International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 13 (2021) pp. 182-191

Submitted: 20 April, 2021; Accepted: 26 May, 2021; Publish: 6 August 2021

TSCTNet: Traffic Signal and Countdown Timer Detection Network for Autonomous Vehicles

Dhananjai Chand[1], Savyasachi Gupta[2], and Ilaiah Kavati[3]

[1]Department of Computer Science and Engineering,
National Institute of Technology Warangal, Telangana, India
dcdevdhan@gmail.com

[2]Department of Computer Science and Engineering,
National Institute of Technology Warangal, Telangana, India
gsavya10@gmail.com

[3]Department of Computer Science and Engineering,
National Institute of Technology Warangal, Telangana, India
ilaiahkavati@nitw.ac.in

Abstract: **The revolution in the field of autonomous vehicles and driver assistance systems has brought with it the need for the development of robust support systems such as traffic signal and countdown timer detection in order to facilitate faster and safer real-world adoption. Various deep learning models and computer vision techniques have extensively been applied to develop systems for the detection of traffic signals. However, the task of detecting an auxiliary timer value along with a traffic signal has relatively been unexplored. In this article, a novel framework, TSCTNet (Traffic Signal and Countdown Timer Detection Network), is proposed that leverages state-of-the-art Mask R-CNN model for object detection followed by efficient image processing techniques for detecting a traffic signal. Subsequently, the detected signals are used to extract a local sub-image, which is then processed and fed to a RetinaNet model for detecting the countdown timer value. The framework has been evaluated on a custom dataset (CFTS) of dash-cam videos and achieves a high average precision value for detecting the traffic signal as well as the countdown timer value. The proposed system was found to be performant in various scenarios and can assist in paving the way for the advancement of autonomous vehicle infrastructure.**

Keywords: RetinaNet, Mask R-CNN, Computer Vision, Autonomous Vehicle, Object Detection, Color Transformation

I. Introduction

Traffic signals are used all over the world to control the right-of-way at intersections, thus facilitating the safe movement of vehicles on roads. Two major classifications of traffic signals based on the underlying mechanism are actuated (automatic) signals and fixed time signals. While actuated signals direct the flow of vehicular movement based on the concentration of traffic via the use of sensors and other devices, fixed time signals are set to repeat periodically a cycle of red, yellow, and green lights. Fixed time signals use traffic signal countdown timers, which are devices that give information about the time remaining until the traffic signal changes color.

Although there are multiple traffic signal control systems which are more efficient [1, 2], there are certain advantages of fixed timer signals that have made them popular in Asian countries, such as India [3]. Fixed time signals help save fuel, smooth traffic flow, reduce traffic violation, and lower driver stress [4]. Moreover, autonomous vehicles can make better and safer traffic decisions based on feedback on the amount of time remaining for the light to switch colors [5]. Therefore, an autonomous vehicle must be able to identify and process not only the traffic light but also the traffic signal countdown timer before it is made practically available for Asian countries.

II. Related Work

Even though the domain of autonomous vehicles has experienced a tremendous amount of research focus, with the detection of traffic lights being one of the fundamentals, research in traffic signal countdown timer detection has been relatively dormant. The reason for this can be attributed to the use of sensor-based traffic signals in western countries as only a dozen Asian countries use fixed-timer traffic lights.

Previous works have been primarily focused on traffic light detection [6]. Classical traffic light detection models are based on image processing and there have been multiple publications on traffic signal detection from image color [7, 8]. Certain techniques involve using the shape of the traffic light for predicting traffic light color [9, 10]. However, these techniques fail to locate the exact location of all traffic lights present.

Custom CNNs have been created for the sole purpose of

traffic light detection. Bach *et al.* [11] performed deep convolutional neural network based semantic segmentation for extracting traffic light color. Furthermore, DeepTLR [12] also performs semantic segmentation where separate bounding boxes are predicted for each differently colored traffic light bulb. Gao & Wang [13] proposed a hybrid approach using classical and self-learning algorithms for detecting traffic light color. Müller & Dietmayer [14] used an augmented form of 'single-shot detection' for detecting traffic lights. Although these techniques give good results, they do not fulfill the requirements of the proposed overall pipeline well because these models are trained only for traffic light detection.

Popular object detection techniques have also been used such as YOLO [15], which predicts bounding boxes within a relatively small amount of time by leveraging only a single layer, and SSD [16], which performs bounding box prediction from multiple layers of the network. MultiBox [17] uses a general approach in predicting bounding boxes for multi-class tasks from a fully connected layer. However, these models have lesser accuracy compared to Mask R-CNN [18], and this work aims to also classify traffic lights based on their color (red, yellow, green) along with detecting the entire traffic signal.

Vitas *et al.* [19] have proposed a two-step deep learning based approach for traffic light detection. They use adaptive-thresholding and blob detection for region-of-interest selection, and a VGG-16 based CNN model for traffic signal localization. Yoneda *et al.* [20] have used a digital map to extract the region-of-interest and a machine learning detector (OpenCV cascade detector trained on AdaBoost) for detecting traffic lights, which sacrifices accuracy for faster computation. Additionally, they detect direction lights using prior spatial information.

The primary goal of the solutions given above is to provide only traffic light detection. Moreover, to the best of the knowledge of the authors, there has been only one other research conducted on detecting the auxiliary countdown timer and its countdown value along with the traffic light. Sathiya *et al.* [21] use color segmentation-based techniques to find traffic light and the auxiliary timer, however, they use images containing only a single traffic light and its auxiliary timer. Additionally, their approach has a high processing time. The approach mentioned in this article uses cascaded multi-task models that can be used to give the desired results. There have been many works published where cascaded networks were successfully used. For instance, the authors of TTNet [22] use a cascaded model for detecting and tracking a table tennis ball on a table. A rough detection is estimated in earlier stages of the network which is then refined by a cascaded network. DeepPose [23] uses a cascaded regressor for refining the joints of human parts that were detected in the preceding networks of its framework. In this work, a new cascaded model for the detection of traffic signals and countdown timers has been devised.

