Identifying universal safety signs using computer vision for an assistive feedback mobile application

Alankrit Mishra*, Nikhil Raj†, Shubham Bodhe‡, and Garima Bajwa§
Department of Computer Science
Lakehead University
Thunder Bay, Ontario, Canada
Email: {*amishra1, †nraj, ‡bodhes, §garima.bajwa}@lakeheadu.ca

***Abstract*—When walking on the road or navigating unfamiliar areas, the elderly, visually handicapped, and persons with hearing loss encounter challenges. The information on numerous safety signs such as road signs, traffic signs, workplace safety signs or industrial hazard signs is usually unhelpful to them. This paper proposes a solution that can detect a comprehensive set of safety signs in real-time. Our app runs a deep learning model pre-trained on a custom-built dataset. The deep learning model we have used is explicitly built for object detection to find regions of interest, create appropriate bounding boxes, and classify the signs with three different levels of severity - Danger, Caution, and Prohibitory. The camera-equipped smartphone relays haptic or audio feedback upon the successful detection of a safety sign.**

***Index Terms*—assistive, computer vision, detection, mobile application, safety sign dataset**

I. Introduction

Nowadays, 90% of the population carries a smartphone with more computation power than Apollo 11 that went to the moon. This population includes the elderly, visually impaired, and deaf. We can leverage this computing power to help the elderly and differently-abled populous with the help of object recognition. We can detect objects of interest or regions of interest in a given image or video and generate a classification output using object recognition.

We propose to use object detection to recognize safety signs such as traffic signs, road signs, and workplace safety signs. Depending on the severity of the signs (three custom levels), we generate appropriate haptic or audio feedback. Our pre-trained model runs on a smartphone with a suitable camera to input video for detection. The app can help both deaf and blind people with haptic feedback. Detection of danger signs triggers audio feedback inspiring passers-by to come to a halt and assist those in need.

We also observed a scope of improvement in the indoor and industrial hazard safety sign recognition and detection research.

II. Literature review

People with visual impairments typically lack access to visual cues such as informative signs, landmarks, and structural characteristics that people with normal vision rely on for navigation [1]. The app iNavigate developed by the authors combined a digital map of the environment with computer vision and inertial sensing to estimate a user's location in real-time. This approach required a small number of training images data and improved sign detection leveraging the YOLOv5 [2] pre-trained model. Two image datasets were used in their experiments. The first dataset was acquired to evaluate the effectiveness of the YOLOv5 object recognition algorithm and the second one to assess the distance estimation algorithm. Their approach allowed real-time detection of multiple indoor sign types with distance and signs orientation to estimate a user's location.

Serna and Ruichek [3] used a real-world dataset from six European countries for traffic sign classification. The dataset had four categories - danger warning signs, regulatory signs, informative signs, and others. Danger signs warn road users about the nature of the potential danger ahead on the road. In contrast, regulatory and informative signs inform commuters about the restrictions and other appropriate information, respectively. Lastly, the others categories enlighten road users about any critical situation.

According to Hasegawa et al. [4], it is necessary to detect traffic signs relatively far away from the point of view. Another fact worth noting is that scale changes occur as an individual approaches a sign, and the weather could also affect the detection process. The authors also used YOLOv2, a high-speed CNN as their model to further improve optimizations. Their method divided the input into SxS regions to predict the bounding box which consisted of five elements. They also used different weights assigned to the coordinates and the number of objects present in the frame. The model was trained on varying sizes of traffic signs to be robust to scale changes. They constructed their dataset in video and used data augmentation to get better accuracy. In clear weather, they achieved 94.6% precision, whereas at night they achieved 83.7% which was significantly higher than the methods used in other papers.

