EasyChair Preprint

№ 8456

Spotting Railway Signs to Build Smart Decision Support Tools in Railway Management Systems

Pramuka Weerasinghe, Mohamed Shaheer, Rochana Rumalshan, Prabhath Gunathilake and Erunika Dayaratna

July 12, 2022

Spotting Railway Signs to Build Smart Decision Support Tools in Railway Management Systems

Abstract*— Railway being an important mode of transportation, it demands highly precise management and decision support as it is extensively used for both commuter and cargo transportation. Also, it is considered as a salient element in smart city and modern infrastructure planning. Railway track diagrams are available with service providers in portable document format where a single document consists of information from one station to another, including information regarding the tracks, signals, crossovers, switches, and their location details denoted using a standard set of symbols and drawn using a computer aided tool. A Management tool that has all details of individual symbols is an important tool for decision support systems. This research focuses on developing an automated system to extract this information based on deep learning techniques. The method consists of two steps: object detection and optical character recognition (OCR). State of the art Convolutional Neural Network (CNN) architectures are used to perform object detection. They include single stage detectors like YOLOv3 and SSD and two stage detectors like Faster-RCNN and RFCN. Among the selected, RFCN resulted in the highest accuracy with the minimum loss value of 0.22, compared to other methods. This is because of RFCNs architecture catering small object detection by dividing the image into small feature maps. Then, OCR is performed on detected Regions of Interest (RoI) to extract and store the text in a dedicated database which has the information of all the signs along with their location details. Image processing techniques such as template matching and Neural Network (NN) based OCR is tested here. Out of these two approaches, NN based technique outperformed template matching drastically with more than 50% accuracy.

Keywords—object detection, railway sign detection, railway management systems, decision support tools

I. INTRODUCTION

In today's world, effective transportation management systems are really important as it directly affects the general public and many industries. This can be vital in many aspects for an instance, at a state of emergency in order to make feasible decisions not just experience and expert knowledge but having decision support data and tools are important. In the domain of transportation, railway services have a significant place around the globe where cargo and passenger trains are used to transport goods and people across cities. Unlike road transportation, infrastructure and resources are limited in this domain where expansion is possible at a higher cost. Therefore, utilizing the resources and managing the system effectively is a prior concern.

There are railway track diagrams available, which include all the information regarding the tracks, signals, crossovers, switches and their location details denoted using a standard set of symbols and drawn using a computer aided tool. These diagrams are in a portable document format (PDF) where a single document contains the track details from one station to another. Even though this information is available, there is no single database that has all details of these individual symbols. This research focuses on developing an automated solution to use these available PDF documents and create a database, which includes information on individual symbols in order to build a decision support tool. The main approach is to extract the document into images and use state of the art deep learning techniques to detect symbols.

II. BACKGROUND

Deep learning approaches based on convolutional neural networks (CNN) have been successful in image classification and object detection. CNNs have the capability to extract localized features and perform image classification better than other neural network architectures. Object detection aims at locating and classifying existing objects in any image, and labeling them with rectangular bounding boxes to show the confidences of existence. Object detection methods can mainly be categorized into two types: (i) Region proposal based (two stage object detection), where a two-step process, matches the attentional mechanism of human brain to some extent, which gives a coarse scan of the whole scenario firstly and then focuses on regions of interest. (ii) Classification based (single stage object detection), where one-step frameworks based on global regression/classification, mapping straightly from image pixels to bounding box coordinates and class probabilities, can reduce time expense. In this research we will be using both region proposal technique based architectures and classification based architectures to detect symbols in rail track diagrams. Figure 1 shows some popular network architectures that are used in this study.

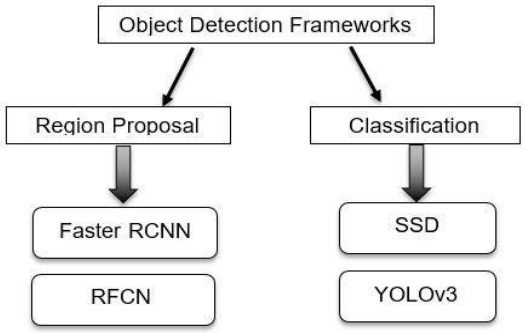

Fig. 1. Neural network architectures used in object detection

A. Faster RCNN

The motivation behind R-CNN's is to tackle the problems with bounding box issues [1]. Given a specific image, the need of having an option to draw bounding boxes over all identified objects is solved by R-CNN methods. The method consists of two stages namely the region proposal step and the classification step. Basic R-CNN strategy runs a neural net classifier on samples taken from image data utilizing remotely computed bounding box proposals. The methodology costs much in computation. Fast R-CNN decreases the calculation by doing the feature extraction just a single time to the entire image. Faster R-CNN goes beyond and utilizes the extracted features to make class-related bounding box proposals.