III. Proposed Approach

This section describes the proposed framework for Traffic Signal and Countdown Timer Detection Network, TSCTNet, whose end-to-end architecture is shown in Figure 1. The goal of the network is to process video frames and swiftly detect traffic lights along with their auxiliary countdown timers. The knowledge gathered from the network can be fed to an autonomous vehicle system so that it may process the information and take necessary actions. The proposed framework performs the major tasks in the following order of stages:

S1: Traffic Signal Detection

S2: Traffic Signal Color Detection and Feature Extraction

S3: Traffic Signal Countdown Timer Detection

S4: Combining all the Stages' results

The following sections explain the different stages involved in performing the aforementioned tasks.

A. Traffic Signal Detection

In this framework, Mask R-CNN [18] is used as the object detection architecture for traffic signal detection, which is shown in Figure 2. Mask R-CNN is used as it improves Faster R-CNN [24] by using Region of Interest (RoI) Align instead of RoI Pooling. The key feature of RoI Align is that it rectifies the location misalignment issue present in RoI Pooling when it takes the input proposals from the Region Proposal Network (RPN) by dividing them into 'bins' using bilinear interpolation. Hence, Mask R-CNN improves upon the core accuracy of Faster R-CNN and is therefore used over it. The mask branch of Mask R-CNN network is removed as it is not applicable for the proposed system and leads to an improvement in the processing performance.

In the proposed network, the Mask R-CNN framework consumes raw video frames of resolution 1920×1080 px (downscaled to 1024×1024 px) sampled at 30 frames per second and outputs bounding boxes for traffic signals. The result of this stage is an output list with each element comprising of the top-left corner and bottom-right corner spatial coordinates of the bounding box, and its detection score.

B. Traffic Signal Color Detection and Feature Extraction

Once the object detection phase is completed for a video frame, only those detected traffic signals are retained in the output list that have a detection score above a specified threshold (taken here to be the standard value of 0.5). Subsequently, images of all the traffic signals are cropped based on their respective coordinates and then processed sequentially. The next step is to identify the color of each traffic signal.

The input image is first converted from RGB (Red-Green-Blue) to HSV (Hue-Saturation-Value) format. This is done because RGB color space has three channels for color encoding while HSV has only the hue channel which makes it easier to segment based on color. After the color space conversion process, color thresholding is performed on the image as shown in Figure 3. Three color ranges are chosen: red, yellow, and green, which represent the three different colors of a traffic light. Color image thresholding involves replacing each pixel in an image with a black pixel ('0' bit) if the pixel HSV value is outside of a specific range, or a white pixel ('1' bit) if the pixel intensity is within the bounds of the constants. This process is performed three times by using the lower and upper HSV range for red, yellow, and green colors to isolate the presence of these colors in the image.