Colour based traffic sign detection usually transform RGB images into different colour spaces to extract the traffic signs based on colour thresholding. Shape-based detection considers various geometric contour shapes of traffic signs. However, these methods take a great deal of computational complexity and a suitable illuminated environment. Xianghua et al. [5] proposed a different solution to solve this issue. They used methods like adaptive thresholding with cumulative distribution function of image histogram and maximum-

978-1-6654-6603-5/22/$31.00 ©2022 IEEE
DOI 10.1109/IRI54793.2022.00067

minimum normalization method to process images according to the threshold [5]. Using this method, they overexposed the foreground, which tended to be the traffic sign and suppressed the background. They also used morphological filtering in the connected domain of the maximum-minimum normalization. They achieved an accuracy of 93.96%.

Chakraborty and Chiracharit [6] focused on the use of convolutional neural networks to detect washroom signs for visually impaired people. The noticeable common signs in washrooms and public places are generally in the shape of a human being. The authors constructed their dataset using a mobile phone camera. While pre-processing, they converted RGB images to grayscale and then filtered out the noise. They also used histogram equalization to equalize the intensity of each pixel in the image. The authors used maximally stable extreme region (MSER) algorithm for detecting regions of interest, which also happens to be a blob detection method. For classification purposes, they used a CNN trained on 2000 images [6]. The authors used a relatively simple seven-layer CNN classifier to detect three different types of washroom signs. Even after numerous optimizations, their method achieved only 90% accuracy.

Although Kantawong [7] worked on the classification of hazardous and fire exit signs, it presented interesting findings in terms of pre-processing the dataset to increase machine understanding. By using simple techniques like image binarization, increasing the contrast, and thinning the edges, the author was able to decrease the complexity of the task. The photos were converted to a 52-digit binary number to process them with a basic back-propagation neural network (BPN). It avoided the need for a convolutional neural network (CNN) and reduced the complexity. However, since our focus is object recognition from complex images, the usage of a simple image processing steps and BPN is undesirable. Furthermore, the author used flat images of hazard and fire exit signs for classification, which contrasts our usage of real-life traffic signs, hazard signs and workplace safety signs.

Another study demonstrates the usage of an attention-based model pre-trained on Imagenet [8]. This model detects traffic signs by merging the Faster R-CNN with the attention mechanism, which is a consideration for our future work.

A. Contributions

- We developed a model that recognizes a universal set of safety signs such as those found in traffic, on the road, in the workplace, and indoors. Our model determines the severity level of a safety indicator regardless of the type of sign supplied as input (danger, caution, or prohibitory). As a result, helping the elderly, visually impaired people, or people with hearing loss in their day-to-day navigation.
- We created a custom dataset (available upon request) [9] and organized the annotations based on the baseline semantics as our requirement (i.e., defining and segregating on the level of severity), which included images taken in various lighting situations, exposure settings, and equipment, providing a feature rich dataset.
- We compared the mAP (mean Average Precision) of various object detection models on both the PASCAL VOC and our custom Safety Sign Dataset (Table II). We have also compared our approach to previous safety sign studies and our paper's accommodation of a comprehensive dataset (Table III).
- We developed a proof-of-concept Android app called "Safety Sign Detection" and performed real-time tests in various naturalistic scenarios, delivering priority-based haptic feedback after detection (supplementary material [1]).

III. Methodology

The usage of bounding box based object detection was more suitable for our purposes than complex image segmentation techniques to ensure faster training and processing times. These bounding boxes are called image annotations, and in our case, they consisted of two coordinates and the width and height of the bounding box, producing a rectangle. We used three different convolutional neural networks - SPP-net [10], YOLOv3 [11], YOLOv5 [2] - that we had reviewed to determine the one that performed the best and ultimately used it in our application.

YOLOv5 [2] is based on continuous improvement of YOLO to YOLOv4 and has got a benchmark accuracy in object detection datasets. It uses CSPDarknet as the backbone by incorporating cross-stage partial network (CSPNet) [12] into Darknet.

