. Assume the output consists of Z with the same format as V. If Z is produced by convolving K across V without flipping K , then

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n} \quad \ldots(1)$$

$$Z_{i,j,k} = c(K,V,s)_{i,j,k} = \sum_{l,m,n} V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n} \quad \ldots(2)$$

Equation (1) and equation (2) help explain the Convolution process as referred to in [1]. This paper does not explore the mathematical backend of the CNN-Conv2D [8] algorithm, it considers the implementation of CNN-Conv2D in detail. Once images from the cluster are obtained either through camera or any other device which requires various CNN-Conv2D models in the background to explain the validity of each image. For this .h5 is used and .json file is used with weights stored from the trained model.

An alternate method explored is to classify between images and map them according to the triggered event. Though it required extreme amounts of computation power. Hence, switching back to event-triggered validation, the algorithm checked for incorrect images on a one-by-one basis. It proved successful in saving time and generating results with lesser use of resources.

Once an event is triggered a screenshot is captured, of the cluster. According to the respective categories images go through various layers of the algorithm and generate results. On the other hand, training models for each of the event was laborious until image processing was used. By using image processing a dataset of about 200*1000 images was generated. Each model used 4 convolution layers and 2 fully connected layers. As seen in Figures 1.1 and 1.2, the model is trained in Anaconda IDE Spyder where the console is displaying the Convolution layers, accuracy, and loss values. Figure 1.3 is the graphical representation of accuracy and loss values through progressive epochs. Figure 1.4 is the input image for the YOLO layer which again uses the CNN algorithm.

CNN-Conv2D has been trained on the images of the correct text-based alert or graphics-based alert which included 100 images for each alert. These images were collected from the live-stream under various light conditions. This was done to prevent the failure of the test cases. The models are trained to be robust and adaptive to any light conditions. Be it day or night the validation process could be carried out at any time.

### 3.2.2. Image processing component

Image processing is used extensively in computer vision to aid the machine learning algorithms. Figure 2.2 is the input for the image processing layer, but only the images having a needle are used in this layer. They not only help in pre-processing but can help extensively in making the dataset more diverse. Various algorithms discussed are extracting the longest line and the largest circle by using Hough Transform [9].

$$r = x\cos\theta + y\sin\theta \quad \text{...(3)}$$

Using Hesse's transform [9], the longest line is obtained in an image where computations are done for pixel values. When the threshold is applied over the values obtained from transform, segregation of different circles and lines helped obtain the required circle.

Similarly, other algorithms can be applying to extract the image. For example: - if the speed of the vehicle is to be calculated from the scale in the speedometer, after extracting the color-filled longest line one can find the speed by calculating $\theta$ and $r$ . Therefore, obtaining the result from the equation below:

$$l = r * \theta \quad \text{... (4)}$$

Another example -Optical Character Recognition [10] can be deployed for checking the correctness of each alphabet or number for each event.

### 3.2.3. YOLO component

The YOLO [7] is a CNN-Conv2D network that learns to identify objects. Primarily used for object detection, it's capabilities can be used to obtain Regions of Interest (ROI) from the frame. Each ROI acts as an object for the algorithm. A grid of 9*9 for a 960*800 image is used. YOLO was trained to identify the speedometer, gauges, middle region for warnings, and upper region for

indicators. This model can be modifying to generate other output. The vector values for each grid object must be updated to obtain variable results.

## 4. SIGNAL-FLOW

Input frames from the video stream are fed as individual images (Figure 4.1) to the YOLO network. YOLO outputs are a set of ROI's (Figure 4.2) which are fed to the image processing layer but only in conditions where further processing is required like:

- To find the angle of gauge needle (Figure 4.2 Right ROI)
- To find the angle of the speedometer needle (Figure 4.2 Left ROI)

From these angles, the RPM and the speed are calculated, as the radius and length are constant.

- In the case of warnings /text-based alerts validation (Figure 4.2 Center ROI), OCR is used.
- For indicators HSV-based-color-filtering as referred to in [6], open CV models are used to validate the color, but for shapes, CNN-Conv2D is used.

All the components of validation are covered once the above categories are completed.

### 4.1. Training of YOLO network

It is accomplished by feeding the network with 500 images of each of the ROI's. A dataset this large can be cumbersome to build. Hence OpenCV libraries were used to increase contrast, decrease brightness, and include noise to obtain a diverse dataset. During testing of the model unseen images were fed to the network to distinguish the regions. It was only trained initially to output the regions, later updated to output the width, height, x, y-axis of the bounding box for the ROI.

Dimensions obtained were then utilized to crop the regions and fed to the subsequent layers. The complete process is depicted using a flowchart in Figure 2.1. After the images have been fed to CNN Layer the result ("0" for PASS and "1" for FAIL) will conclude if it is "PASS TESTCASE" or "FAILED TESTCASE" which in turn will conclude the process.