B. SSD

Single Shot Multi-Box Detector (SSD) [2] differs from the R-CNN based methodologies by not needing a second stage for each region proposal. This makes it quick enough for ongoing real-time applications. Nonetheless, this accompanies a cost of decreased accuracy. Here the key thought is a single network (for speed) and no requirement for region proposals rather it utilizes bounding boxes and afterward changes the bounding box as a component of prediction.

C. RFCN

In contrast to previous region-based detectors such as Fast/Faster R-CNN that apply a costly per-region subnetwork hundreds of times, region-based detector is fully convolutional with almost all computation shared on the entire image [3]. To achieve this goal, a position-sensitive score maps to address a dilemma between translation-invariance in image classification and translation-variance in object detection. This method can thus naturally adopt fully convolutional image classifier backbones, such as the latest Residual Networks (ResNets), for object detection.

D. YOLOv3

You Only Look Once version 3 is the available stable & fastest method among YOLO versions proposed yet and it works on different principles than the before- mentioned R-CNN models. Like in SSD, this runs a single convolutional network on the whole input image (once) to predict bounding boxes with confidence scores for each class simultaneously [4]. The advantage besides the simplicity of the approach is, the YOLO model is fast (compared to Faster R-CNN and SSD) and it learns a general representation of the objects. However, this increases the localization error rate. Another drawback in this model is that it performs poorly with images with new aspect ratios or small object flocked together, but it reduces the false-positive rate. In the overall scenario, this method is fast in predicting results.

III. LITREATURE REVIEW

With the advancement of Deep Learning and Computer Vision the area of Object Detection has gained a lot of success over the recent years. Many new models have been introduced to outperform the state of the art and achieve a high degree of accuracy in detecting objects efficiently. Although this is the case there are still areas where object detection can improve specially when the object size gets smaller and also when the environment conditions change. Many object detection frameworks have been introduced to provide satisfactory results and obtain high degrees of accuracy.

Frank D. Julca-Aguilar in 2017 proposes a method to detect handwritten symbols using Faster R-CNN object detection algorithm [5]. In their study they discuss the issues relative to the handwritten nature of data. Their results show that Faster-RCNN can be effectively used on both publicly available flowchart and mathematical expression (CROHME-2016) datasets. Guo X. Hu in 2018 proposes an effective approach to detect small objects by extracting features at different convolutional levels of the object and using multi-scale features to detect small objects [6]. In their results they show that their accuracy in detecting small objects is 11% higher than the state-of-the-art models. As previous studies have not looked to detect small objects such as railway symbols which has a very specific shapes and sizes. In this study we look to implement different object detection models and compare their results while showing the state-of-the-art model for small object detection in the railway domain.

IV. METHODOLOGY

A. Dataset

As the dataset, first we had a PDF of Railway signal diagrams that was constructed using a CAD (Computer Aided Design) tool. In order to obtain individual images of the dataset, railway signal diagram images were extracted from the PDF and saved into a separate directory in .jpg format. The dataset contained 60 railway signal diagram images, which consists of four main symbols. Figure 2 shows a sample image used for training and validation where Figure 3 shows the list of symbols/objects to be detected. The mileposts are labelled as a single object where the detected region will be used to perform optical character recognition and store the milepost details for respective symbols in a database.