As proposed in our objective, the final model should be portable to be used standalone with a small application interface. Taking this requirement into consideration, we compared lightweight models like YOLOv5 [2], YOLOv3-tiny [11], and SPP [10]. Ultimately, we deployed the YOLOv5 [2] model trained with our dataset to the Android application due to its performance, as will be presented in later sections.

For real-time object detection with video input, we used a real video with various outdoor and indoor safety signs, and our model detected the class of the safety sign. Once the sign is detected, the application sends a haptic or vibration feedback to the device that was running the application (Figure 1).

IV. Experiment Setup

A. Dataset

We focused on building our own custom dataset. *In our review, we found no comprehensive dataset that consists of various types of safety signs such as hazard signs, cautionary signs, workplace safety signs, road signs and so on.* We assimilated images that described different categories of safety signs and then divided them into suitable classes to give three types of warnings. Our dataset is an amalgamation of workplace safety, cautionary signs, traffic, and road signs datasets [13]–[15] that are available publicly. Additionally, we performed web-scraping to increase our sample size. Our dataset contains

[1]A video demonstrating our mobile application can be accessed at: https://tinyurl.com/irisupplement

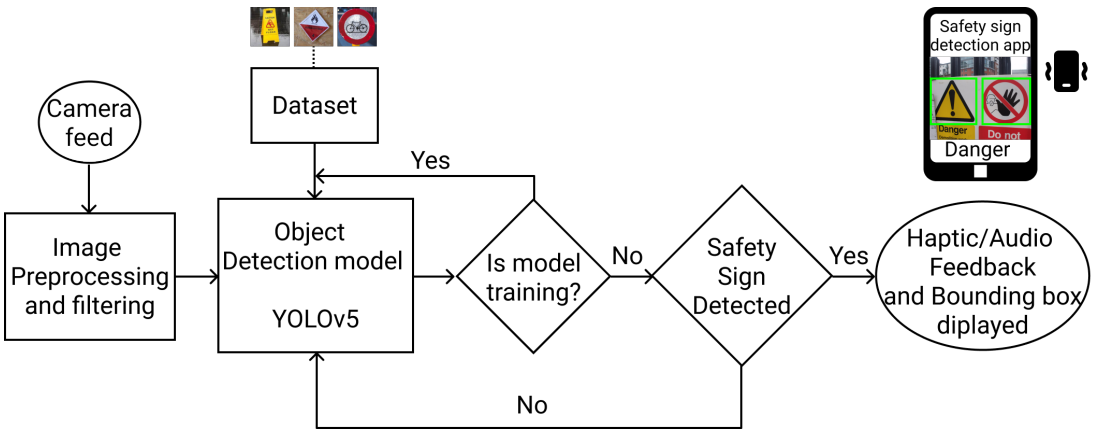


Fig. 1: System architecture

312 images with 458 manual annotations (Table I) taken under different conditions concerning the region of interest, such as lightning, exposure, and distance to the object of interest. Moreover, the cameras used for the scraped images is different for each, thus, adding more variation in the feature set.

(a) Danger class image 1

(b) Danger class image 2

Fig. 2: Sample images from class 'Danger'

Upon reviewing previous work done in this domain, we divided our dataset into three categories as follows;

1) **Danger:** It is generally indicated with a solid red background having a symbol and/or text in the middle (examples shown in Figure 2). This was the easiest to segregate due to a clear description.
2) **Caution:** It consists mostly of signs that have a solid yellow or an orange background with a symbol and/or text in the middle (examples shown in Figure 3). There can also be situations that may include a black dashed border on the sign itself.
3) **Prohibition:** It is usually denoted by a white background accompanying red border with a symbol and/or text in the middle (examples shown in Figure 4). The signs may also consist of a stroke going through the center if the shape of the sign is a circle.

