Traffic sign detection and recognition using few-shot learning

Do Dang Gia Vuong, Nguyen Chinh Dat, Do Xuan Bach

A thesis submitted in part fulfillment of the degree of BSc. (Hons.) in Artificial Intelligence

with the supervision of Dr. Bui Van Hieu

Bachelor of Artificial Intelligence

Hoa Lac campus - FPT University

April 2023

ACKNOWLEDGMENTS

First of all, we would like to express our sincere appreciation to my supervisor, Dr. Bui Van Hieu, for his patience, professional attitude, and immense knowledge. He has always dedicated himself to assisting us in the most difficult situations. My completion of the thesis would not have been possible without the assistance and encouragement.

I would also like to thank all the teachers in the Information Technology department of FPT University who provided us with various knowledge and useful skills that helped us to complete our research.

Lastly, I would also like to thank my family for their support during my four years of study and for their encouragement as I worked on this graduation project.

DECLARATION

This thesis is composed of our original work and contains no material previously published or written by another person except where due reference has been made in the text. The content of our thesis is the result of work we have carried out since the commencement of the course AIP490 and does not include work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution.

We acknowledge that an electronic copy and printed copy of our thesis must be lodged with the University Library and, subject to the policy and procedures of FPT University, the thesis be made available for research and study unless a period of embargo has been approved by the Head of the Information Technology Department.

We acknowledge that the copyright of all material contained in our thesis resides with the copyright holder(s) of that material. Where appropriate, we have obtained copyright permission from the copyright holder to reproduce material in this thesis.

Abstract

With the advance of self-driving car technology, the demand for high quality computer vision systems is increasing. On roads, different signs offer important information to a human driver through their visual system. For a self-driving system information gathered from cameras can be processed with the help of deep learning and computer vision. With object detection, traffic signs information can be identified with a high accuracy rate and will help meta-learning for classifying traffic signs easily. With many methods for identifying images, our thesis coupled computer training for image recognition with machine learning techniques to help categorize in a limited collection of accessible data because many methods with high accuracy also need a large amount of data to complete the learning process. This allowed us to get findings that helped correctly forecast outcomes while reducing the amount of computing work required. We were able to complete this project despite the challenge posed by the fact that just a limited quantity of data was utilized in the process, but our Siamese network had achieved the accuracy of 0.9966 with only 26.65 minutes training runtime .

Keywords: Computer vision, Object detection, Meta-learning, Machine-learning.

Table of Contents

1. Introduction

Intelligent traffic sign detection and recognition has become a basic function for intelligent network-linked automobile systems, as well as a direction for intelligent traffic development. However, because of the complexities of driving surroundings, traffic sign detection and recognition are frequently influenced by elements such as passing vehicles, buildings, and roadside plants. Neither the driver nor the vehicle vision system can accurately understand the meaning of traffic signs and, as a result, cannot make an informed decision on the next driving maneuver. As a result, academics have proposed ways for identifying traffic signs in real time.

The capacity to determine the location of a traffic sign from a picture is the first stage in identifying a traffic sign, and the second is to identify the sign's meaning. At the moment, there are two common ways for identifying traffic signs. The traffic sign is classified based on its color and shape. For example, the literature [1] uses a GAN to produce more realistic and varied training pictures and uses Yolo V3 and V4 to identify the image or more efficient model to detect object like Yolo V5 [23]. The literature [2] uses an improved SSD algorithm named MF-SSD to classify traffic signs. Deep learning methods based on neural networks have become more effective at picture recognition challenges as neural network algorithms have improved. This research employs the Faster Region-based Convolutional Neural Network (Faster - RCNN) algorithm [3], which is built on the concepts of deep learning and transfer learning. The thesis [4] presents a new traffic sign detection (TSD) network called EmdNet and an efficient traffic sign classification (TSC) network named ENet.

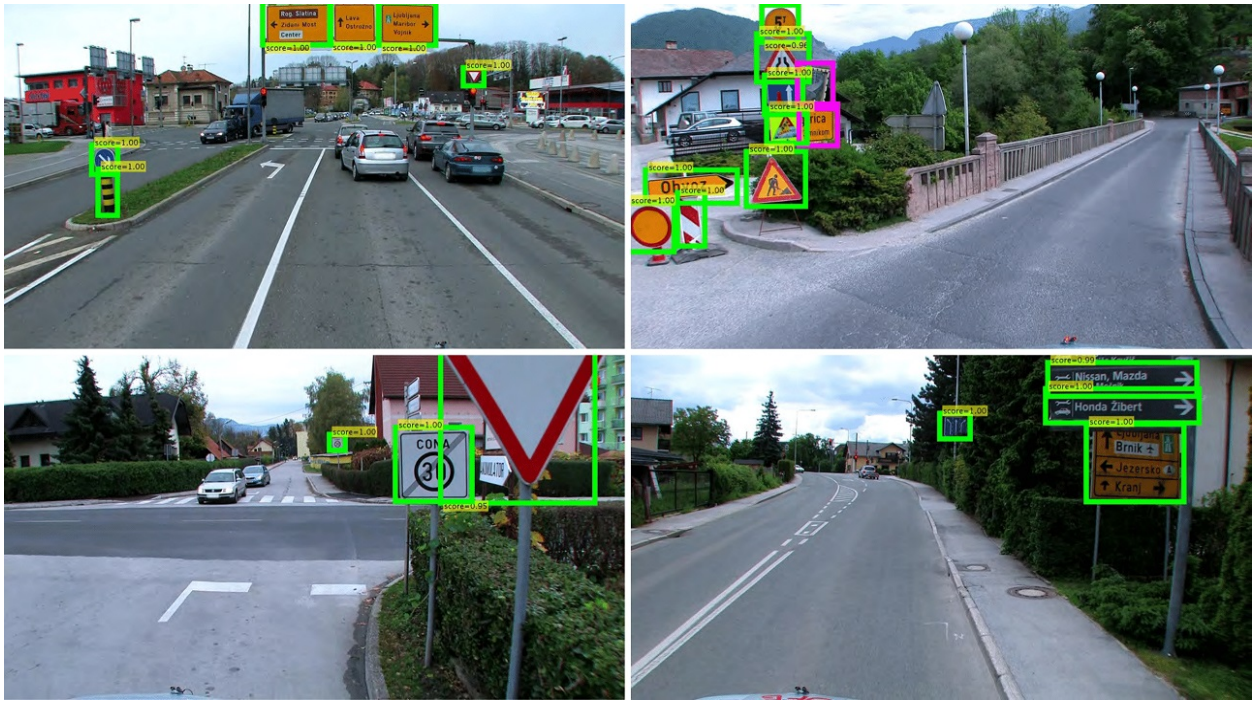

Figure 1. *Results of CNN detection in example data*

Aside from that, we also have to deal with a huge quantity of data, which affects the time it takes for machine learning to get longer. Additionally, there are occasions when the amount of data is so scarce that machine learning does not accomplish the outcomes that were anticipated. Since that time, an optimal algorithm has been required in order to construct a predictive model that is appropriate for the circumstance - few-shot learning. This makes it simple for drivers to recognize signals thanks to an intelligent system that has full access to all of the information on the journey and contributes to the driver's ability to make safer ride decisions. Some of our discoveries that contribute to the solution of the issue stated include a straightforward and efficient method for learning many times based on computer prototypes for each class. This method includes the use of a Siamese Network [15] for rapid machine learning, which enables the process to

become both inexpensive and quick. Or some methods based on comparing the similarity of features [20] and contrastive learning [21] for machine learning that help to save time for classifying.