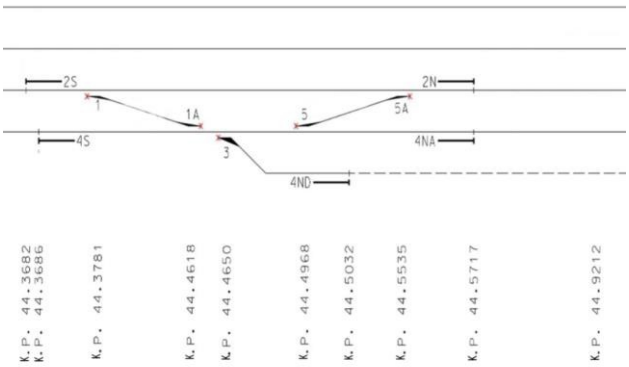

Fig. 2. Sample Railway Signal Diagram

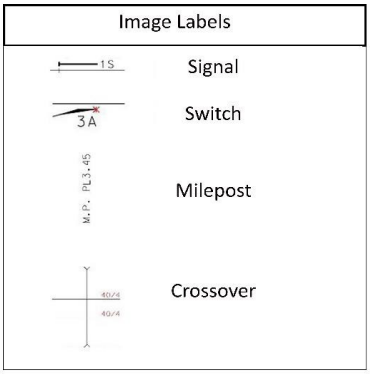

Fig. 3. Railway Signal Diagram labels

B. Image Labelling

The railway signal diagram images extracted is labelled using a labelling/annotation tool where each symbol of interest is marked with a corresponding bounding box and a corresponding label is assigned to each symbol. After labelling the images the labelled data annotations was exported into Extensible Markup Language (XML) format where a corresponding XML file was created for each labelled image. The XML file contains the label names and the coordinates of the bounding boxes that were drawn for each symbol in the image.

C. Experimental Setup

The dataset was separated as train and test sets where 2/3rd of the images were taken for training and 1/3rd of the images were taken for testing. For the training environment, we used Google Colaboratory which is a free online GPU service. The dataset was trained on an Nvidia Tesla K80 GPU with 12GB of RAM. Due to the scarcity of training data due to confidentiality concerns, Transfer Learning approach has been used.

D. Use of Transfer Learing

Transfer learning is a technique that reuses already trained models on a new problem. In transfer learning, the information on a previously prepared machine-learning model is applied to an alternate yet related issue. For instance, if you prepared a basic classifier to anticipate whether a picture contains a bus, you could utilize the information that the model picked up during its preparation to perceive different items like a truck.

With Transfer learning, we fundamentally attempt to exploit what has been realized in one errand to improve the speculation in another. We move the weights that a network has learned at "task A" to another "task B". The overall thought is to utilize the information a model has gained from an undertaking with a great deal of accessible labeled data, in another errand that does not have many training data. For training we implemented the Tensorflow Object Detection API, which has pre-trained, models in-built known as the model zoo. These pre-trained models we trained on the popular MS COCO dataset. MS COCO is a large-scale object detection segmentation and captioning dataset.

E. Faster-RCNN, SSD and RFCN based object detection

Faster RCNN: Faster-RCNN is one of the most popular region proposal based object detection networks [1]. It consists of two networks: Region Proposal Network (RPN) for generating region proposals and a network using these proposals to detect objects. Faster-RCNN showed good detection results on the famous PASCAL VOC 2007 test set giving high mAP (mean average precision %) values. Our study uses Faster-RCNN with a batch size of twelve, with thousand training steps and fifty testing steps together with the tensorflow object detection API (framework for creating a deep learning network that solves object detection problems) for training and evaluating the object detection model.

SSD: SSD is a single-shot-detector, which has no delegated region proposal network and predicts the boundary boxes and the classes directly from feature maps in one single pass. SSD showed new records in performance and precision for object detection tasks, scoring over 74% mAP at 59 frames per second on standard datasets such as PASCAL VOC and COCO. Our study uses SSD with a batch size of twelve, with thousand training steps and fifty testing steps together with the tensorflow object detection API (framework for creating a deep learning network that solves object detection problems) for training and evaluating the object detection model.

RFCN: RFCN is a Region-based Fully Convolutional Network, which uses a method known as position-sensitive-ROI-pool, which is similar to the ROI pool in Fast R-CNN. In position-sensitive-ROI-pool process, it maps the score maps and RoI's to the vote array. After calculating all the values for the position-sensitive ROI pool, the class score is taken as the average of all its elements. Both Faster RCNN and RFCN uses ResNet 101 for feature extraction. And RFCN performs 20 times faster than R-CNN giving higher mAP (mean average precision %). Out study uses RFCN with a batch size of eight, with thousand training steps and fifty testing steps together with the tensorflow object detection API (framework for creating a deep learning network that solves object detection problems) for training and evaluating the object detection model.