The above classes gave us a good distribution while also suitable labels for a real-time application. However, we realized that traffic signs of different countries did not adopt this methodology. In the end, we relied on the descriptions given in the datasets to segregate our custom dataset into these three classes. Another critical issue was that we needed annotations or ground truths for the object detection, and these were made available by only a few of the datasets we had selected. As a result, we faced some challenges with annotations (discussed in Section VII).

(a) Caution class image 1

(b) Caution class image 2

Fig. 3: Sample images from class 'Caution'

(a) Prohibitory class image 1

(b) Prohibitory class image 2

Fig. 4: Sample images from class 'Prohibitory'

B. Data Pre-processing

After collecting the dataset, we moved to label these images as per the threat level – danger, caution, or prohibitory. For this, we leveraged the makesense.ai platform [16]. It is an open-source platform that does not require installation and runs on a browser that supports JavaScript. We uploaded our dataset and annotated the images with appropriate class labels. Later, we downloaded two versions of the annotations - text and CSV files. Text annotations facilitated the training of YOLO based models, and a CSV file facilitated the training of the SPP model.

As mentioned earlier, our dataset was an amalgamation of multiple datasets, and this posed unique challenges in terms of inconsistency in image dimensionality. After annotating the images, we faced the daunting task of resizing the images to a

uniform size without negatively impacting the image annotation. So, we took the assistance of the Roboflow platform [17]. Roboflow is a developer framework for Computer Vision that enables improved data collection, pre-processing, and model training approaches. The platform also supports pre-processing data workflows like changing image orientations and resizing and contrasting. However, we used Roboflow to resize the images to 640 by 640 pixels without any repercussions on the annotations. Later, we split the dataset into 75%-25% for training and validation purposes on Roboflow itself.

C. Model Training

After organizing the labelled dataset into a train-validate split, we initiated the model training process. The SPP, YOLOv3, and YOLOv5 models were used due to their accuracy and performance of object detection in many applications. After training, we compared the confusion matrix of the three models and found YOLOv5 to be the best performer. We saved the weights of the custom trained YOLOv5 model to be used later for object detection. We performed a preliminary testing of the object detection using a webcam.

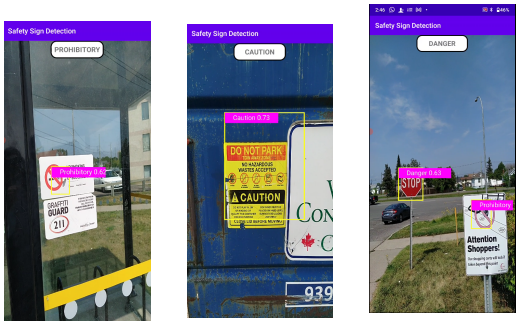
Since the project aimed to provide haptic feedback upon threat detection, we decided to build an Android application, and leverage the native camera and vibration features to achieve this. Before building a native Android application, we had to find a way to utilize the trained weights in the Android platform. Pytorch allows to seamlessly go from training a model and deploying it while staying within the Pytorch Ecosystem [18]. It provides us with a Pytorch Mobile version with an efficient mobile interpreter for Android and IOS platforms. We trained our model on Google Colab to utilize its GPU runtime environment for our application. We expected a few hiccups while running on a low powered computationally limited edge device compared to its desktop counterpart. Hence, an essential step was to convert the Python-dependent model to TorchScript. Pytorch provides us with a utility called *optimize_for_mobile* that optimizes the model for mobile use. TorchScript version 1.9.0, based on the same version on Pytorch, was used in our Android application [19].

D. Application Development

We built an Android app, "Safety Sign Detection" (supplementary material[2]), to demonstrate the objection detection capability. The optimized model was packaged inside the application and it was loaded using a *LiteModuleLoader* [20] [21]. The application sent the live camera frame at every 250 ms. This gap was reasonable to detect the objects, achieve a quick feedback post detection, and avoid performance bottleneck issues. Our app labelled detected signs with bounding box and confidence score.

