Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 28, S. 1469-1475, Kasım 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 28, pp. 1469-1475, November 2021

Research Article

Turkish Traffic Sign Recognition: Comparison of Training Step Numbers and Lighting Conditions

Kaan Kocakanat[1*], Tacha Serif[1]

[1*] Yeditepe University, Faculty of Engineering, Department of Computer Engineering, Istanbul, Turkey, (ORCID: 0000-0002-5906-7969, 0000-0003-1819-4926), kaan.kocakanat@std.yeditepe.edu.tr, tserif@cse.yeditepe.edu.tr

(1st International Conference on Applied Engineering and Natural Sciences ICAENS 2021, November 1-3, 2021)

(**DOI:** 10.31590/ejosat.1015972)

ATIF/REFERENCE: Kocakanat, K. & Serif, T. (2021). Turkish Traffic Sign Recognition: Comparison of Training Step Numbers and Lighting Conditions. *European Journal of Science and Technology*, (28), 1469-1475.

Abstract

With the ever increasing number of vehicles on the roads, traffic signs are becoming more and more important every passing day. Despite the fact that traffic signs are simple and easy to understand, in congested traffic drivers may miss them. Considering that even milliseconds can make a huge difference in preventing accidents, it would make a big help if a system could assist the driver with traffic signs. In order to achieve this, a traffic sign recognition system needs to be implemented. Accordingly, this study aims to develop a Turkish traffic sign detection and recognition system using the Faster R-CNN algorithm. The proposed solution utilizes TensorFlow framework and specifically makes use of the Faster R-CNN Inception-v2-COCO to train the object detection model. For training purposes, indigenous dataset is created containing 54 classes and 10842 Turkish traffic sign images. The training process of the model is carried out twice with step numbers 51,217 and 200,000, respectively. Then, these two models are used to detect 10 Turkish traffic sign images taken both daytime and nighttime. The results indicate that the proposed system's average precision is 67.2% and average recall is 78.3% when trained with 51,217 steps; on the other hand, the average precision increases to 76% and average recall to 82.8% when trained with 200,000 steps.

Keywords: Turkish Traffic Signs, Traffic Sign Recognition, Faster R-CNN, Object Detection, TensorFlow.

Türk Trafik İşareti Tanıma: Eğitim Adım Sayıları ve Aydınlatma Koşullarının Karşılaştırılması

Öz

Yollardaki araç sayısının her geçen gün artmasıyla birlikte trafik işaretleri her geçen gün daha da önem kazanmaktadır. Trafik işaretleri basit ve anlaşılması kolay olmasına rağmen, sıkışık trafikte sürücüler bunları gözden kaçırabilir. Milisaniyelerin bile kazaları önlemede büyük fark yarattığını göz önünde bulundurarak, sürücüye trafik işaretleri konusunda yardımcı olacak bir sistemin olmasının büyük bir fayda sağlayacağı oldukça açıktır. Bunun için bir trafik işareti tanıma sisteminin geliştirilmesi gerekmektedir. Bu makalede, Daha Hızlı R-CNN algoritması kullanılarak bir Türk trafik işareti tespit ve tanıma sisteminin geliştirilmesi amaçlanmaktadır. Önerilen çözüm, TensorFlow çerçevesi ile nesne algılama modelini eğitmek için Daha Hızlı R-CNN Inception-v2-COCO'yu kullanır. Modelin eğitilmesi için 54 sınıf ve 10842 adet Türk trafik işareti görüntüsünü içeren yeni bir veri seti oluşturulmuştur. Modelin eğitimi sırasıyla 51.217 ve 200.000 eğitim adım numaraları ile iki kez gerçekleştirilir. Daha sonra bu iki model kullanılarak gündüz ve gece çekilen 10 adet Türk trafik işareti görüntüsü tespit edilmeye çalışılmıştır. Sonuçlar, önerilen modellerin 51.217 eğitim adımıyla eğitildiğinde ortalama hassasiyetin %67,2 ve ortalama hatırlamanın %78,3 olduğunu göstermektedir; Öte yandan, model 200.000 eğitim adımıyla eğitildiğinde ortalama hassasiyet %76'ya ve ortalama hatırlamanın da %82,8'e yükselir.

Anahtar Kelimeler: Türk Trafik İşaretleri, Trafik İşareti Tanıma, Daha Hızlı R-CNN, Nesne Algılama, TensorFlow.