Figure 2. *Results of classification network*

In this thesis, we also focus on gaining the best accuracy of traffic sign detection and limit the resources for classifying traffic signs by combining two

methods to improve the recognition model about speed and capacity of data. In our model, we use YOLOv5 to detect the traffic sign in the images to achieve the best accuracy and after that, we classify by methods of Few-Shot Learning to reduce time and resource usage for this project.

2 Related works

2.1 Traffic signs detection

To accomplish the job of traffic signs identification, researchers have recently largely chosen a visual scheme based on deep CNNs (convolutional neural networks). Such one-stage detectors as YOLOv1-v4 [5]–[8], SSD [9], and RetinaNet [10] are based on the regression technique, which immediately outputs the position and category of the bounding boxes densely in a single-shot. R-CNN [11], Fast R-CNN [12], Faster R-CNN [13], and Mask R-CNN [14] are all examples of region proposals that form the basis of a two-stage detector.

The actual arrangement of photos is supplemented with more realistic and diverse training images generated using Generative Adversarial Networks (GAN) [1]. The study blends synthetic photos with actual photographs to improve datasets and validate the efficacy of synthetic datasets, and it shows how the quality of synthetic pictures made by DCGAN, LSGAN, and WGAN is assessed. The author's training methodology involves varying the quantity and size of pictures. Similar to how they used the Mean Square Error (MSE) to evaluate accuracy, they also used the Structural Similarity Index (SSIM) to do the same for images. The report also provides a numerical representation of the SSIM gap between the simulated and real-world examples. When more training photos are employed, the

synthetic image closely resembles the real thing. A total of 200 photos with a resolution of 32x32 yields the greatest SSIM result. The author then compares the original image model to the synthesis image model after augmenting the original picture dataset with synthetic images. The most recent versions of Yolo (Yolo V3 and Yolo V4) are used in this thesis's experiment. LSGAN's synthetic picture is combined with the original image to boost identification performance; this approach yields an accuracy of 89.33% on Yolo V4 and 84.91% on Yolo V3, respectively.

The authors begin by presenting YOLOv5 [22], a cutting-edge object detection system that employs a deep neural network to recognize and categorize things in a picture. They highlight the enhanced precision, speed, and adaptability of YOLOv5 in comparison to earlier versions of the algorithm. After introducing the datasets they used (the PASCAL VOC 2007 dataset, the COCO dataset, and a custom dataset comprising pictures of vehicles), the authors outline their studies. They evaluate the models' efficacy by contrasting their speed and accuracy. While the bigger YOLOv5 models (Yolov5l and Yolov5x) are slower and need longer training durations, their findings suggest that they outperform the smaller models (Yolov5s and Yolov5m) on the more complicated datasets (COCO and the custom vehicle dataset). On the simpler PASCAL VOC 2007 dataset, the authors write, there is little to no difference in accuracy across the models. The authors conclude that the accuracy and processing speed requirements of a given application should be taken into account when deciding which YOLOv5 model to use. They also point out the necessity for more study into the best parameters for training and deploying YOLOv5 models.

In order to get the multiscale features in pyramids, the author suggests using a cascaded R-CNN [3]. When training a cascaded network, all layers save the first combine their output bounding boxes into a single one. The identification of traffic signs may benefit from this approach. The author next suggests a multiscale attention technique, whereby the weighted multiscale features are obtained using dot-product and softmax, and then fine-tuned by adding the features together to emphasize the traffic sign characteristics and enhance the accuracy of the traffic sign recognition. In order to reduce the negative effects of complicated settings and comparable misleading traffic signals, the research concludes by increasing the quantity of challenging negative examples for dataset balance and data augmentation in the training phase. The data enhance technique uses a model of complicated environmental changes to enlarge the German traffic sign training dataset. This approach achieves an accuracy of 98.7 percent and a recall of 90.5 percent on the GTSDB, 99.5 percent and 83.62 percent on the CCTSDB, and 98.1 percent and 85.6 percent on the Lisa dataset.

In order to recognize traffic signals in real time across several scales, this study suggests a revised version of the YOLOv5 algorithm [23]. The suggested method is an attempt to solve the problem of recognizing traffic signs from a variety of angles and environments. The difficulties in doing so and the limits of conventional object identification algorithms are first discussed by the author. Next, they suggest a refined version of the YOLOv5 method that uses a multi-scale detection technique to further enhance traffic sign identification precision. The author uses a collection of traffic sign photos of varied sizes and lighting conditions to assess the efficacy of the updated YOLOv5 algorithm. They evaluate the enhanced YOLOv5 algorithm against the baseline YOLOv5 and other cutting-edge object identification methods. According to the author, the enhanced

YOLOv5 algorithm exhibited state-of-the-art performance in terms of accuracy and speed while detecting traffic signs. They also highlight that the enhanced system could recognize traffic signs despite differences in scale and brightness, both of which pose difficulties for standard object detection software.

2.2 Few-shot object classification

Overfitting the network and failing to fulfill the potential of deep networks are common results of using a training set with too few samples, since the performance of big neural networks is limited by the size of the training set. When training and making predictions using deep learning, few-shot learning is a technique that may be used when there is not enough data. Models trained with insufficient data are prone to overfitting the training data and underfitting the job in few-shot tasks.

The unique use of meta-learning may increase the precision of target recognition for the categorization of new categories, and it has shown tremendous promise in handling few-shot challenges [15, 16]. To swiftly master new tasks, the primary concept is to employ meta-information amassed from past activities as previous knowledge, and then learn a limited set of target samples. This approach is very flexible, so it can handle anything. A fine-tuning-based multi-scale few-shot detection model was suggested by Zhao et al. [17], which makes use of residual involution blocks to build the all feature learning architecture and designs PAM to aggregate from all feature levels. The first step of this process makes use of the semantic information contained in shallow features to pinpoint the position of objects, while the second is partially calibrated using a small balanced dataset. In order to automatically obtain the significance of each feature channel, Whang et al. [18] proposed a general object detection system that integrates the feature-based

domain attention mechanism with sequence and exception networks, and assigns network activation to different domains via SE adapter library learning. To improve performance, SENet's central principle is to learn the feature weight in relation to the loss, making the weight of an effective feature map big and the weight of an ineffective feature map small. The overall detection algorithm, however, fails to account for the issue of spatial dislocation, which is shown in the subpar results achieved while trying to see traffic signs despite their tiny targets and cluttered backgrounds. Candidate suggestions for new classes and missing high IOU boxes were made possible because of the RPN stage improvements made by Han et al. [19]. To increase the recall of the few novel class candidate boxes, a coarse-grained prototype matching network (meta-RPN) was proposed. This network uses a non-linear classifier based on metric learning in place of the conventional linear target classifier, and it addresses the issue of the similarity between anchor boxes and novel classes in query images. As a prototype, a matching network (meta-classifier) with a high level of granularity was developed. To improve detection accuracy, the network employs modules to align spatial features and pay attention to the foreground, which helps to cope with the overlap between noise and new classes. However, the issue of prototype deviation exists inside the meta optimizer itself. This issue arises because gradient estimation relies on averages, which might be inaccurate for small sample sizes with incomplete labels.

In this portion of the research, we concentrate on the process of finding and categorizing models with a minimal quantity of data. The reference results from the other studies that we have gathered serve as the basis for this section. We favor the most recent models because they provide data processing results that are either more accurate or the same as those produced by other models. However, the

number of resources that need to be invested in research is not excessive in order to shorten the amount of time required for image processing, and also, it does not have an excessive impact on the correctness of the results.