We discovered that a confidence score of 50 was a suitable threshold during our testing. In a possible scenario where the model detects objects of two or more classes with greater than 50 % confidence, a feedback was given for the most severe sign while showing the bounding boxes for all the detected classes. The application generates a distinct vibration pattern based on the the level of severity of a sign. If a danger signal is detected in the feed, the device will produce a short vibration (200 ms), a pause (400 ms), and then a long vibration (1000 ms). It will generate two brief vibrations (200 ms each) with a 400 ms delay in between for caution signs and a single 500 ms vibration in the case of prohibitory signs. When the signs are detected again in the next frame, the vibrations will reoccur after a 400 ms delay between each feedback. There is a speech feedback of "danger detected" every time it detects a danger sign. Speech feedback plays at the interval of 5000 ms if the model detects a danger sign for two consecutive frames.

App maintains a low confidence score for object detection to avoid false positives. This prevents missing a safety sign when it is present and keeps a person from entering potentially lethal situations.



(a) Prohibitory (b) Caution (c) Danger and Prohibitory

Fig. 5: Screenshots showing a live output of our Safety Sign Detection android application giving feedback for the various signs identified by the model.

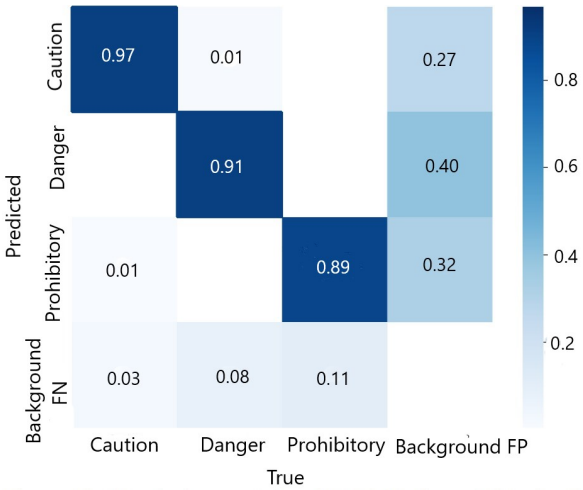


Fig. 6: Confusion matrix of YOLOv5 model trained our dataset

[2]A video demonstrating our mobile application can be accessed at: https://tinyurl.com/irisupplement

TABLE I: Performance of the YOLOv5s model on our custom dataset.

Class	Annotated Labels	Precision	Recall	mAP@.5	mAP@.5:.95
All	458	0.966	0.915	0.948	0.765
Caution	190	0.964	0.947	0.967	0.785
Danger	102	0.979	0.912	0.946	0.767
Prohibitory	166	0.954	0.886	0.931	0.744

TABLE II: Comparison matrix of mAP (mean average precision) for PASCAL VOC dataset with our Safety Sign dataset.

Model	Dataset	mAP@.5	mAP@.5:.95
YOLOv1 [22]	PSACAL VOC [23]	-	0.634
Fast - RCNN + YOLO [22]		-	0.707
SPP [10]		-	0.592
Fast RCNN [24]		-	0.7
YOLOv3 [11]		-	0.836
YOLOv5 [ours]	Safety Sign Dataset [9]	0.948	0.765
YOLOv3 [ours]		0.908	0.646
SPP [ours]		0.911	0.662

TABLE III: Comparison matrix our comprehensive approach with respect to previous research on safety sign detection.

Paper	Dataset type	Approach/Model	Accuracy	Testing Environment
Real-time sign detection for indoor navigation [1]	Indoor Safety sign	YOLOv5	0.98	Phone app iNavigate
Japanese traffic sign detection [4]	Traffic Sign	YOLOv2	0.95	Generic
Smart data-driven traffic sign detection [5]	Traffic Sign	Shape-based detection	0.94	Generic
Washroom sign detection [6]	Indoor Safety sign	Seven-layered CNN classifier	0.9	Generic
Hazardous signs and fire exit signs classification [7]	Workplace safety signs	Binary encoded images with BPN	NA	Generic
Traffic sign detection [8]	Traffic sign	Attention-based NN	0.92	Generic
Safety Sign detection (Our)	Comprehensive*	YOLOv5	0.95	Android app**

*This dataset includes traffic, indoor and workplace safety signs.
**Gives haptic feedback based on the level of safety.