[*] Corresponding Author: kaan.kocakanat@std.yeditepe.edu.tr

1. Introduction

Undeniably, there is a rapid growth in the number of vehicles globally (Davis & Boundy, 2020). As the number of vehicles in traffic increases, so does the level of uncertainty in the traffic environment. To eliminate this uncertainty and manage traffic flow, traffic signs are placed on the roads. They are designed to be easy to understand to inform drivers about the current situation and other important information on the road. However, accidents can still occur when drivers do not pay attention to the traffic signs (Bucsuházy et al., 2020). For this very reason, a traffic sign recognition (TSR) system needs to be implemented in order to prevent accidents. Even though there are many studies on TSR systems, very few of these studies do not evaluate real-life driving conditions, such as at night or adverse weather conditions. Furthermore, designing a TSR system for a specific country is challenging because of a lack of public datasets - i.e even though multiple TSR studies were undertaken for various countries in the literature, most of them have based their work on the German traffic sign datasets (Stallkamp et al., 2011), (Houben et al., 2013). In addition, while traffic signs in most countries may seem to be similar, there are actually major differences between country traffic signs even within the same economical regions - e.g. Europe, United Kingdom, Americas, Asia and Australia. Hence, this study aims to utilize TensorFlow to implement a Turkish TSR system and evaluate it both at daytime and nighttime conditions. Therefore, as part of this work a Turkish Traffic Sign Dataset containing 54 different traffic signs is created and the Faster R-CNN Inception v2 COCO model is trained within the TensorFlow framework.

Accordingly, this paper is structured as follows; section II describes the previous work undertaken in TSR, section III highlights the methods used for TSR are reviewed and elaborated upon, section IV analyzes the requirements for the proposed system and explains in detail the tools selected for the proposed solution. Section V describes the development of the prototype and the creation of the Turkish Traffic Sign Dataset for the proposed system, as well as the essential steps taken for training the model. As a follow up, section VI highlights the evaluation methods used and elaborates on their outcomes and findings. Last but not least, Section VII draws the conclusion and discusses possible future venues for improvement.

2. Background

Traffic signal detection and recognition is a popular topic in computer vision with a wide range of applications. Since it is challenging to develop a system approach capable of identifying various types of traffic signs, this topic has been the focus of several research.

Yaliç and Can's TSR prototype is one of the earliest examples of Turkish TSR prototypes in the literature. (Yaliç and Can, 2011). In this work, the authors develop a system for the automatic recognition of 52 signs in Turkish roads. Their proposed system first detects probable sign regions in previously recorded videos of roads from a moving car, and then matches each identified region to an existing sign in the database or classifies it as a non-sign region. The prototype system achieves this by utilizing the Scale-invariant feature transform technique for feature extraction, and applying the normalized correlation approach for some traffic signs with the same shape but different angles. To create an evaluation content, the authors have recorded 164 traffic signs from a moving car at the speeds ranging between 30 to 70 km/h. Then this content is fed to the proposed system and it was able to identify 154 of the traffic signs, which sets the success rate of the system to around 94%.

A novel method is proposed for detecting circular traffic signals (Gündüz et al., 2013). This study is unique because it uses a combination of the two algorithms. The proposed system utilizes a newly developed circle detection algorithm for circular TSR alongside histograms of oriented gradients (HOG)-based features. Thus, by estimating the circular actual borders of the traffic signs, an unnecessary background image can be eliminated. During the evaluation stage, two model classifiers, Support Vector Machine (SVM) and Random Forest (RF), are used on 13,541 images containing 13 classes of German Traffic Sign Recognition Benchmark (GTSRB) to evaluate the performance of the proposed method. When only HOG-based features are used, the highest correct classification rate is 96.46%. On the other hand, when the proposed algorithm is combined with the HOG-based features, this rate increases to 97.17%. The most successful results obtained using the SVM classifier.

Another study, which utilizes feature extraction and recognition of traffic sign images, is carried out by Cinar et al. (Cinar et al., 2020). In this study, convolutional neural networks (CNN) are used to extract the image features and then the obtained image features are classified using the RF method. As part of this study, a dataset created by authors containing 1500 images of 14 Turkish traffic signs to extract 1000 features. The evaluation of the system resulted with a 93.7% success of the proposed classifier.