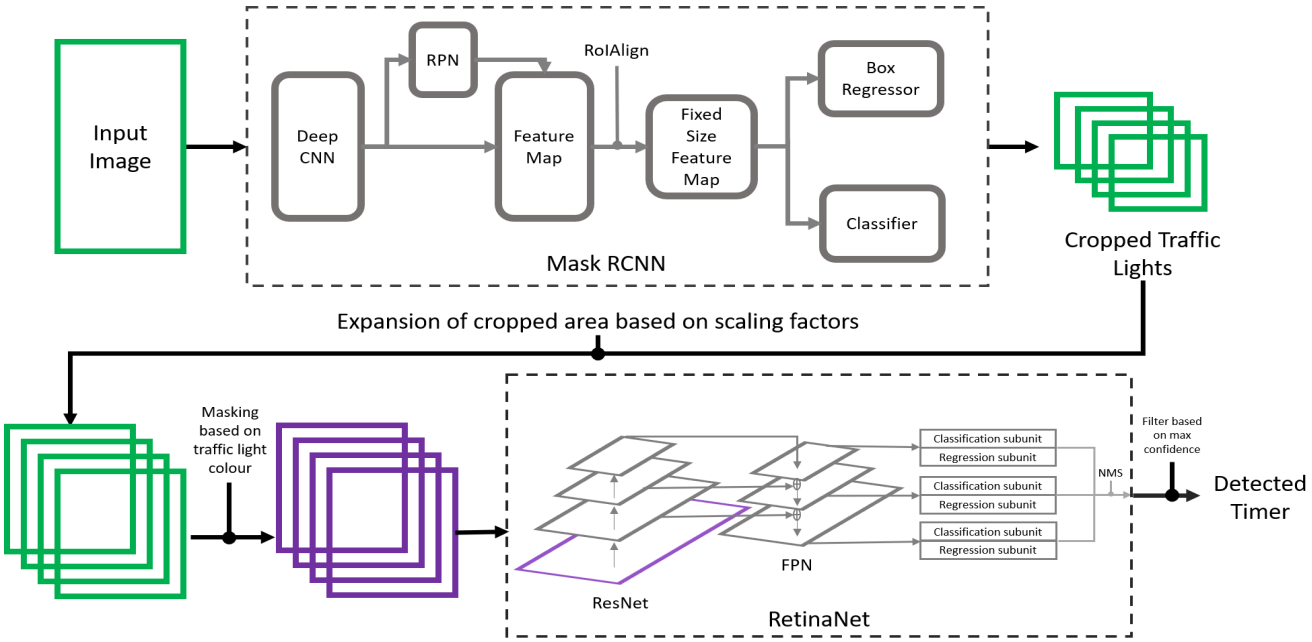

Figure. 1: TSCTNet architecture

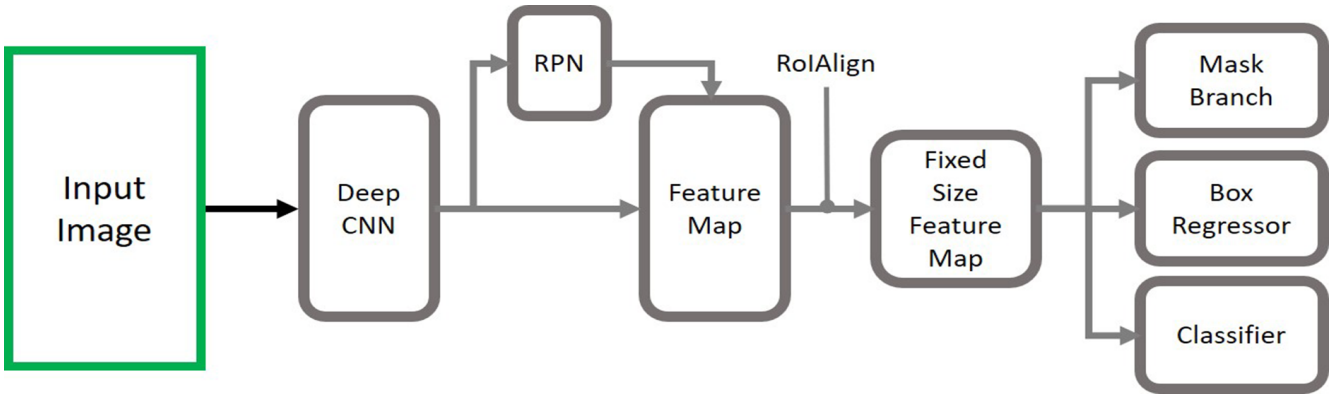

Figure. 2: Mask R-CNN architecture

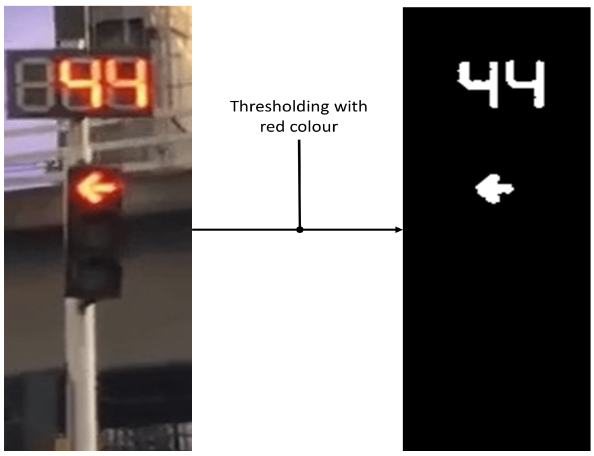

Figure. 3: Thresholding an image with red colour

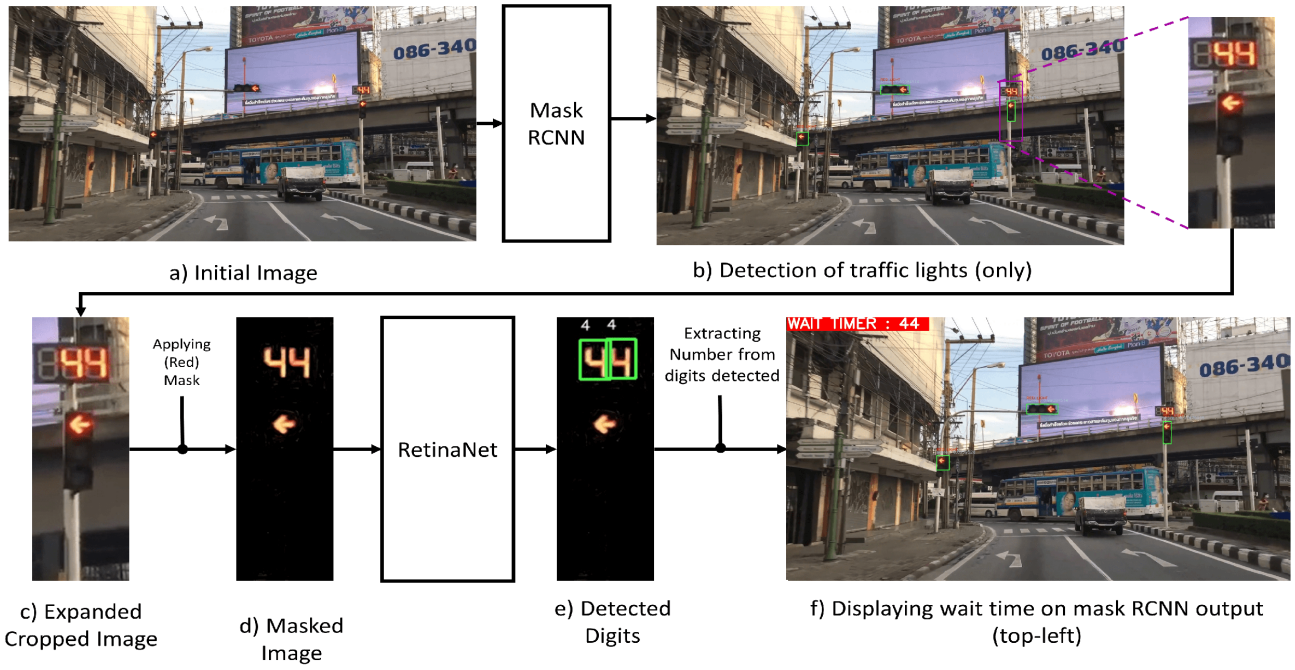

Figure. 4: A visual example depicting how countdown timer value is detected in a frame