YOLOv3: One of the state-of-the-art, one-stage object detector, YOLOv3, has been implemented as a customized version for this particular problem scenario using the generic implementation which is thoroughly explained by

(Pylessons, 2018) at their tutorial series and the implementation which is provided at Git repository [1] and it has been modified to address the problem scenario with the custom dataset and the requirements.

Annotations file conversion: XML to YOLOv3 file structure. Labeled data set annotations have been exported as XML data and those were needed to convert into a YOLOv3 implementation understandable single file format as follows. Each row contains all the labeled bounding box for a single image. To train the custom object detection model it is required the annotations file and class file. Both of these files have been created with an external single script.

TRAIN_INPUT_SIZE = 416
TRAIN_DATA_AUG = True
TRAIN_TRANSFER = True
TRAIN_LR_INIT = 1e-4
TRAIN_LR_END = 1e-6
TRAIN_WARMUP_EPOCHS = 2
TRAIN_EPOCHS = 300

F. Optical Character Recognition

In order extract the required milepost text our first approach looked at implementing optical character using template matching where a template image for each character including Numeric, Alpha Numeric and Special Symbols was given and that image was used as a template so that the cropped image of the milepost that is extracted is matched using image processing techniques with the template to recognize similar characters

The second approach looked at using a neural network based Optical Character Recognition using open source libraries. By implementing this approach together with image processing techniques we were able to extract the milepost texts as required.

V. RESULTS

A. Object Detection

The following are the results obtained after training railway signal diagram images and building the models using Faster-RCNN, RFCN, SSD and Yolov3 detecting the railway signal diagram symbols.

Faster-RCNN, SSD RFCN were all trained on 1000 steps with 24 training images where Faster-RCNN with an initial learning rate of 0.00002 and a learning rate of 0.000002 at the end, SSD with 24 training images with an initial learning rate of 0.004 using RMSprop Optimizer and RFCN with an initial learning rate of 0.003 and a learning rate of 0.000003 at the end. The Loss Graphs for Faster-RCNN, SSD and RFCN are shown in Figure 4. Table I. shows the results obtained from Faster-RCNN, SSD and RFCN where accuracies of detection box precision and recall with small and large mAP and loss values are shown.

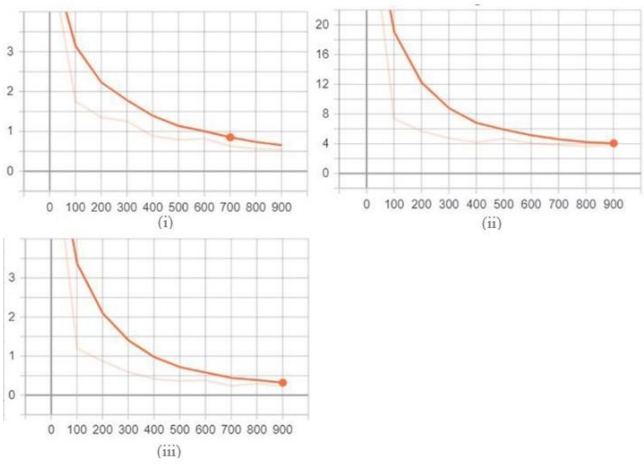

Fig. 4. Loss graphs of (i) Faster RCNN (ii) SSD (iii) RFCN

TABLE I. RESULTS FOR FASTER RCNN, SSD AND RFCN

	Faster RCNN	SSD	RFCN
Detection Boxes Precision/mAP	0.39	0.02	0.41
Detection Boxes Precision/mAP (large)	0.38	0.03	0.49
Detection Boxes Precision/mAP (medium)	0.41	0.00	0.29
Detection Boxes Precision/mAP@.50IOU	0.75	0.07	0.83
Detection Boxes Precision/mAP@.75IOU	0.40	0.00	0.37
Detection Boxes Recall/AR@100 (large)	0.55	0.04	0.55
Detection Boxes Recall/AR@100 (medium)	0.49	0.00	0.47
Loss/Box Classifier Loss/Classification	0.22	9.94	0.28
Loss/Box Classifier loss/Localization Loss	0.25	4.36	0.38
Loss/RPN Loss/Localization Loss	0.13	θ	0.13
Loss/RPN Loss/Objectness Loss	0.04	-	0.02
Loss /Total Loss	0.63	14.5	0.82
Final Loss	0.30	3.88	0.22