Irfan and Galip used the TensorFlow framework to detect and recognize traffic signs in Turkey (Kilic & Aydin, 2020). To achieve this, the authors created a Turkish Traffic Sign Dataset with 1250 pictures under diverse traffic and weather conditions, containing 41 distinct traffic signs. The dataset is trained on the Faster R-CNN Inception V2 COCO model using the TensorFlow framework and the model is trained with 200,000 steps. In the evaluation of the proposed system, 313 images were used and the system detected 447 traffic signs. It has been noted that when the threshold value is greater than 0.50, the model detects 423 signals with a 94.63% accuracy.

Çetinkaya and Acarman conducted a study on the detection of traffic signs using a novel image pre-processing technique to enhance the performance of a traffic sign detector (Çetinkaya & Acarman, 2021). In this study, the Faster R-CNN Inception Resnet V2 COCO model is used and the model is trained on German Traffic Sign Detection Benchmark dataset (GTSDB). The proposed image pre-processing method is based on filtering unimportant regions of images and emphasizing the more useful ones, reducing noise and allowing more effective detector training. Accordingly, the model is trained with 250,000 steps and training is repeated twice in the context of main experiments; first using original images and second time using pre-processed images. Precision, recall, and F-measure values for pre-processed images were 94.53 %, 91.91 %, and 93.2 %, respectively, when the threshold value was set to 0.7. As a result, the suggested image preprocessing approach improved the object detector's performance.

3. Methodology

Many different techniques and algorithms are used for TSR systems in the literature. In particular, techniques based on color and shape features have become popular, and these are referred to as traditional methods. Examples of some traditional methods are

the HOG feature descriptor, color segmentation and the Hough transform (De La Escalera et al., 2004), (Dalal & Triggs, 2005), (Garcia-Garrido et al., 2006), (Maldonado-Bascón et al., 2007). Many detection methods have been developed with traditional methods in the past decades.

In recent years, one of the most commonly used techniques is deep learning-based methods such as CNN. With the introduction of AlexNet (Krizhevsky et al., 2012) in 2012, CNN models and deep learning became popular. This CNN architecture won the ImageNet LargeScale Visual Recognition Challenge (ILSVRC) and performed substantially better when compared to the traditional approaches. Hence, CNN has been widely used for TSR problems (Girshick et al., 2014), (Girshick, 2015), (Ren et al., 2015), (Cinar et al., 2020), (Kilic & Aydin, 2020), (Çetinkaya & Acarman, 2021).

3.1. Convolutional Neural Networks (CNN)

CNN is the one of the most popular deep learning algorithms, which achieved very significant results in some complex visual tasks. CNN consists of multiple layers placed one after the other for feature extraction of curves and edges and builds them up to more abstract concepts (Figure 1). The continuation of these layers is a trainable classifier.

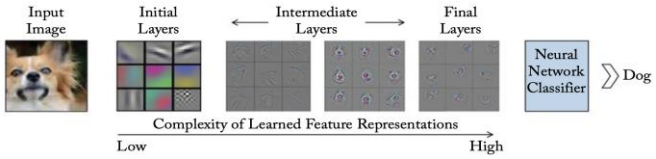

Figure 1. Different representations of the object on different layers in the network (Khan et al., 2018).

In CNN, after the input data is received, the training process is carried out layer-by-layer. Finally, it gives a final output to compare with the correct result. As a result of the comparison, an error occurs equal to the difference between the produced result and the desired result. By updating the weights at every iteration, the error is reduced.

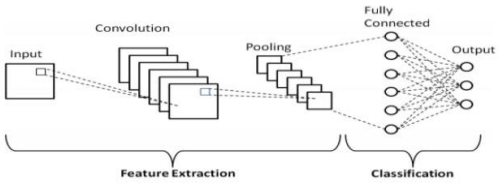

Figure 2. A basic diagram of a CNN architecture (Phung & Rhee, 2018).

As seen in Figure 2, a typical CNN is constructed by input layer, output layer and the repetition convolutional layers, pooling layers and fully connected layers. Each layer has its own function. The input layer is the first layer of the whole CNN. In this layer, data is given raw to the network. The input layer in CNN should contain image data, and the image data is usually represented by a matrix that holds the pixel values of the image. A convolutional layer is the fundamental component of a CNN. In the convolutional layer, the input to the convolution layer is convolved with what is called a kernel, a convolution matrix, or a filter to generate an output feature map. Convolution process is based on the process of sliding a filter over the input image (Figure 3).