3 Data preparation

3.1 Data description

Our research employs photographs of streets in Vietnam as the primary data source. The data set encompasses a diverse range of images collected from various sources, including research projects and mobile applications that pertain to traffic signs in Vietnam. The dataset used in this study contains 1,170 images, each with a resolution of 1622x626 pixels. This dataset was selected as it provides a diverse and challenging set of images to evaluate the performance of the proposed model. The high resolution of the images allows for the capture of fine details and subtle variations in the visual content, which is essential for achieving accurate and reliable predictions. Additionally, the large size of the dataset ensures that the model is trained on a sufficient amount of data to avoid overfitting and to generalize well to new, unseen images. The use of such a dataset is expected to enable the model to learn complex visual patterns and to achieve high levels of performance on a range of tasks.

Figure 3. *Examples of data in dataset*

The images in the dataset contain traffic signs of varying sizes, types, and locations, captured at different times and weather conditions. The dataset includes images captured from various perspectives of roadways, encompassing city streets, urban areas, freeways, crossroads, roundabouts, and tunnels. The images in our dataset were captured from multiple street locations, each offering a diverse range of viewing angles and taken mostly from the vantage point of drivers who are operating their vehicles.

To conduct our study, we had manually relabeled the dataset. There are 29 types of signs categorized by the meanings and shapes of the signs which are divided into 29 classes including the additional classes for signs that aren't identified. They are:

Class	**Class name**
0	*one way prohibition*

1	*no parking*
2	*no stopping and parking*
3	*no turn left*
4	*no turn right*
5	*no u turn*
6	*no u and left turn*
7	*no u and right turn*
8	*no motorbike entry/turning*
9	*no car entry/turning*
10	*no truck entry/turning*
11	*other prohibition*
12	*indication*
13	*direction*
14	*speed limit*
15	*weight limit*
16	*height limit*
17	*pedestrian crossing*
18	*intersection danger*
19	*road danger*
20	*pedestrian danger*
21	*construction danger*
22	*slow warning*
23	*other warning*

24	*vehicle permission lane*
25	*vehicle and speed permission lane*
26	*overpass route*
27	*no more prohibition*
28	*other*

The regulatory signs provide drivers with positive instructions, such as indication, no more prohibition, and pedestrian crossing, as well as negative instructions, such as *speed limit*, *no stopping and parking*, *one way prohibition*, etc . Dataset also contains warning signs that alert drivers of potential hazards ahead, such as curves, intersections, pedestrians, construction and so on. The informative signs also included, which provide useful information to drivers, such as *directions*, *vehicle permission lane*, etc. There are also 3 abnormal classes, *another prohibition* to indicate restriction signs that are undefined, *other warning* for unidentified warning signs and *others* are signs that do not belong in any other 28 classes.

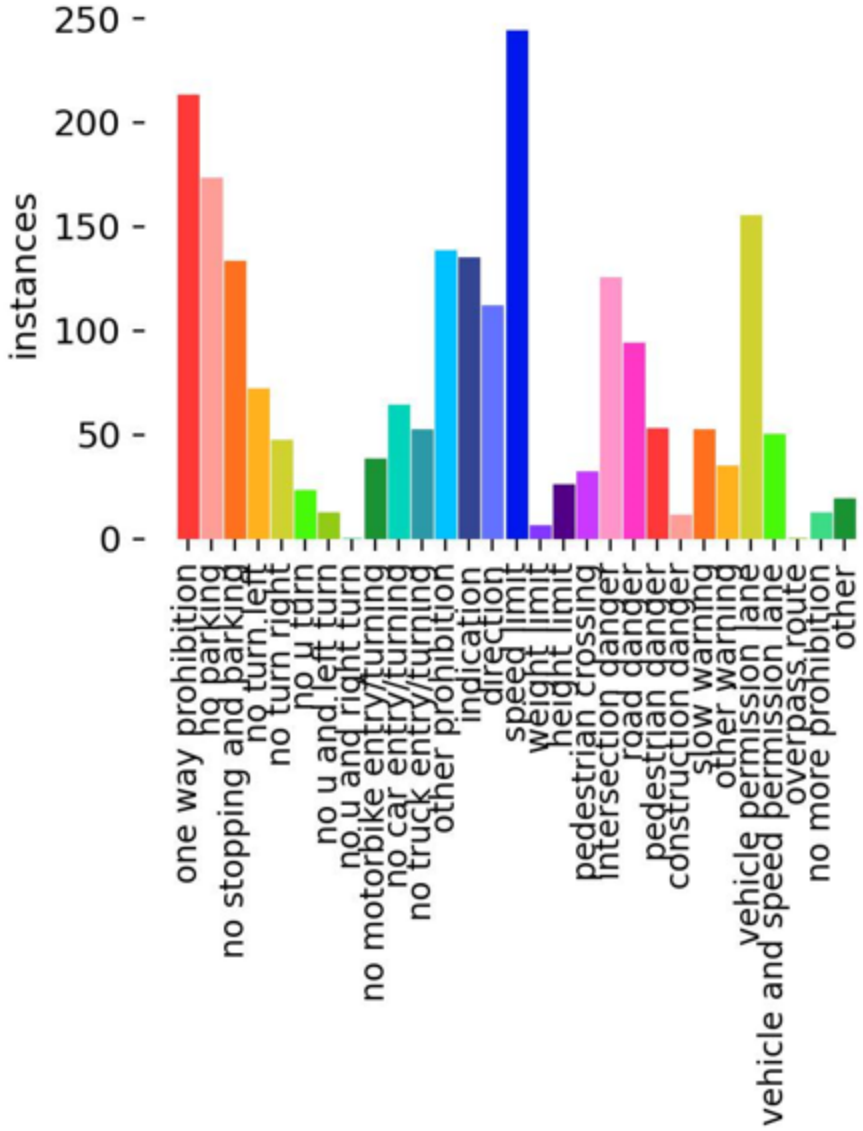

Figure 4. *Data labels*

Figure 4 shows that the distribution of 29 types of traffic signs in the dataset is imbalanced, ranging from 0 to 250. Traffic signs mostly come from *one way prohibition*, *speed limit*, *no parking* and *vehicle permission lane*, with the most

appearances being *speed limit* signs. Classes with the least amount of data are *weight limit*, *no more prohibition*, *no u turn*, *no u and left turn*, *construction danger* and *other*. Classes with no image which are *no u and right turn* and *overpass route*.

Data is taken from Viet Nam traffic sign(YOLO format) on Kaggle with 847 images and 323 additional custom images from zalo_traffic_sign. The dataset is suitable for traffic sign classification tasks, which can be useful for building smart cars that can automatically recognize traffic signs along the road.

3.2 Data split

Data set is splitted into training, validation and test set. There are 900 images in the training set, 180 images in the validation set and 90 remaining images in the test set. The training set is used to train a YOLO model that can detect and classify the traffic signs into one of the 29 classes of traffic signs. The validation set is used to tune the hyperparameters of the model and evaluate its performance during training. The test set is used to measure the final accuracy of the model on unseen data.

3.2 Data format

Traffic signs are classified by the type of each sign detected. In an image, each traffic sign is indicated by a bounding box. A bounding box consists of values bx, by ,bh, bw given that bx, by are the coordinates of the center of the box and bh, bw are the height and the width of the box. A class is labeled based on the type of the sign. The dataset includes image files and the corresponding text file with bounding boxes' coordinates and the class indication number. Model will be trained using the YOLO method, which provides object detection.