The range of HSV values used by *OpenCV* (the library used for image processing) [25] for Hue, Saturation, and Value are 0-180, 0-255 and 0-255 respectively. The thresholding of the image for the three traffic signal colors is performed based on the condition shown in Eq. 1 for Hue, Saturation, and Value of the input image to obtain three thresholded images: I_R, I_Y, and I_G respectively. It should be noted that H denotes the Hue value at the corresponding pixel (x, y), and similarly S denotes Saturation and V denotes Value.

$$I_R(x,y): \begin{cases} 1, & \text{if } (0 <= H <= 10 \; and \\ & \quad 70 <= S <= 255 \; and \\ & \quad 50 <= V <= 255) \; or \\ & \quad (170 <= H <= 180 \; and \\ & \quad 70 <= S <= 255 \; and \\ & \quad 50 <= V <= 255) \\ 0, & otherwise \end{cases}$$

$$I_Y(x,y): \begin{cases} 1, & \text{if } 20 <= H <= 30 \; and \\ & \quad 100 <= S <= 255 \; and \\ & \quad 100 <= V <= 255 \\ 0, & otherwise \end{cases} \quad (1)$$

$$I_G(x,y): \begin{cases} 1, & \text{if } 40 <= H <= 100 \; and \\ & \quad 50 <= S <= 255 \; and \\ & \quad 50 <= V <= 255 \\ 0, & otherwise \end{cases}$$

Next the percentage coverage (C) for each of the three thresholded images is calculated. The percentage coverage is defined to be the percentage of white pixels in a thresholded image. Let n_w be the count of white pixels in a binary image. For an image of width w and height h, percentage coverage is defined as shown in Eq. 2.

$$C = \frac{n_w}{w \times h} \times 100 \quad (2)$$

Coverage for I_G is C_G, I_Y is C_Y, and I_R is C_R respectively. The maximum coverage among the three is found out using Eq. 3.

$$C_{max} = max(C_G, C_Y, C_R) \quad (3)$$

If C_{max} is above a certain constant, then the corresponding color is assigned as the color of the traffic signal. Else, the traffic light is assumed to be turned off.

C. Traffic Signal Countdown Timer Detection

The penultimate stage deals with the detection of the countdown timer value. The output list from Stage III-A is used to obtain the approximate presence of the timer. The traffic signal countdown timer is an auxiliary device that is attached to the same structure as the traffic light. This information is used to an advantage by detecting the timer, not from the entire image but from an extended cropped image (ECI).

An extended area around the traffic signal is cropped, thresholded, masked, and fed forward to a cascaded digit detection network with RetinaNet architecture (which comprises a ResNet-50 backbone). Let the center of the cropped image to be the pair (x_0, y_0), and height and width to be h and w respectively. The top-left (x_1, y_1) and bottom-right (x_2, y_2) coordinates can be represented as given in Eq. 4.

$$
\begin{aligned}
x_1 &= x_0 - 0.5(w) \\
y_1 &= y_0 - 0.5(h) \\
\\
x_2 &= x_0 + 0.5(w) \\
y_2 &= y_0 + 0.5(h)
\end{aligned}
\tag{4}
$$

To obtain this extended cropped image, two scaling factors for the original image, σ_h and σ_w, are chosen based on heuristics. Next, an extended image is cropped after introducing the scaling factor by updating the top-left ($x_1{}'$, $y_1{}'$)and bottom-right ($x_2{}'$, $y_2{}'$) coordinates as shown in Eq. 5.

$$
\begin{aligned}
x_1{}' &= x_1 - \sigma_w(w) \\
y_1{}' &= y_1 - \sigma_h(h) \\
\\
x_2{}' &= x_2 + \sigma_w(w) \\
y_2{}' &= y_2 + \sigma_h(h)
\end{aligned}
\tag{5}
$$

A new image (ECI, Figure 4(c)) cropped from these updated coordinates has a high chance of capturing the auxiliary timer device. In the event that a timer isn't detected in the first attempt, the scaling factor is increased, and the process is repeated once more. If the model fails to detect a timer again, it is concluded that the traffic light pole does not have an attached auxiliary timer device.

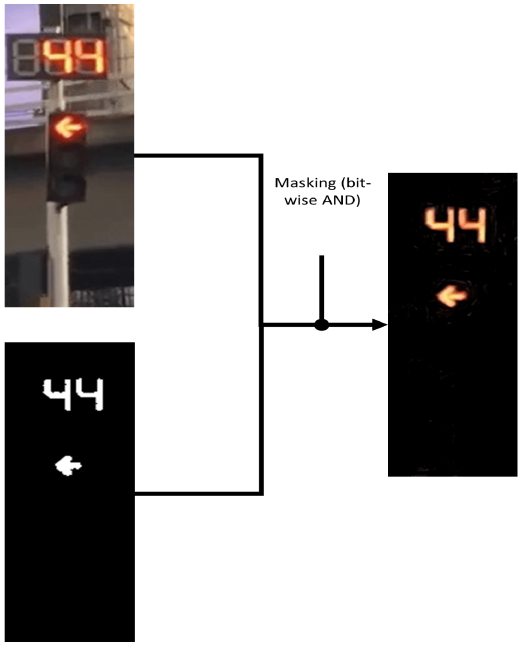

Figure. 5: MECI: Output after thresholding and masking the ECI

Once ECIs are obtained, an important intermediate step is performed to increase the robustness of the architecture. The prior knowledge of the colors of the traffic lights from the previous stage coupled with the fact that the color of the timer digits is the same as the traffic light enables the framework to process the ECI by converting it to a Masked ECI (MECI, Figure 4 (d)). This is done by thresholding the ECI in the range of the color of the contained traffic light and masking (performing bitwise AND operation) the resultant with the original ECI, thus generating MECI as shown in Figure 5. This process results in a more suitable image to work with as it facilitates the job of the neural network by masking the noise. Additionally, this process removes any unwanted digits appearing in the scene.