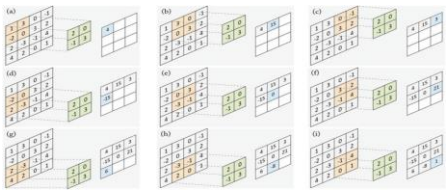

Figure 3. A convolution process with a 2x2 filter applied to a 4x4 input image (Khan et al., 2018).

Pooling layer is another layer that is often added between successive convolutional layers in the CNN. A pooling layer operates on blocks of the input feature map and combines the feature activations. The task of this layer is to reduce the spatial size of the data representation, the parameters within the network, and the number of computations. Thus, it also controls overfitting.

3.2. Faster R-CNN

Region-based convolutional neural networks (R-CNN) are introduced by Girshick et al., and it is a foundation method for all modern object detection networks (Girshick et al., 2014). This study is unique since it is the first to combine region proposals with CNN. In this proposed method, the system receives an input image and extracts around 2000 region proposals from the bottom-up with Selective Search (Uijlings et al., 2013).

After R-CNN, Girshick proposed a new approach for object detection called Fast R-CNN to solve the difficulties of R-CNN (Girshick, 2015). The approach is similar to the R-CNN algorithm but Fast R-CNN uses a main CNN with multiple convolutional layers to detect features in the picture before proposing regions. In this model, the input image is given to CNN to generate a convolutional feature map. Thus, there is no need to use a separate CNN for each region proposal. In R-CNN and Fast R-CNN models, the performance of the network is slow as selective search is used to find region proposals. In order to solve this problem, Ren et al. abandoned the selective search method completely and developed the Region Proposal Network (RPN) algorithm (Ren et al., 2015). When RPN and the Fast R-CNN object detector are combined, a new model called Faster R-CNN is developed. Using a RPN algorithm in Faster R-CNN instead of selective search cuts down on the number of proposed regions while also ensuring accurate object detection. RPN is a deep convolutional neural network that is used to generate proposal regions.

The Faster R-CNN can be evaluated in two stages (Figure 4). In the first stage, RPN instructs the second stage of Faster R-CNN where to search for the object. RPN takes images of any size as input and produces a set of rectangular objects which are proposed as the locations of the object based on the object score. It makes these proposals by sliding a small spatial window over the feature map that is generated by the convolutional layer.

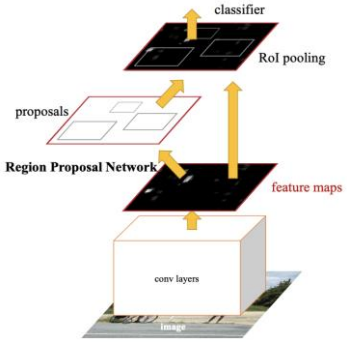

Figure 4. The architecture of Faster R-CNN model (Ren et al., 2015)

The Fast R-CNN object detector is the second stage of the Faster R-CNN. It takes the feature maps that the initial CNN generated and performs RoI pooling on them. With the introduction of the RPN, Faster R-CNN eliminates the region proposal limitations inherited by Fast R-CNN. Thus, Faster R-CNN is better than Fast R-CNN in terms of speed.

3.3. Evaluation Methodology

Since it is widely used to benchmark the performance of object detection, the COCO metrics are utilized as evaluation criteria for the detection accuracy in this study. As shown in Fig. 5, Average Precision (AR) and Average (Recall) are averaged over multiple Intersections over Unity (IoU) values. In COCO evaluation metric, the IoU threshold ranges from 0.50 to 0.95 with a step size of 0.05 represented as AP@[.5:.05:.95]. This is known as "mean average precision" (mAP) in the object detection field.

```
Average Precision (AP):
  AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
  AP^small            % AP for small objects: area < 32^2
  AP^medium           % AP for medium objects: 32^2 < area < 96^2
  AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
  AR^max=1            % AR given 1 detection per image
  AR^max=10           % AR given 10 detections per image
  AR^max=100          % AR given 100 detections per image
AR Across Scales:
  AR^small            % AR for small objects: area < 32^2
  AR^medium           % AR for medium objects: 32^2 < area < 96^2
  AR^large            % AR for large objects: area > 96^2
```