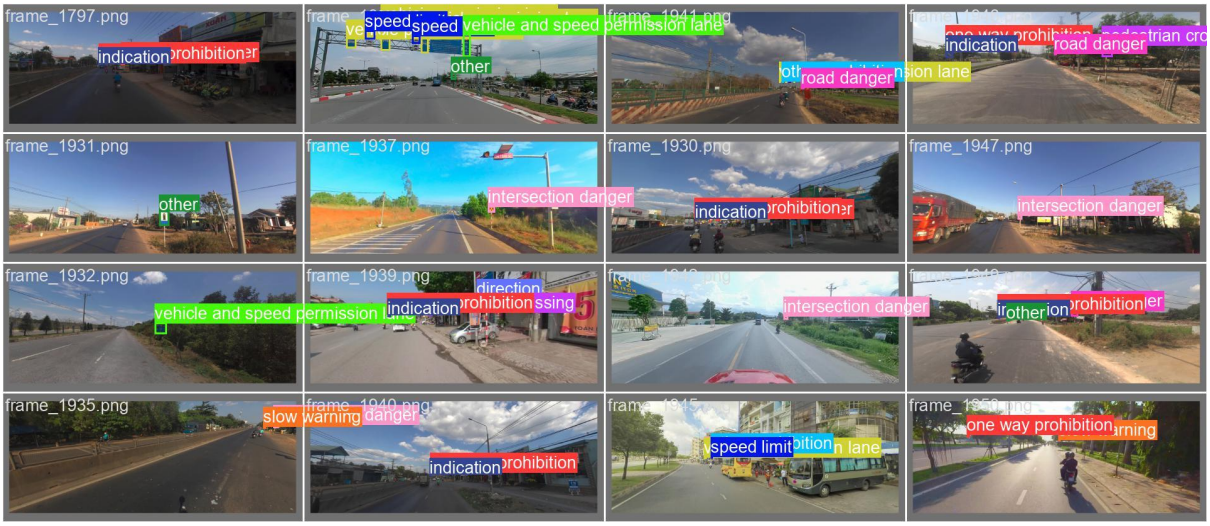

Figure 5. *Data format*

Figure 5 illustrates a batch of training data images. Each image is labeled according to YOLO format with traffic signs indicated by a bounding box and the class of the traffic sign is shown above as a part of the bounding box.

4 Methodology

In order to accomplish what was set out to do, we relied on two different models: YOLOv5 and Few-Shot Learning. We want to get a result with a high level of accuracy by using YOLOv5 for the purpose of traffic signs detection; this will allow the subsequent processes to proceed without any hitches. After that, we decided to employ Few-shot Learning to categorize the traffic signs that had been detected by YOLOv5. This decision was made to assist in the reduction of the resources that were required for computer training and to help accomplish the desired end result as intended.

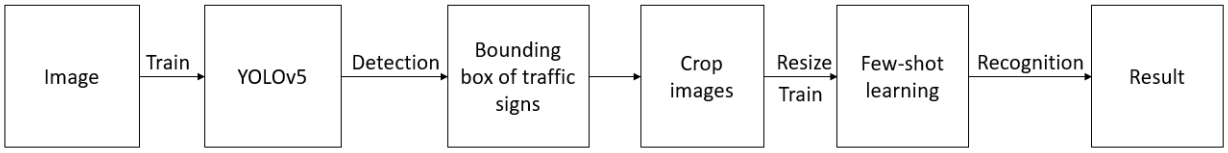

Figure 6. *Method illustration*

4.1 Object Detection

The process of object recognition through images involves a crucial stage known as object detection. Thus, our research endeavors are directed toward exploring an object detection model that is easily accessible while maintaining the requisite level of accuracy in the research methodology. The YOLOv5 model was selected for this study due to its utilization of the standard YOLO structure, which offers optimization advantages over its predecessors, YOLOv3 and YOLOv4.

"You Only Look Once", or YOLO. This model can recognize items in each cell of the feature map across all photos.

YOLOv5 is a real-time object identification system that makes use of deep learning in order to recognize items included within still photos and moving video. The You Only Look Once (YOLO) family of object detection models has been updated to its fifth iteration with this version. YOLOv5 was developed to be quick, accurate, and effective, and it has the capacity to identify things in real time.

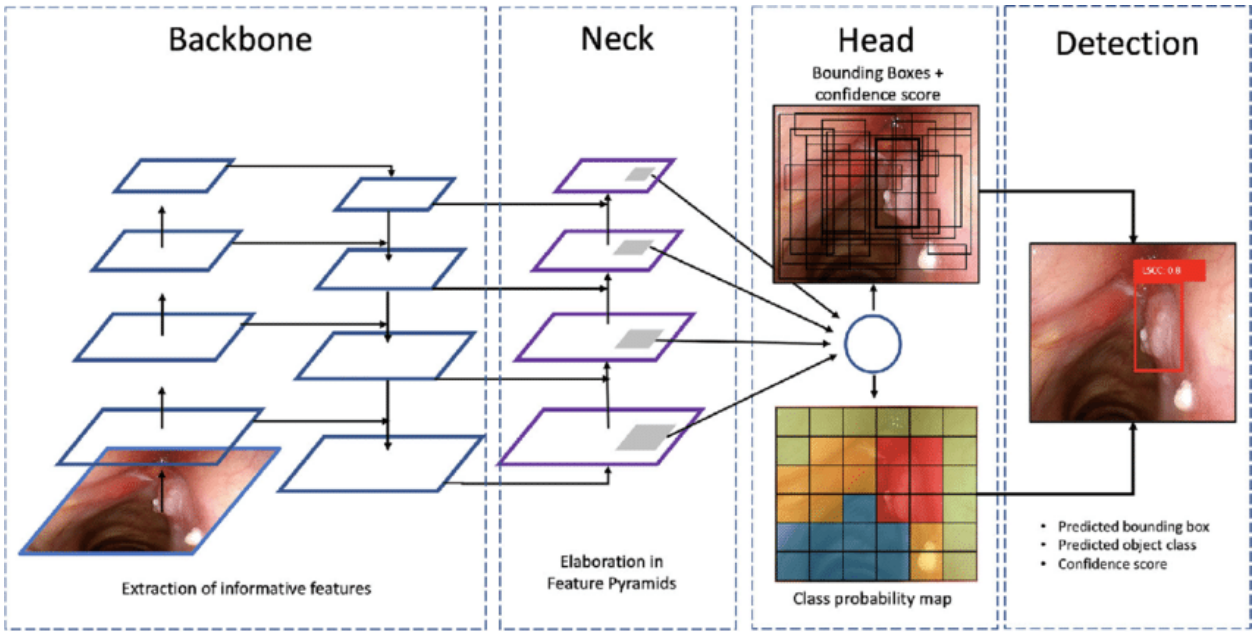

Figure 7. *An example of YOLOv5 model*

YOLOv5 is constructed using the standard architecture of the YOLO model, which consists of a Convolutional Neural Network (CNN) for the extraction of features and an output element for the detection of objects. In addition, YOLOv5 has several enhancements, such as an updated neural network structure featuring expanded convolutional layers for enhanced feature extraction, data augmentation techniques to mitigate overfitting and enhance precision, transfer learning methodologies for training the model on smaller datasets while maintaining high performance, and automated hyperparameter tuning to optimize the model's accuracy.

Overall, The architecture of YOLOv5 comprises three primary constituents, namely the backbone network, the neck network, and the head network.