V. Results

We collected all parametric results after training the YOLOv5 model with our dataset freezing the backbone with pre-trained weights.

During training, we observed that the loss reached zero and there was no significant improvement after 90 epochs. We terminated the training process at 100 epochs to avoid over-fitting. Table I shows the results of the experiment run mentioned above with 100 epochs for the validation set after training.

We observed that the model was trained well with mAP (mean average precision) at IOU (Intersection over Union) 0.5; the graph started flattening after 40 epochs. However, to ensure model optimization at IOU 0.5:0.95, we continued the training up to 100 epochs; the mAP graph started to flatten after 70 epochs in this case.

From Table I, we can infer that there are 458 labels (190 caution, 102 danger, and 166 prohibitory) for 312 images. The mAP at 0.5 IOU was almost in level for all classes, indicating that the model was trained uniformly at IOU 0.5. However, at IOU 0.5:0.95, the prediction of class prohibitory suffered a little.

Figure 6 illustrates the confusion matrix for the YOLOv5 model. We also discovered that the prohibitory class had a lower accuracy of 0.89 than the other two classes, possibly because of the lack of labels. We also discovered several false positives for the background, which inhibited model training and impacted the overall prediction of all classes.

VI. Discussion

Table II presents the comparative results for the experiments carried out in this paper with our custom safety sign dataset and those achieved in previous works using the PASCAL VOC dataset [23].

The baseline mAP of the initial YOLO model for object detection in the PASCAL VOC dataset was 0.634. The paper [22] also combined Fast-RCNN with the YOLO layer and obtained an improvement in mAP (0.707), which is close to the mAP Fast-RCNN paper [24]. As we stated in our previous Section III, SPPnet did an excellent job of lowering the model's complexity as no fine-tuning was necessary. However, the model's standalone performance with mAP 0.592 fell short of expectations.

YOLOv3, as shown in Table II, improved the mAP score of object detection PASCAL POC dataset to a significant level, with Darknet [11] backbone. We chose the YOLOv5 model to train our dataset because it is enhanced and built on YOLOv3's Darknet and SPP, making it a lightweight yet optimized model to provide better performance.

After comparing the performance of the above object detection models, we decided to train our dataset with YOLOv5, YOLOv3 and SPP models. As we mentioned in Section IV, our dataset was relatively small and these three models were the best fitted for a dataset like ours. Table II also depicts the mAP scores of these models trained on our dataset. We found that YOLOv5 outperforms SPP and YOLOv3, leading with the highest mAP score of 0.765 at IOU 0.5:0.95. Therefore, we deployed the trained weights of the YOLOv5 model in our

Android application for real-time object detection.

VII. Challenges and Limitations

As our study entails the construction of a new dataset, we came across a few issues with the logistics of data collection, processing, and implementation as described follows.

- After extensive research, we found only one publicly accessible dataset [15] of hazard signs. Our novel attempt to create a universal safety sign dataset was challenging as the nature, notation, and format of these signs differ among countries. We will be expanding our dataset with more such samples in the near future. We will also attempt to include more challenging scenarios for feature extraction, like low-light photographs, partially obscured signs and warped or distorted signs.
- We attempted to accommodate various significant and universal signs seen throughout most countries, although segregation was a time-consuming task. However, the semantics of safety signs for generalization need improvement.

VIII. Conclusion

This research aims to help and assist the elderly, visually impaired people, and people with hearing loss with their daily commute, travel, or work routine. We presented a proof-of-concept of our idea with a smartphone application.