Layer Type	Number of iterations	Kernel Size (h × w × d)	Number of filters	Stride
Input Layer				
Conv1 (C1)	1	7×7×3	64	2
Max pool	1	3×3	1	2
Conv2 (C2)	3	1×1×64 3×3×64 1×1×64	64 64 256	1 1 1
Conv3 (C3)	4	1×1×256 3×3×128 1×1×128	128 128 512	1 1 1
Conv4 (C4)	6	1×1×512 3×3×256 1×1×256	256 256 1024	1 1 1
Conv5 (C5)	3	1×1×1024 3×3×512 1×1×512	512 512 2048	1 1 1

Table 1: Layers involved in modified ResNet-50 architecture

Multiple MECIs are generated and fed into a RetinaNet architecture to detect numerical digits in the frame. The following steps indicate how an image is passed through RetinaNet:

- **Preprocessing**: The input image is preprocessed by normalizing and centering. Normalizing is done by scaling the input image such that its pixel values are between 0 and 1. Thereby, the pixels are channel-wise centered by taking the mean of pixels across all training examples for the 3 color channels and subtracting the mean from the pixel value of the input image. This is done to center the values around 0. Images are then scaled between a minimum side length of 224 and a maximum side length of 256.

- **ResNet-50 backbone (Bottom-Up-Traversal)**: The ResNet-50 architecture is modified by removing the last 3 layers (average pooling, FCN, and Softmax), and adding a Feature Pyramid Network (FPN). The architecture is shown in Table 1. Each convolutional layer is followed by a batch normalization layer and a ReLU activation unit.

- **FPN (Top-Down-Traversal)**: Similar to the working of Mask R-CNN [18], the Conv5 (Convolution stage 5) layer output is directly used as a feature map M5. Further feature maps are generated by downsampling the preceding feature maps from top-down layers by a factor of 2 and combining them with corresponding bottom-up convolution stage output via a lateral connection. The layers in the bottom-up pathway are passed through a 1×1 convolutional layer so that the depths

can be downsampled to their corresponding depths of the top-down layer for in-place addition to take place. This is the process for generating feature maps M4 and M3.

- **Feature maps (M3-M5)**: Each of these feature maps is passed through a 3×3 convolutional layer to generate pyramid feature maps (P5-P3). P5 is passed through a max-pooling layer to generate feature pyramid P6.
- **Classification subnet**: Pyramid feature maps are fed to this subnet, which consists of four 3×3 convolutional layers with a depth of 256. This is followed by ReLU activation, 3×3 convolutional layer having 50 filters, and sigmoid activation.
- **Regression subnet**: This subnet performs regression on the anchor boxes, which also consists of four 3×3 convolutional layers similar to the classification subnet. However, the following 3×3 convolutional layer has only 20 filters, which is then followed by sigmoid activation.

An important point to note is that the images fed into the network need not be of a fixed size as the network does not have any fully connected layers. Instead, 1×1 convolutional layers are used to perform the part of fully connected layers. This helps in feeding variable-sized cropped traffic light images to the network.

The detected digits are combined to form a number, with the rightmost digit taken as the one's position. Subsequent digits from right to left are taken with the increasing power of ten's positions. If multiple MECIs successfully detect numbers, the one which detects the number with the highest confidence is taken and the rest are removed.

D. Combining all the Stages' results

The overall workflow of the framework is displayed in Figure 4. The input image is passed through the Mask R-CNN network to detect traffic signals which are in turn pre-processed and fed into the RetinaNet architecture to obtain the countdown timer value.

Now, instead of detecting the countdown time every frame, the value is stored and is reduced by one every second. However, some precautions need to be taken with this approach. Each second, a fixed number of frames are passed. When the traffic signal, along with its countdown timer, is detected for the first time (with a good confidence score), it is highly likely that the particular frame was not the first frame for that countdown timer value. On storing this value and starting an internal countdown every second (thirty frames), there may be discrepancies with the actual countdown value of the timer. Therefore, the following modified approach is taken to ensure that the internal countdown clock is synchronous to the actual countdown time:

S1: The first time a countdown timer's value is detected, it is not stored but the network continues detecting the countdown value every frame until the next lower countdown value is detected.

S2: Once the next countdown value is obtained, it is guaranteed that the frame was the first frame for that particular value, therefore the new countdown value is stored internally, and the framework stops detecting countdown values for every frame.

S3: Finally, to ensure robustness in synchronization of the internal countdown clock and the actual countdown value, every two seconds (sixty frames), the countdown value is detected again and calibrated with the current internally stored value. In case of discrepancies, the newer value replaces the older one. This works because as the automobile gets nearer to the traffic signal, the image of the traffic signal increases in size and becomes clearer, thus enabling the neural network to get better detection results.

Figure 4(d) displays the detected countdown value as the 'wait time' for the autonomous vehicle on the top-left corner of the image. 'Wait' timer (Red light) is counted down until zero while the 'Go' timer (Green light) is reset once the vehicle crosses the traffic signal i.e., the traffic light is not in the frame.

IV. Experimental Evaluation

A. Operating Environment

The authors have utilized Intel® Xeon® CPU @ 2.30GHz with NVIDIA Tesla K80 GPU, 12 RAM (accessible on the Google Colab platform) to perform the training and testing for the models involved. The program modules were written in $Python - 3.6$ and made use of $Tensorflow - 1.14.0$ and $Keras - 2.2.4$ libraries. Video and image processing was done using $OpenCV 4.1.2$ library.