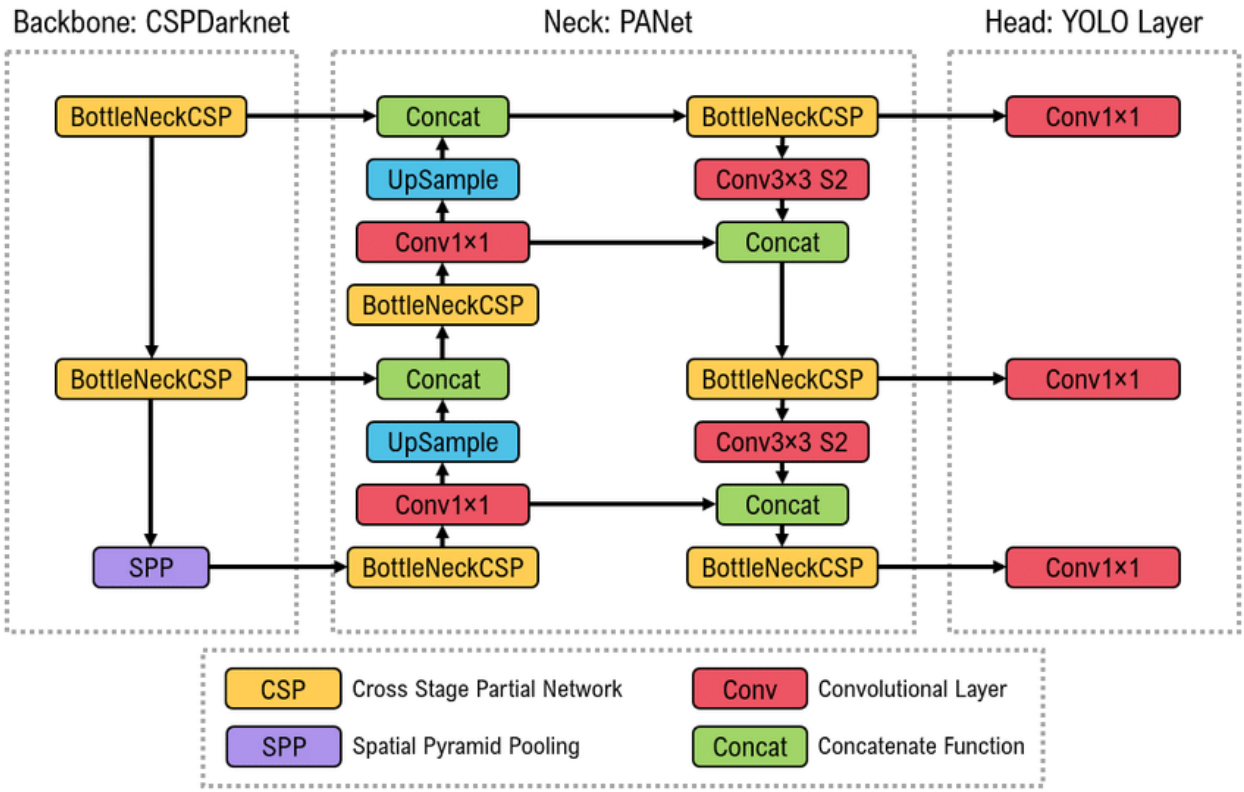

Figure 8. *YOLOv5 Architecture*

4.1.1 Bounding Box

Our approach, in accordance with YOLO9000, forecasts the bounding boxes by using the dimension clusters as anchor boxes [6]. The network makes four predictions on the coordinates of each bounding box, which are as follows: t_x, t_y, t_w and t_h. If the cell is moved up and to the left by a distance of (c_x, c_y), and the bounding box previous has dimensions of (p_w, p_h), then the following predictions are accurate:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

In order to accurately capture the locations of items inside the input picture, YOLOv5 makes use of bounding boxes. A confidence score is assigned to each bounding box, and this value reflects how certain the model is that the item in question is located inside the bounding box in question. The confidence score is determined by the intersection-over-union (IoU) value that is calculated using the ground-truth bounding box in comparison to the anticipated bounding box.

During training, the YOLOv5 model acquires the ability to provide bounding box predictions that are very similar to the actual bounding box values. This is accomplished by minimizing a loss function that punishes the model for making false predictions about the bounding boxes.

The YOLOv5 model makes use of the anticipated bounding boxes in order to ascertain the position of the items in the picture as well as their dimensions throughout the inference process. Additionally, the model makes predictions on the class labels of all of the objects included inside the bounding box.

Through the use of logistic regression, YOLOv5 is able to forecast an objectness score for each bounding box. If the bounding box prior overlaps a ground truth object by a greater amount than any other bounding box prior, then this should be set to 1. If a previous bounding box is not allocated to a ground truth object, the object suffers no loss in its ability to forecast coordinates or classes; the only loss it experiences is in its objectness.

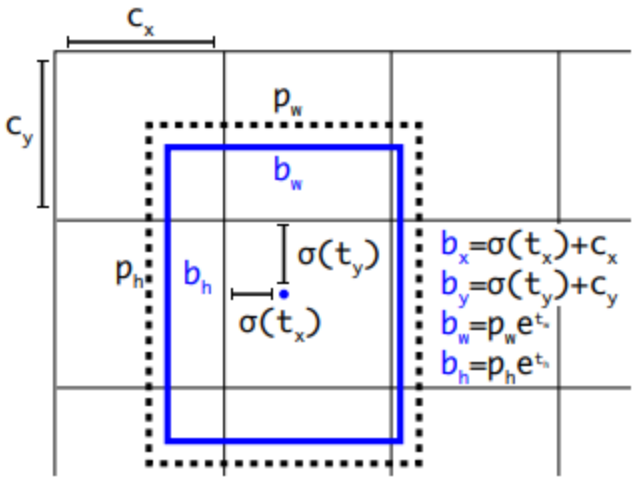

Figure 9. *Bounding box prediction*

In general, bounding boxes are an essential part of the YOLOv5 object detection system. They play a significant part in precisely localizing and categorizing objects in photos, and they are one of the core components of the system.

4.1.2 Backbone Network

The backbone network is the primary feature extraction network in YOLOv5, and it is responsible for processing the input picture and extracting high-level features for the purpose of object identification. The Cross-Stage Partial (CSP) design, which is a variant of the ResNet architecture, is what is used to build the backbone network.

The architecture of CSP is intended to minimize computational burden and expedite the processing of input images. The technique involves partitioning the network into smaller stages that operate concurrently to handle features, thereby diminishing the total computational burden. The architectural design incorporates

abbreviated connections that facilitate the direct flow of gradients across stages, thereby enhancing gradient flow and expediting the training procedure.

In YOLOv5, the backbone network is divided into many stages, with each stage including a number of residual blocks. Complex features may be extracted with the aid of the several convolutional layers that make up each residual block. The model's ability to recognize objects of varying sizes is a result of the backbone network's architecture, which extracts characteristics at several scales.

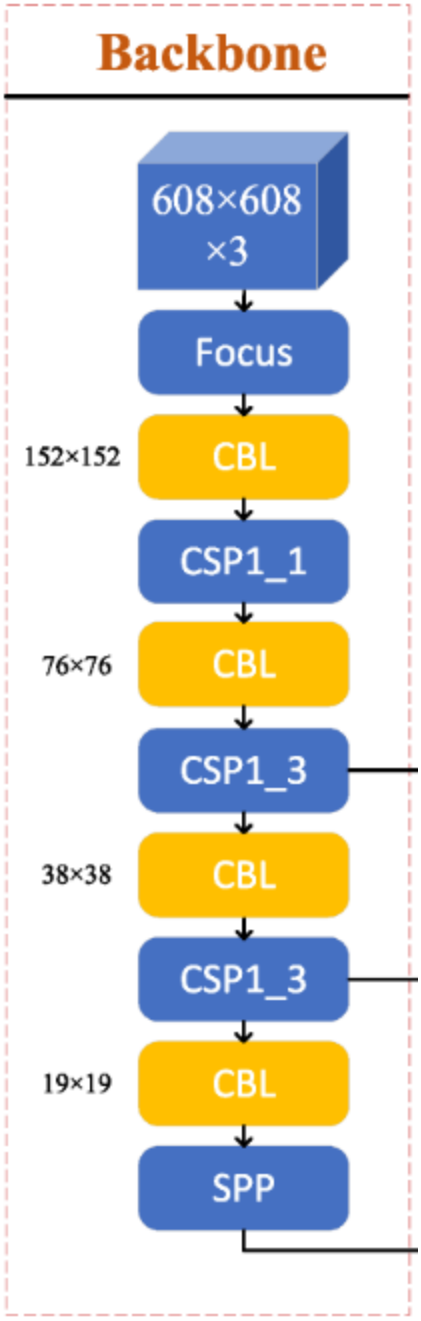

Figure 10. *Backbone Network using CSPDarknet53*

First, the input picture is processed by a set of convolutional layers, and then the spatial size of the feature maps is shrunk by a max-pooling layer. After that, the feature maps go through many levels of residual blocks, each of which does feature