## 5. RESULTS

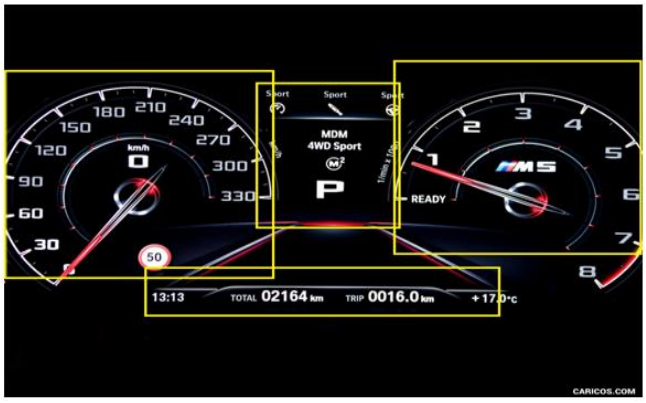


Figure 4.1 Bounding boxes from YOLO, Obtained from Reference [9]

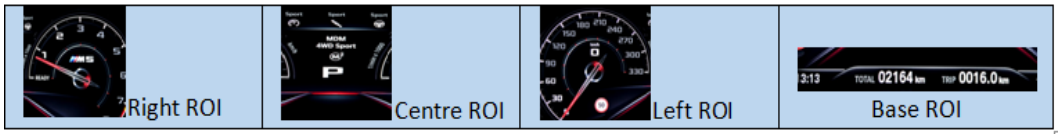


Figure 4.2 ROI's from YOLO/ Output from YOLO

While taking into account the accuracy obtained from the model whereas it was taking 16-17 minutes through manual testing. So, the whole validation process was completed within 5 hours. The entire cost was reduced by hundreds of dollars and efficiency was increased. The tool thus developed was able to generate output at 98% (average from all the models) accuracy overall. The dataset has been tested on the frames of the video stream for car-cluster with different speedometer and gauge values. The 2% where the tool failed was in areas like Optical Character recognition. OCR was not able to identify a 4WD and gave output as AWD. Another challenge was colors; hence a fixed brightness and contrast was set from the camera to detect without errors. The report was generated automatically from the outputs and it completed the validation process. Throughout the process Logitech webcam was used as a camera and a Windows/Linux PC with core Xeon without any external GPU has been used. The solution checked all the parameters of automated testing reduced time, efforts, saved cost, and improved efficiency (Qualitatively and Quantitatively). The various process has claimed to obtain good results, but the solution is validation situation oriented. The methods which can work for one cannot be suitable for the other as the conditions differ drastically with processing power, lighting conditions, and use of various interfaces. Sometimes complete autonomous automation is not the goal, although in cases it is required the paper extensively helps in building solutions with remarkable accuracies.

## 6. STIMULATION

The method is currently deployed in the test environment to test its reproducibility. During its thousand's run, it has not produced any false positives which are crucial to production. The 2% corresponds to the false negatives which can be rechecked if needed by the test engineer. The low brightness and high contrast are being fixed for the test to aid OCR.

Other competing solutions are usually not in-house, the cost, development, as well as maintenance, is expensive. This process gives more power to the user and reduces dependency on third-party software. Moreover, with open-source libraries available it becomes easy to deploy it over a variety of systems under validation.

## 7. CONCLUSIONS

The methods discussed if executed sequentially will help in converting semi-Autonomous testing to fully autonomous testing. Semi-autonomous testing involves automating all signals and messages while the decision is taken by a human. This task is not required in fully autonomous testing where the decision is taken by the program. There can be ways to optimize the flow of algorithms and improve efficiency to 100%, which may require training an in-house OCR. Moreover, process improvement can be achieved by optimizing algorithms and image processing over the video frame.

## ACKNOWLEDGMENTS

I would like to acknowledge and thank the python community in GitHub and other forums. Also, the open-source community of anaconda IDE without whom the paper would not be successful.

## REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016,chapter-9,p. 332.
[2] Nimara, Sergiu & Popa, Dorina & Bogdan, Razvan. (2017). Automotive instrument cluster screen content validation. 1-4. 10.1109/TELFOR.2017.8249425.
[3] Raj, Mohan & Kumar, Sathiesh. (2017). Vision based feature diagnosis for automobile instrument cluster using machine learning. 1-5. 10.1109/ICSCN.2017.8085671.
[4] Deepan Raj M, Prabu A. (2019) [PDF],Available:https://www.visteon.com/wp-content/uploads/2019/01/multilingual-string-verification-for-automotive-instrument-cluster-using-artificial-intelligence.pdf [Accessed:4-May-2020]
[5] Selenium Automation Testing[Online],Available: https://www.guru99.com/introduction-to-selenium.html[Accessed:4-May-2020]
[6] Harrison,Color Filtering OpenCV Python Tutorial[Online],Available:https://pythonprogramming.net/color-filter-python-opencv-tutorial/ [Accessed:5-May-2020]
[7] G. Nishad, You Only Look Once(YOLO): Implementing YOLO in less than 30 lines of Python Code [Online]Available:https://towardsdatascience.com/you-only-look-once-yolo-implementing-yolo-in-less-than-30-lines-of-python-code-97fb9835bfd2[Accessed: 1-May-2020]
[8] E.Allibhai, Building a Convolutional Neural Network (CNN-Conv2D) in Keras[Online].Available:https://towardsdatascience.com/building-a-convolutional-neural-network-CNN-in-keras-329fbbadc5f5[Accessed: 1-May-2020]
[9] OpenCV Documentation, Hesse's Transform[Online],Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html[Accessed:4-May-2020]
[10] Optical Character Recognition(OCR) Python documentation, [Online],Available:https://pypi.org/project/pytesseract/[Accessed: 5-May-2020]
[11] Caricos. (2019). 2019 BMW M5 COMPETITION [Online image]. Retrieved October 2, 2019, from https://www.caricos.com/cars/b/bmw/2019_bmw_m5_competition/images/140.html

## AUTHOR

I am currently working as a backend engineer at SAP Labs India, mostly I work on application development, but deep learning is one of my interests. This is one of the papers regarding my previous work. I hope you enjoy it.