B. Datasets Used

The following datasets have been used for training and validating the deep neural networks in the TSCTNet framework:

- **MS COCO:** The Microsoft Common Objects in Context (MS COCO) dataset [26] is a well-known dataset that has annotations for instance segmentation and bounding boxes which are used to evaluate how well object detection and image segmentation models perform. It contains 91 common objects as classes. Out of the total classes, 82 have 5,000 annotations or more. There are overall 328,000 images with more than 2,500,000 annotated objects. The dataset has been utilized to perform training on the Mask R-CNN [18] framework on traffic signals object class.
- **SVHN dataset:** The Street View House Number (SVHN) dataset is an annotated image dataset [27] consisting of real-world colored house-number images with annotated bounding boxes for each digit. It contains ten classes, one for each numerical digit. A subset of 50,000 images (40,000 for training subset; 10,000 for validation subset) has been taken for training and testing the RetinaNet model.
- **Custom Fixed Traffic Signal (CFTS) dataset:** The performance measure for traffic signal detection using Mask R-CNN along with the overall framework has

been evaluated on dash-cam footage of vehicles crossing pre-timed traffic lights, compiled from YouTube. The dash-cam videos have been taken at 30 frames per second and traffic signals and countdown timers have been annotated manually. The dataset consists of dash-cam video footage from various countries around the world, including India, Thailand, and Cambodia. A handful of dataset images are shown in Figure 6.

Figure. 6: Images taken from test videos of traffic light detection dataset

C. Results, Statistics, and Comparison with Existing Models

The performance of the two key stages of TSCTNet is evaluated based on two parameters: precision (Eq. 6), and recall (Eq. 7), which are calculated from the number of True Positives (TP), False Positives (FP), and False Negatives (FN). A curve is generated between precision and recall with detection values sorted based on their confidence. As there is a trade-off between precision and recall, and the precision-recall curves may intersect for various models, an additional criterion, average precision (AP), is used to obtain the accuracy of the model. Average precision is the precision value averaged across all unique recall levels. To reduce the impact of the noise in the curve, the precision is first interpolated (Eq. 8) at multiple recall levels before truly calculating average precision (Eq. 9).

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{7}$$

$$p_{interp}(r) = \max_{r' \geq r} p(r') \tag{8}$$

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1}) \tag{9}$$

Additionally, the traffic signal countdown timer is said to be present within the extended cropped image (ECI) if the boundaries of the ECI completely inscribe that of the countdown timer. This is done by comparing the ground truth countdown timer bounding box with the computed ECI bounding box (which has expansion factors σ_w and σ_h). The results are illustrated in Table 2.

Evaluation Parameter	Total Inscription
Success Rate (%)	95.65

Table 2: Evaluation result for the success rate of traffic signal countdown timer inscription

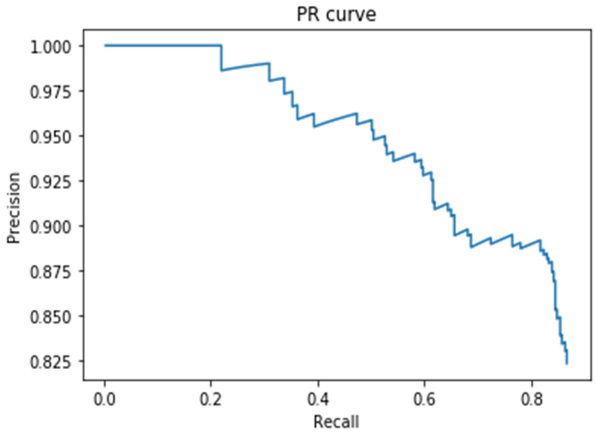

Figure. 7: Precision vs Recall curve plotted on the output of Mask R-CNN framework tested on the test set at IoU of 0.5

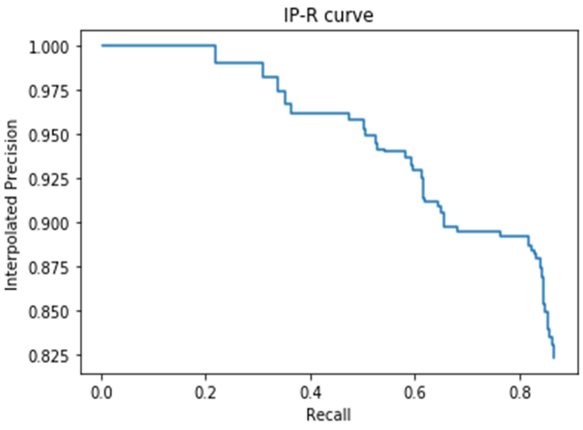

Figure. 8: Interpolated Precision vs Recall curve plotted after interpolating precision values from the curve in Figure 7

The first stage of the network (Mask R-CNN), which has been trained to detect traffic signals, is evaluated on the CFTS test set and the metrics are displayed in Table 3. To get True Positives (TP), False Positives (FP), and False Negatives (FP), Intersection over Union (IoU) is used as the comparison parameter. Since every part of the image where an object is not predicted is considered a negative, the measurement of True Negative is considered ineffectual. The threshold for the IoU is empirically chosen as 0.5, therefore if the