extraction at a different granularity. The residual blocks employed in every stage of the architecture adopt the CSP design, featuring shortcut connections that facilitate the seamless propagation of gradients across stages. The enhancement of gradient flow is observed, resulting in a decreased probability of vanishing gradients and an expedited training process.

Following the stage of feature extraction, the output is subsequently transmitted to the detection head, whereby it anticipates the positioning and classification of entities within the inputted image. The process of identifying objects involves the utilization of convolutional layers and anchor boxes by the detection head to estimate the class probabilities and bounding boxes.

4.1.3 Neck Network

The neck network is an optional module that may be put between the backbone network and the detecting head in YOLOv5. It is located in the middle of the two networks. Its principal function is to combine the feature maps that have been generated at various levels of the backbone network and to carry out further feature extraction so that object detection may be made with greater precision. The SPP (Spatial Pyramid Pooling) module is the foundation of the neck network in YOLOv5. This module does spatial pooling at several scales to collect context as well as spatial information at a variety of different levels. The SPP module is used in order to extract features from various scales included within the feature maps that are created by the backbone network.

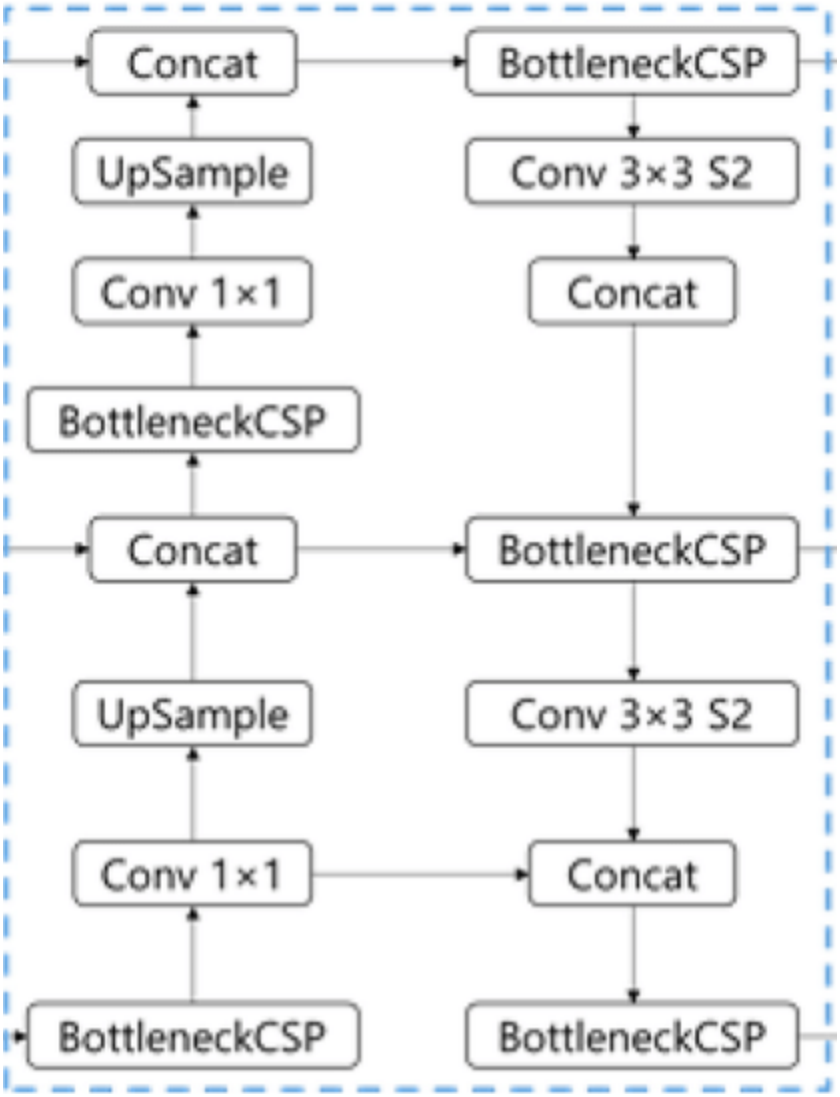

Figure 11. *Neck Network architecture*

In YOLOv5, the neck network is made up of many convolutional layers, max-pooling layers, and SPP modules respectively. The output of the backbone network is sent into the neck network so that more complicated characteristics may be extracted at a variety of scales. Concatenation is then used by the neck network to assist in combining the characteristics from the various scales, which contributes to the network's ability to record both local and global information. It is possible to

adapt the YOLOv5 neck network to a variety of input sizes and detection tasks since it was developed with flexibility in mind from the start. It also has the capability of being altered to manage the amount of feature maps produced by the backbone network as well as the number of SPP modules that are used inside the network.

4.1.4 Head Network

The concluding component of the YOLOv5 model is the head network, which is accountable for the detection and classification of objects within the input image. The process involves utilizing the feature maps produced by the backbone network and the neck network (if applicable), as input to generate a collection of bounding boxes and corresponding probabilities for the classification of each object present within the image.

The feature maps that are produced by the backbone network and the neck network are processed by the head network in YOLOv5, which is made up of a succession of convolutional layers that are employed to do so. The convolutional layers are followed by a collection of anchor boxes, which are used to forecast the bounding boxes of objects in the input picture. This prediction is done after the convolutional layers have been completed. The anchor boxes are made to accommodate a wide range of items, both in terms of their dimensions and their aspect ratios. The YOLOv5 detection head is a modification of the YOLOv3 detection head that is used by the head network that is part of YOLOv5. The predictions that are generated by the YOLOv5 detection head are the result of a mixture of several modules, including convolutional layers, SPP (Spatial Pyramid Pooling), and PAN (Path Aggregation Network). Both the SPP module, which is used to gather context information at multiple scales, and the PAN module, which

is used to synthesize information from different scales in order to make accurate predictions, are both components of the system.

The detection head of YOLOv5 employs dynamic anchor assignment, a method that adapts the dimensions and aspect ratios of anchor boxes to the object distribution within the input image. By ensuring that the anchor boxes are suitable for the objects present in the image, this technique aids in enhancing the precision of object detection.