Figure 5. COCO evaluation metrics (COCO, 2015)

On the other hand, Figure 6 describes the precision, recall and UoI calculation formulas, where TP, FP, and FN indicate true positive, false positive, and false negative, respectively. Hence, it can be clearly seen that the higher the precision and recall, the better the accuracy. In this formula, the object is the area of the correct object and the detected box is the predicted candidate area. IoU is obtained by dividing the overlap area by the union area.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{IoU} = \frac{(Object \cap Detected\ box)}{(Object \cup Detected\ box)}$$

Figure 6. The Formula of Precision, Recall and IoU

4. Dataset Creation And Implementation

This section describes the steps taken to process the creation of the Turkish Traffic Sign Dataset and the necessary processes applied to train the model.

4.1. Turkish Traffic Sign Dataset

In order to create a successful model, a new indigenous dataset is created since previous studies mostly used the German dataset. This approach is taken because, even though the German and Turkish traffic signs are similar, there are also distinct signs that are only available in Turkey. As part of this study, Turkish traffic sign images are collected to train the model, which are obtained from Google maps. While gathering the pictures, it was taken into consideration that an equal amount of pictures were collected for each class to ensure balance in the dataset. The images are stored in PNG and JPEG formats in various resolutions. The proposed dataset includes 11353 labeled images in a total of 10842 images containing 54 traffic signs. Figure 7 illustrates the proposed Turkish Traffic Sign Dataset classes.

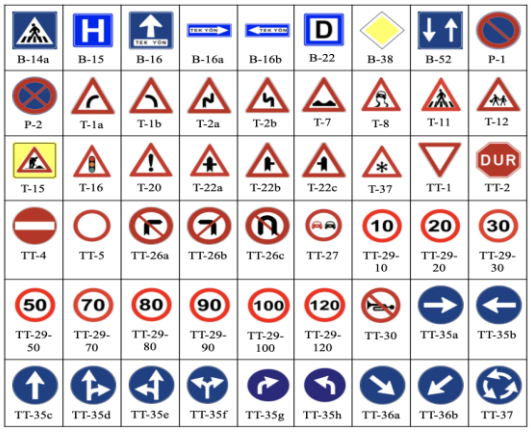

Figure 7. Examples of Turkish Traffic Signs in 54 categories

A total of 10842 images are divided into 80% train data and 20% test data. Two versions of test data are collected to compare the accuracy of the model during day and nighttime.

4.2. Implementation

During the prototype system creation process, the proposed model is trained on Google Colab (Bisong, 2019) with TensorFlow Object Detection API (Huang et al., 2017). The pre-trained Faster R-CNN Inception V2 COCO (Szegedy et al., 2016) model is used to retrain the model using the proposed Turkish Traffic Signs Dataset.

Table. 1 Pre-Trained Models for TensorFlow (Tensorflow, 2018)

Model Name	Speed (ms)	COCO mAP
faster-rcnn-inception-v2-coco	58	28
faster-rcnn-resnet50-coco	89	30
faster-rcnn-resnet50-lowproposals-coco	64	-
rfcn-resnet101-coco	92	30
faster-rcnn-resnet101-coco	106	32
faster-rcnn-resnet101-lowproposals-coco	82	-
faster-rcnn-inception-resnet-v2-atrous-coco	620	37
faster-rcnn-inception-resnet-v2-atrous-lowproposals-coco	241	-
faster-rcnn-nas	1833	43
faster-rcnn-nas-lowproposals-coco	540	-

Table. 1 compares the pre-trained models. For the COCO mAP column, the higher the value, the more successful the model is. Considering the speed and COCO average precision (mAP) values, it is decided to utilize the Faster R-CNN Inception V2 COCO model in this study. The training of the model is conducted twice with step numbers (num_step value) 51,217 (Model-A) and 200,000 (Model-B). 8672 pictures with 9114 class labels are used as train data, and 2170 pictures with 2241 class labels are used as test data. The steps of the implementation process are shown in Figure 8.