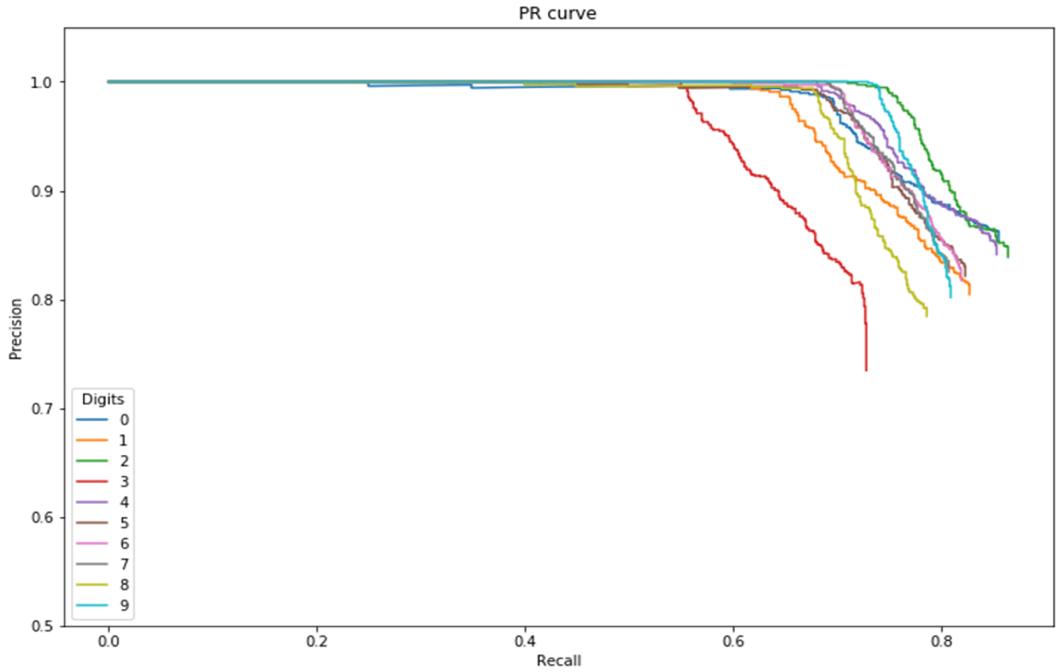

Figure. 9: Precision vs Recall curve plotted for RetinaNet model tested on SVHN test set at IoU of 0.5

IoU is greater than 0.5, it is considered as True Positive, else it is considered as False Positive. The Precision-Recall and the Interpolated Precision-Recall curves have been plotted as shown in Figure 7 and Figure 8 respectively. As the graphs suggest, the recall increases with each prediction, but the precision value may increase or decrease based on whether the result was True Positive (increases) or False Positive (decreases) and the general trend for precision is decreased in value. This result is expected because the values are plotted based on sorted confidence scores of predictions, where there is a high chance of a True Positive in case of high confidence, whereas the recall value is bound to increase as False Negative values can only decrease when a greater number of detections have been made.

Evaluation Parameter	IoU = 0.5
Precision	0.8235
Recall	0.8668
Average Precision (AP)	82.21%

Table 3: Evaluation result for Mask R-CNN tested on CFTS test dataset

The final stage of the network, RetinaNet (which comprises a ResNet-50 backbone), is trained, tested, and validated on a dataset made with the combination of Street View House Numbers (SVHN) dataset [27] and annotated countdown timer digits from the custom test dataset (CFTS) for digit recognition. The model is trained for 40 epochs with each epoch having 10,000 steps with a learning rate of 0.0001. Training batch size is 4, therefore one epoch goes through a random assortment of the 40,000 training images on which random transformations like cropping and translation were performed to increase the training set size where each step processes four images. The criterion for evaluating the network is the same as that of Mask R-CNN, i.e., precision, recall, and average precision. As there is more than one class in this network, the mean of average precision values across all 10 classes is evaluated using Eq. 10. Precision-Recall curves are illustrated in Figure 9. The precision values are interpolated to calculate the average precision. It can be interpreted from the curve that more complex shapes of digits '3' and '8' make them harder to detect while other digits have comparative results. Also, digit '0' has the highest precision and recall due to its easily distinguishable shape.

Class (Digit)	Precision (IoU = 0.5)	Recall (IoU = 0.5)	Average Precision (IoU = 0.5, %)
0	0.8562	0.8643	83.73
1	0.8104	0.8332	80.65
2	0.8422	0.8688	85.26
3	0.7833	0.7763	70.90
4	0.8411	0.8534	83.91
5	0.8295	0.8319	80.87
6	0.8273	0.8290	80.70
7	0.8321	0.8135	79.69
8	0.7938	0.7958	77.14
9	0.8042	0.8122	80.15

Table 4: Evaluation result for RetinaNet tested on the combination of SVHN and CFTS test set

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{10}$$

The mean average precision (mAP) from Table 4 results in 80.30%.

The results of this work have been compared with other studies as shown in Table 5. It is important to note that the

Approach	Precision (%)	Recall (%)	Elapsed Time (in seconds per frame)
Sathiya *et al.* [21]	98.00	88.00	3.06
TSCTNet	82.20	82.78	0.20

Table 5: Comparison of results to other frameworks for auxiliary timer countdown value recognition

results of the approach of Sathiya *et al.* [21] (the only other suitable study found in the given domain of traffic signal auxiliary timer countdown value detection) can be misleading as there are certain notable issues with it. Firstly, their approach only works for single digits between 0 to 9 and therefore their approach cannot be utilized for countdown values containing two or more digits. Furthermore, they use cropped static images instead of videos, and their approach does not work in real-time due to the long processing times of their approach for each frame.

V. Conclusions

In this article, a new deep learning based network (TSCTNet) has been proposed for a fixed traffic signal and auxiliary countdown timer detection. The use of cascaded networks and the incorporation of lightweight computer vision and color space manipulation techniques make the framework both efficient and accurate. Since the extended cropped image (ECI) is a feature-rich image with less noise, the framework localizes the auxiliary timer value and detects the countdown value with high accuracy.