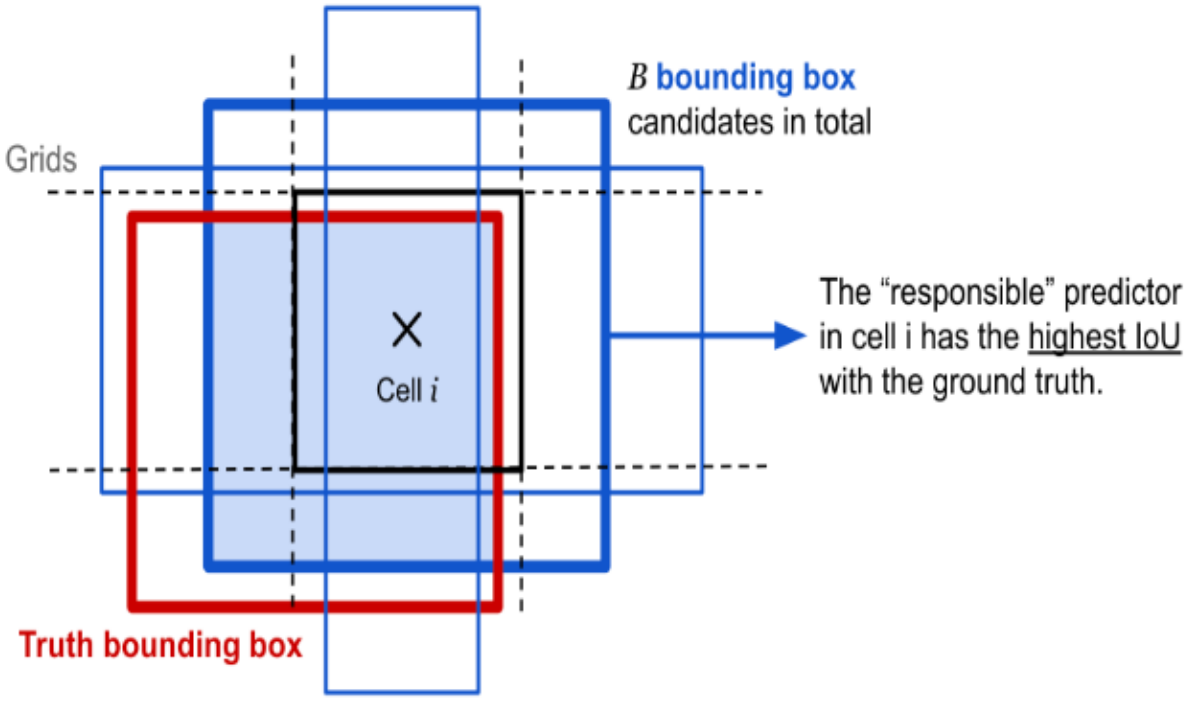

Figure 12. *Anchor box with highest accuracy will be chosen to bounding box*

4.2 Loss Function

The Loss Function of the YOLO model:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(x_i - \hat{x}_i \right)^2 + \left(y_i - \hat{y}_i \right)^2 \right]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} 1_{i}^{obj} \sum_{c \in classes} \left[\left(p_i(c) - \hat{p}_i(c) \right) \right]^2$$

1_i^{obj} : denotes if object appears in cell i

1_{ij}^{obj} : implies that the prediction was made by the *j*th bounding box predictor in cell *i*.

$$Loss_{yolo} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \longrightarrow \text{Bounding Box coordinate}$$

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \longrightarrow \text{Confidence}$$

$$\sum_{i=0}^{S^2} 1_{i}^{noobj} \sum_{c \epsilon classes} (p_i(c) - \hat{p}_i(c))^2 \longrightarrow \text{Classification}$$

Figure 13. *Steps of Loss Function algorithm formula*

4.3 Object Recognition

Insufficient amount of training data usually led deep learning models to over-fitting on the training set. Because our training set, or in few-shot learning terms, support set, is very small, there is a risk of overfitting the model to the limited training data. . We have built a CNN to evaluate the performance of a normal CNN model on a small dataset.

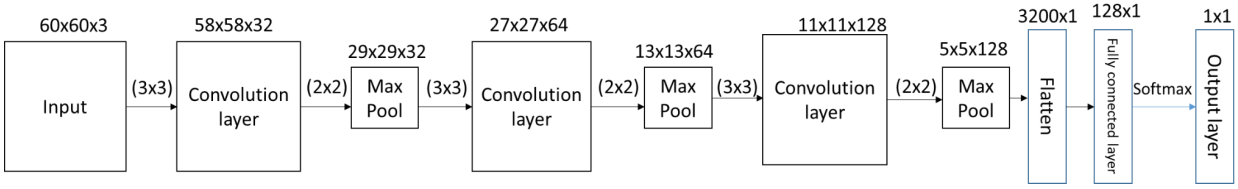

Figure 14. *Test CNN architecture*

Our CNN has three convolutional layers followed by max pooling layers. The input shape is (60,60,3), which means that the network expects color images with height and width of 60 pixels and 3 color channels (RGB). The first convolutional layer has 32 filters of size (3,3) with ReLU activation function, followed by a max pooling layer of size (2,2). The second convolutional layer has 64 filters of size (3,3) with ReLU activation function, followed by another max pooling layer of size (2,2). The third convolutional layer has 128 filters of size (3,3) with ReLU activation function, followed by a final max pooling layer of size (2,2). After the convolutional layers, the output is flattened and passed through two fully connected layers. The first dense layer has 128 units with ReLU activation function and the second dense layer has 24 units with softmax activation function, which produces the probability distribution over the 24 output classes. The network is trained to minimize the cross-entropy loss between the predicted probability distribution and the true labels.

We will see if our normal CNN model will over-fit our support set and compare the training results and performance with few-shot learning methods.

4.3.1 Few-shot Learning

Few-short learning (FSL) is a machine learning method where a pre-trained model can learn new data using only a few labeled samples per class. For example, in the medical field we often have difficulty in diagnosing bone diseases through X-rays. With some rare diseases, there might be insufficient images to use for training models. And this is where FSL comes in handy. FSL is an interesting sub-area of machine learning because the pre-trained model is trying to learn how to classify new objects given a small set of previous training data, which avoids computational power waste due to retraining the model on the new dataset.

Researchers have classified FSL into four categories:

- Zero-Shot Learning (ZSL): The goal of ZSL is to detect classes that have not been trained.
- One-Shot Learning (OSL): With OSL we only have a single sample of each class during training
- Few-Shot Learning (OSL): FSL have only a few samples per class during training
- N-Shot Learning (NSL): When we talk about FSL, we usually think of N-way-K-Shot. N is the number of classes, K is the number of samples from each class used for training. N-Shot Learning (NSL) is seen as a generalized concept. Few-Shot, One-Shot, Zero-Shot Learning are special cases of NSL.

To better understand N-way K-Shot classification, assume we have:

1. *Support set*: consists of a few labeled samples per novel category of data, which the pre-trained model should learn to generalize them.

a. N number of classes
b. K number of samples from each class (less than 10 per class)

2. *Query set:* contains Q samples from the new and old categories of data, the model needs to generalize using previous knowledge from old categories and novel information from the *support set*