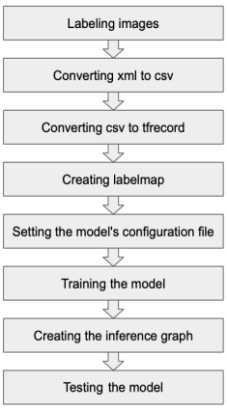

Figure 8. The steps of the implementation process

The labeling process, indicated in Figure 8, involves the creation of an xml formatted file that stores the location of the signs in the image. LabelImg is used to label the images in this process (Tzutalin, 2015). Since the model used in the training requires tfrecord formatted files as input, xml files are first converted to csv and then csv files are converted to tfrecord. After this process, a labelmap is created containing the name and id information of each class to be detected and recognized by the model. After the label map is created, the num_steps value in the model's config file is set. Accordingly, after the required pre-processing is undertaken, the model is trained on a Nvidia Tesla K80 graphic card, using Python programming language and TensorFlow version 1.15.2. In order to use the model after training is carried out, a frozen inference graph is created. After this process is done, the model is ready to be used for classification of traffic signs.

5. Testing and Evaluation

After the model training is carried out with the proposed Turkish Traffic Sign Dataset using the Faster R-CNN Inception V2 COCO model, the newly trained models are evaluated. In order to conduct the evaluation of models, day and night pictures of Turkish traffic signs are collected. Some of these signs are specifically selected since they are exclusively seen in Turkish traffic signs and are not included in the datasets of GTSDB and GTSRB. Stop and one-way traffic signs are examples of these. Test images are taken throughout the day and nighttime. In order to perform the test with more accuracy, extreme attention is paid so that the images are taken at the same place and at the same angle. The test code for testing the models was written using Python and executed on Colab.

Figure 9. Model-A: Recognition trials of the B-14a and the TT-2 traffic signs in daytime and nighttime

Model-A is evaluated with stop (B-14a) and pedestrian crossing (TT-2) traffic signs in daytime and nighttime. The both traffic signs are recognized by the Model-A as can be seen in Figure 9. Model-A recognized the B-14a with 99% accuracy and the TT-2 with 97% accuracy in the nighttime. Also, it recognized the B-14a with 99% accuracy and the TT-2 with 95% accuracy in the daytime.

Figure 10. Model-B: Recognition trials of the B-14a and the TT-2 traffic signs in daytime and nighttime

When Model-B is evaluated with the same traffic signs and under the same conditions, it recognizes both traffic signs with high accuracy, as can be seen in Figure 10. Model-B recognized the B-14a with 100% accuracy and the TT-2 with 99% accuracy in the nighttime. It recognized the B-14a and TT-2 with 99% accuracy in the daytime.

Figure 11. Model-A: Recognition trials of the B-16 traffic sign in daytime and nighttime

Model-A is evaluated for another traffic sign, one-way (B-16) in Figure 11. Since B-16 contains Turkish character, it is critical for evaluation. Model-A recognized the B-16 with 97% accuracy in the daytime. However, it could not recognize the B-16 in the nighttime.

Figure 12. Model-B: Recognition trials of the B-16 traffic sign in daytime and nighttime

Unlike the Model-A, the Model-B recognized the B-16 both in the daytime and nighttime when evaluated with the same traffic sign under the same conditions in Figure 12. Model-B recognized the B-16 with 92% accuracy in the nighttime and 99% accuracy in the daytime.

Figure 13. Model-A: Recognition trials of the B-38 and the TT-29-70 traffic signs in daytime and nighttime

On the side of a vehicle road, the Model-A is examined with a 70 km speed limit (TT-29-70) and priority road (B-38) traffic signs in Figure 13. It recognized the B-38 both in the daytime and nighttime. However, it could only recognize the TT-29-70 in the daytime. Model-A recognized the B-38 with 84% accuracy in the nighttime. It recognized the B-38 with 97% accuracy and the TT-29-70 with %84 accuracy in the daytime.

Figure 14. Model-B: Recognition trials of the B-38 and the TT-29-70 traffic signs in daytime and nighttime

On the other hand, during the evaluation of Model-B, it recognized the B-38 and the TT-29-70 traffic signs both daytime and nighttime as can be seen Figure 14. Model-B recognized the B-38 with 97% accuracy and the TT-29-70 with 99% accuracy in the nighttime. Also, it recognized the B-38 with 100% accuracy and the TT-29-70 with 100 % accuracy in the daytime.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.672
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.851
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.832
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.490
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.683
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.691
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.783
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.785
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.785
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.583
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.770
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.800
```

Figure 15. Average Precision and Average Recall values of the model-A

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.760
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.903
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.893
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.554
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.763
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.778
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.828
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.828
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.828
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.617
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.818
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.844
```