Additionally, the framework can be pragmatically utilized in real-world applications as its methodology has also been explained using real-world test scenarios. It is to be noted that the proposed framework can detect traffic lights with an average precision of 82.21%, while the auxiliary countdown timer digits are detected with a mean average precision of 80.30%. The experiments were performed on dash-cam videos under various conditions such as daylight, nighttime, and rain. The proposed framework can be used to augment autonomous vehicle systems so that they may perform well in areas where fixed-timed traffic lights are in use. However, the framework has a limitation in that it performs well only on high-definition videos, and the presence of multiple auxiliary timers (in very rare instances) in a frame can sometimes lead to incorrect results. Lastly, it is noteworthy that although the traffic signal countdown timer detection method is invariant to its relative position to the traffic signal, its efficacy does deteriorate as the distance between the timer and the traffic signal increases. Future works can be conducted to address these issues and improve real-time performance.

Acknowledgments

This research is supported by the National Institute of Technology Warangal in terms of computer lab facilities, guidance from institute faculty, and the motivation to undertake this work. However, it is to be noted that this research received no specific monetary grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*, 2020.

[2] Syed Shah Sultan Mohiuddin Qadri, Mahmut Ali Gökçe, and Erdinç Öner. State-of-art review of traffic signal control methods: challenges and opportunities. *European Transport Research Review*, 12(1):55, Oct 2020.

[3] Anuj Sharma, Lelitha Vanajakshi, V. Girish, and M. S. Harshitha. Impact of signal timing information on safety and efficiency of signalized intersections. *Journal of Transportation Engineering*, 138(4):467–478, 2012.

[4] Kit Meng Lum and Harun Halim. A before-and-after study on green signal countdown device installation. *Transportation Research Part F: Traffic Psychology and Behaviour*, 9(1):29–41, 2006.

[5] Fuquan Pan, Lixia Zhang, Changxi Ma, Haiyuan Li, Jinshun Yang, Tao Liu, Fengyuan Wang, and Shushan Chai. Impact of vehicular countdown signals on driving psychologies and behaviors: Taking china as an example. *Journal of Advanced Transportation*, 2017:5838520, Apr 2017.

[6] Mohammadreza Javanmardi, Ziqi Song, and Xiaojun Qi. Automated traffic sign and light pole detection in mobile lidar scanning data. *IET Intelligent Transport Systems*, 13(5):803–815, 2019.

[7] Felix Lindner, Ulrich Kressel, and Stephan Kälberer. Robust recognition of traffic signals. *IEEE Intelligent Vehicles Symposium, 2004*, pages 49–53, 2004.

[8] Hyun-Koo Kim, Y Shin, Sa gong Kuk, Ju H. Park, and Ho-Youl Jung. Night-time traffic light detection based on SVM with geometric moment features. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 7:472–475, 2013.

[9] Masako Omachi and Shinichiro Omachi. Traffic light detection with color and edge information. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 284–287, 2009.

[10] Masako Omachi and Shinichiro Omachi. Detection of traffic light using structural information. In *IEEE 10th International Conference on Signal Processing Proceedings*, pages 809–812. IEEE, 2010.

[11] Martin Bach, Stephan Reuter, and Klaus Dietmayer. Multi-camera traffic light recognition using a classifying labeled multi-bernoulli filter. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1045–1051. IEEE, 2017.

[12] Michael Weber, Peter Wolf, and J Marius Zöllner. DeepTLR: A single deep convolutional network for detection and classification of traffic lights. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 342–348. IEEE, 2016.

[13] Feng Gao and Caimei Wang. Hybrid strategy for traffic light detection by combining classical and self-learning detectors. *IET Intelligent Transport Systems*, 14(7):735–741, 2020.

[14] Julian Müller and Klaus C. J. Dietmayer. Detecting traffic lights by single shot detection. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 266–273, 2018.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788. IEEE, 2016.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[17] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2155–2162, 2014.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[19] Dijana Vitas, Martina Tomic, and Matko Burul. Traffic light detection in autonomous driving systems. *IEEE Consumer Electronics Magazine*, 9(4):90–96, 2020.

[20] Keisuke Yoneda, Akisuke Kuramoto, Naoki Suganuma, Toru Asaka, Mohammad Aldibaja, and Ryo Yanase. Robust traffic light and arrow detection using digital map with spatial prior information for automated driving. *Multidisciplinary Digital Publishing Institute*, 20(4):1181, 2020.

[21] S. Sathiya, M. Balasubramanian, and D. V. Priya. Real time recognition of traffic light and their signal countdown timings. In *International Conference on Information Communication and Embedded Systems (ICICES2014)*, pages 1–6, 2014.

[22] Roman Voeikov, Nikolay Falaleev, and Ruslan Baikulov. Ttnet: Real-time temporal and spatial video analysis of table tennis. *arXiv preprint arXiv:2004.09927*, 2020.

[23] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[25] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[27] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Author Biographies

Dhananjai Chand currently works as a Software Engineer with Salesforce.com, Inc. in Hyderabad, India. He received the B.Tech. degree in computer science from the Department of Computer Science and Engineering, National Institute of Technology Warangal, Telangana, India in 2020. His current research interests include deep learning, computer vision, and autonomous systems. Contact him at *dcdevdhan@gmail.com*.

Savyasachi Gupta currently works as a Software Engineer with Visa, Inc. in Bangalore, India. He received the B.Tech. degree in computer science from the Department of Computer Science and Engineering, National Institute of Technology Warangal, Telangana, India in 2020. His current research interests include deep learning, computer vision, and developing solutions for resolving COVID-19. Contact him at *gsavya10@gmail.com*.

Ilaiah Kavati received his Ph.D. from University of Hyderabad, India. He received his B.Tech. and M.Tech. from Jawaharlal Nehru Technological University, Hyderabad. He is currently working as an Assistant Professor in National Institute of Technology, Hyderabad. His research interests include biometrics, pattern recognition, image processing, internet of things and security. He published more than 30 publications in various international journals and conferences of repute.