In YOLOv3, the loss function of YOLOv3 can be summarized as follows. Confidence loss determines whether there are objects in the prediction frame (conf_loss). Box Regression loss, calculated only when the prediction box contains objects (giou_loss). Classification loss, determine which category the objects in the prediction frame belong to (prob_loss).

The YOLOv3 model was trained with 2400 steps with 57 training images. Initially the learning rate was set to 0.0001 and it was 0.000001 at the end. Fig. 5. Shows the loss graphs plotted over number of steps and the Table 2 shows the results obtained with YOLOv3 model.

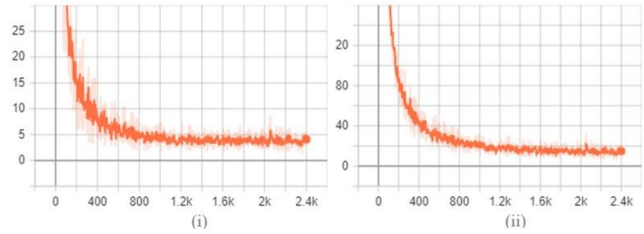

Fig. 5. Loss graphs of YOLOv3 (i) Classification loss (ii) total loss

TABLE II. RESULTS OBTAINED WITH YOLOV3

	giou loss	conf loss	prob loss	total loss
Training	8.81	2.88	4.22	15.91
Validation	5.63	4.01	4.47	14.11

B. Optical Character Recognition

For extracting characters in the railway signal diagrams, two approaches are used. Template matching based approach and also a neural network-based architecture together with a LSTM implementation. Character detection accuracy using template-matching technique is 42.5% and the neural network-based character detection accuracy is 92%.

VI. CONCLUSION

According to the results obtained, we can see that all the models (Faster-RCNN, SSD, RFCN and YOLOv3) mostly are better at detecting larger objects than smaller ones. When considering the final loss values, we can see that RFCN outperforms the other model architectures. YOLO and SSD, which are single stage detectors they are better at detecting real time images with less inference time, we can see that the accuracies are less compared to region proposed networks. Here we have tried two region proposed networks namely Faster-RCNN and RFCN. Both of these networks outperform the single stage detectors even though their inference time is comparatively higher. In this scenario we are more concerned about the accuracy than the inference speed therefore we can arrive at a conclusion that region proposed networks are more suitable network architectures for detecting railway signal diagrams. Here we can see that RFCN has the best detection accuracies as they divide the image into small feature maps and create a voted array for each of these feature maps as a result it is able to easily identify smaller objects better than other architectures as it focuses on all the smaller areas of the image rather than on a specific region. For optical character recognition, we have implemented two different techniques. The template matching approach was drastically outperformed by the neural network-based approach.

REFERENCE

[1] Ren, Shaoqing & He, Kaiming & Girshick, Ross & Sun, Jian. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39. 10.1109/TPAMI.2016.2577031.

[2] Liu, Wei & Anguelov, Dragomir & Erhan, Dumitru & Szegedy, Christian & Reed, Scott & Fu, Cheng-Yang & Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. 9905. 21-37. 10.1007/978-3-319-46448-0_2.

[3] Dai, Jifeng & Li, Yi & He, Kaiming & Sun, Jian. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks.

[4] Redmon, Joseph & Farhadi, Ali. (2018). YOLOv3: An Incremental Improvement.

[5] Julca-Aguilar, Frank & Hirata, Nina. (2018). Symbol Detection in Online Handwritten Graphics Using Faster R-CNN. 151-156. 10.1109/DAS.2018.79.

[6] Hu, Guo & Yang, Zhong & Hu, Lei & Huang, Li & Han, Jia. (2018). Small Object Detection with Multiscale Features. International Journal of Digital Multimedia Broadcasting. 2018. 1-10. 10.1155/2018/4546896.