Figure 16. Average Precision and Average Recall values of the model-B

The COCO metrics are used to compare the AP and AR values of the models that as a result of the training. Model-A and Model-B achieved the results in Figure 15 and in Figure 16, respectively. These two models are evaluated on 2170 test data. In order to measure the success of the model during the evaluation, AP and AR values are found and averaged for 10 different threshold values, starting from 0.5 and going up to 0.95 with 0.05 increments. As can be seen in Figure 15 and Figure 16, AP is 67.2% for Model-A and 76% for Model-B. AR is 78.3% for Model-A and 82.8% for Model-B. As a result of the comparison, it is obtained that Model-B is more successful than Model-A.

6. Conclusion and Future Work

This paper proposed a Faster R-CNN Inception V2 COCO based Turkish traffic sign recognition system. As part of this study, the Turkish Traffic Sign Dataset is created by collecting images of Turkish traffic signs. The training of the model is conducted twice with two different step numbers. Models were evaluated with pictures taken in daytime and nighttime. During the evaluation phase, the two models are compared and it is observed that the Model-B is much more successful than the Model-A. In future work, it is aimed to increase the success values of the model for more classes by increasing the number of collected data.

References

Davis, S., & Boundy, R. G. (2020). *Transportation Energy Data Book: Edition 38.2* (No. ORNL/TM-2019/1333). Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).

Bucsuházy, K., Matuchová, E., Zůvala, R., Moravcová, P., Kostíková, M., & Mikulec, R. (2020). Human factors contributing to the road traffic accident occurrence. *Transportation research procedia*, *45*, 555-561.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011, July). The German traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks* (pp. 1453-1460). IEEE.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., & Igel, C. (2013, August). Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *The 2013 international joint conference on neural networks (IJCNN)* (pp. 1-8). Ieee.

Yaliç, H. Y., & Can, A. B. (2011, September). Automatic recognition of traffic signs. In *2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 361-366). IEEE.

Gündüz, H., Kaplan, S., Günal, S., & Akınlar, C. (2013, April). Circular traffic sign recognition empowered by circle detection algorithm. In *2013 21st Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.

CINAR, I., TASPINAR, Y. S., SARITAS, M. M., & KOKLU, M. (2020). FEATURE EXTRACTION AND RECOGNITION ON TRAFFIC SIGN IMAGES. *Selçuk-Teknik Dergisi*, *19*(4), 282-292.

Kilic, I., & Aydin, G. (2020, September). Traffic Sign Detection And Recognition Using TensorFlow's Object Detection API With A New Benchmark Dataset. In *2020 International Conference on Electrical Engineering (ICEE)* (pp. 1-5). IEEE.

Çetinkaya, M., & Acarman, T. (2021, May). Traffic Sign Detection by Image Preprocessing and Deep Learning. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1165-1170). IEEE.

De La Escalera, A., Armingol, J. M., Pastor, J. M., & Rodríguez, F. J. (2004). Visual sign information extraction and identification by deformable models for intelligent vehicles. *IEEE transactions on intelligent transportation systems*, *5*(2), 57-68.

Garcia-Garrido, M. A., Sotelo, M. A., & Martin-Gorostiza, E. (2006, September). Fast traffic sign detection and recognition under changing lighting conditions. In *2006 IEEE Intelligent Transportation Systems Conference* (pp. 811-816). IEEE.

Maldonado-Bascón, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gómez-Moreno, H., & López-Ferreras, F. (2007). Road-sign detection and recognition based on support vector machines. *IEEE transactions on intelligent transportation systems*, *8*(2), 264-278.

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). Ieee.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097-1105.

Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, *8*(1), 1-207.

Phung, V. H., & Rhee, E. J. (2018). A deep learning approach for classification of cloud image patches on small datasets. *Journal of information and communication convergence engineering*, *16*(3), 173-178.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154-171.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*, 91-99.

COCO (Common Objects in Context). (2015) Retrieved August 29, 2021, from https://cocodataset.org/#detection-eval.

Tensorflow (2018) TensorFlow 1 Detection Model Zoo. Retrieved August 30, 2021, from https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md.

Bisong, E. (2019). *Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners*. Apress.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310-7311).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Tzutalin. (2015) LabelImg. Git code. Retrieved August 30, 2021, from https://github.com/tzutalin/labelImg.