We successfully created our dataset consisting of traffic and hazardous signs from various sources. We manually scanned each image for safety signs and assigned an appropriate threat label - danger, caution, or prohibitory. We trained and compared three models - YOLO v5, YOLO v3, and SPP. From the comparison, we found that the YOLOv5 object detection model was superior in detection speed and accuracy. We developed an Android app, "Safety Sign Detection", to test our model in real-life scenarios. The model deployed on the application classified the object it read from the video feed into predetermined threat categories. Additionally, these classification labels or the severity of the threat were communicated to the user through haptic and speech feedback using the same device. The application also handled feedback from multiple-class detection by prioritizing the most severe safety sign.

In the future, our goal is to expand our dataset with construction, industrial, and other types of safety signs, and to improve the semantics of safety signs for generalization. Creating sub-classes at each threat level for precise identification could also enhance the application.

References

[1] S. A. Cheraghi, G. Fusco, and J. M. Coughlan, "Real-time sign detection for accessible indoor navigation," in *Journal on technology and persons with disabilities:... Annual International Technology and Persons with Disabilities Conference*, vol. 9. NIH Public Access, 2021, p. 125.

[2] G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, L. Changyu, A. V, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, J. Hajek, L. Diaconu, Marc, Y. Kwon, oleg, wanghaoyang0106, Y. Defretin, A. Lohia, ml5ah, B. Milanko, B. Fineran, D. Khromov, D. Yiwei, Doug, Durgesh, and F. Ingham, "ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," Apr. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4679653

[3] C. G. Serna and Y. Ruichek, "Classification of traffic signs: The european dataset," *IEEE Access*, vol. 6, pp. 78 136–78 148, 2018.

[4] R. Hasegawa, Y. Iwamoto, and Y.-W. Chen, "Robust detection and recognition of japanese traffic sign in the complex scenes based on deep learning," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2019, pp. 575–578.

[5] X. Xu, J. Jin, S. Zhang, L. Zhang, S. Pu, and Z. Chen, "Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry," *Future Generation Computer Systems*, vol. 94, pp. 381–391, 2019.

[6] D. Chakraborty and W. Chiracharit, "Washroom sign detection using convolutional neural network in natural scene images," in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, 2020, pp. 706–709.

[7] S. Kantawong, "Hazardous signs and fire exit signs classification using appropriate shape coding algorithm and bpn," in *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*, 2012, pp. 23–27.

[8] J. Zhang, L. Hui, J. Lu, and Y. Zhu, "Attention-based neural network for traffic sign detection," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1839–1844.

[9] A. Mishra, N. Raj, S. Bodhe, and G. Bajwa, "Safety Signs Dataset," https://github.com/alankritmishra/Safetysign_detetection_with_haptic-feedback/blob/main/Safe_sign_dataset.md, (Published on 08/13/2021).

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[12] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[13] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.

[14] I. KK Technologies, "Road signs datasets," Apr 2020. [Online]. Available: https://makeml.app/datasets/road-signs

[15] M. Mohamed, J. Tünnermann, and B. Mertsching, "Hazmat label dataset - seeing signs of danger," Aug 2018. [Online]. Available: osf.io/b5dap

[16] P. Skalski, "Make sense," https://www.makesense.ai/, (Accessed on April 14, 2022).

[17] "Roboflow," https://app.roboflow.com/, (Accessed on April 14, 2022).

[18] "Pytorch," https://pytorch.org/mobile/home/, (Accessed on April 14, 2022).

[19] "Pytorch android," https://pytorch.org/mobile/android/, (Accessed on April 14, 2022).

[20] PyTorch.org, "Lite module loader," https://pytorch.org/javadoc/1.9.0/org/pytorch/LiteModuleLoader.html, (Accessed on April 14, 2022).

[21] "Pytorch demo app," https://github.com/pytorch/android-demo-app, (Accessed on April 14, 2022).

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[24] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.