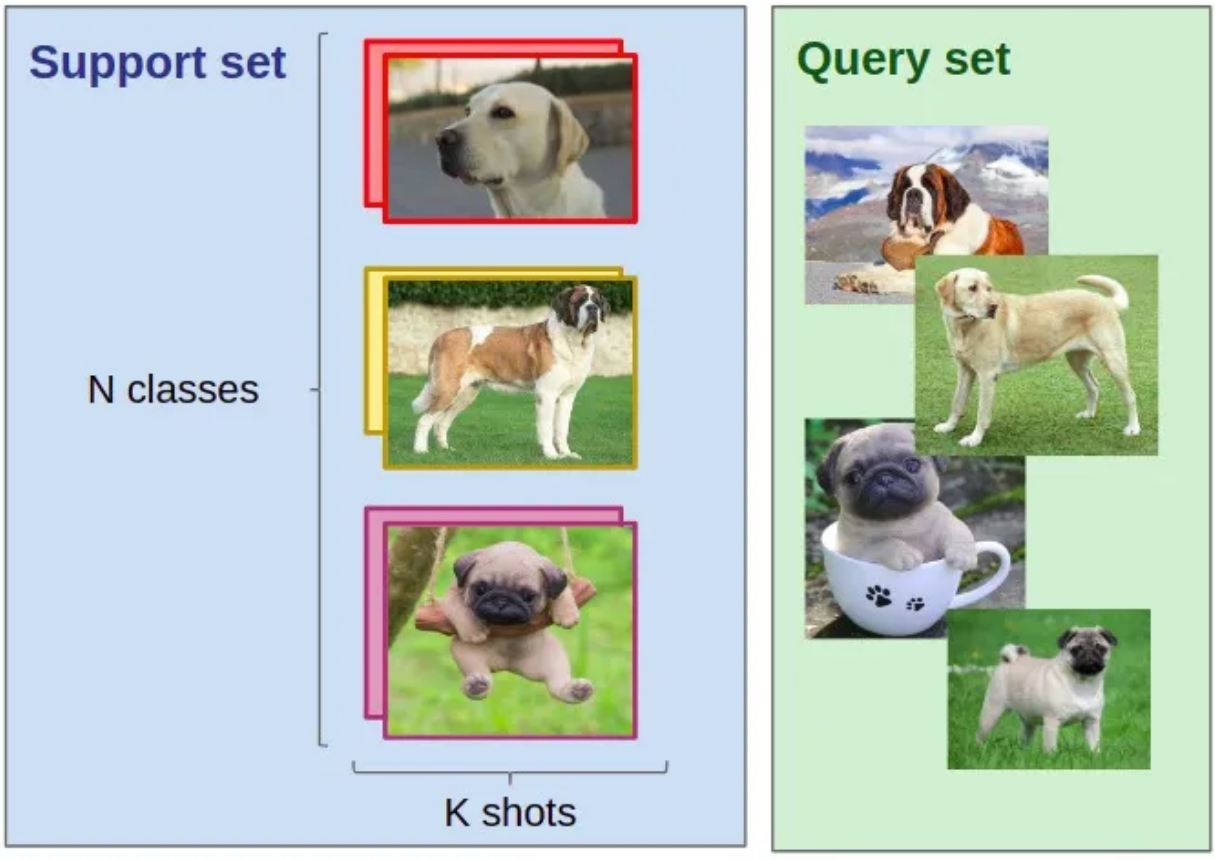

Figure 15. *Samples of sets in few-shot learning*

The objective is to categorize Q images into N classes, however, due to the limited amount of data (N*K) available for the training set, the challenge is the insufficient data to train the model.

The first step in FSL is to gain experience from similar problem-solving approaches. This is why FSL is described as Meta-Learning. In a traditional problem, we try to learn classification from training dataset and evaluate using testing dataset. In Meta-Learning, we "learn how to learn" from a training dataset, this dataset is a collection of classes that are not the classes we are dealing with.

4.3.2 Few-shot Learning approaches

4.3.2.1 Data-level

This method is based on a straightforward concept. Given the scarcity of labeled data, we may supplement our models in a number of ways to prevent over- or under-fitting. Augmentation and picture production are two common approaches. Despite the fact that augmentation does not generate any brand-new data, it does allow FSL to make better use of existing unstructured data by extracting more insights from it. When compared to other data generation techniques like GAN models, FSL requires a pre-trained generative model in order to produce novel data samples.

4.3.2.2 Parameter-level

Overfitting is a common problem in FSL due to the shortage of labeled samples. As a result, regulating inputs is an option. Parameter space restriction, regularization methods, and loss function tuning are all options. As part of its training, the model is instructed to discover the optimum path across the parameter space, yielding the most accurate predictions possible. Specifically, the meta-learning algorithm is used in this method.

4.3.2.3 Metric-level

In Metric-level, we want to determine a distance function between data points. First, features are extracted from images by a convolutional neural network and then embedded in a domain space. The distance between images can be computed now by a distance function, which can be Euclidean distance, Earth Mover Distance, Cosine Similarity-based distance, etc. Those distances show the similarity between the support and query sets.

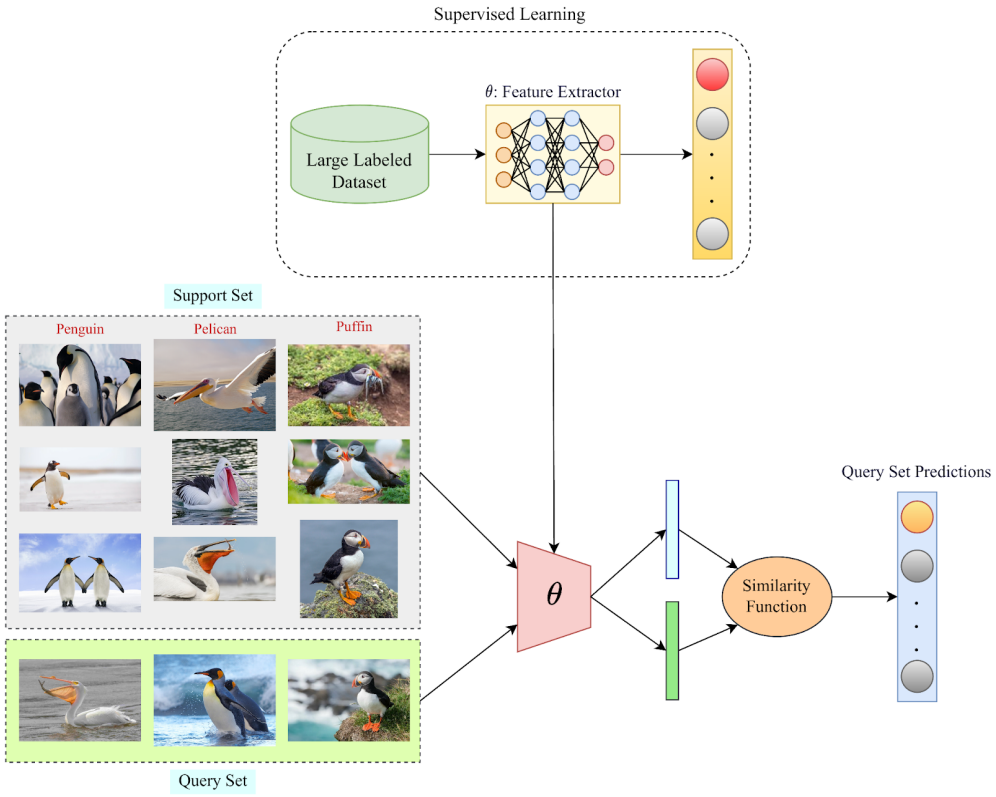

Figure 16. *Metric-Learning*

The primary idea in Metric-Learning is to find a similarity function that can map the similarities between the classes in the support and query sets. We use a huge labeled dataset to train the parameters of that similarity function. When we

have a trained similarity function, it could be used in FSL to determine similarity probabilities on the query set by using the support set information.

4.3.2.4 Gradient-based Meta-learning

In this approach, we need to build a meta-learner and a base-learner, in which, the meta-learner is a model that learns across episodes, whereas a base-learner is a model that is initialized and trained inside each episode by the meta-learner.

The execution steps of an episode of Meta-training with some tasks defined by support-set N*K and query set Q:

- Choose a meta-learning model
- Episode is started
- Initialize base-learner from meta-learner
- Training base-learner on support-set (algorithm used to train base-learner, determined by meta-learner),
- Base-learner predict on query set
- Meta-learner parameters are trained on the results of base-learner prediction

Using the information from the support set, the base-learner is trained to make predictions on the query set samples. Then the loss derived from the base-learner is then used to train the meta-learner, which makes it proficient in the classification task.

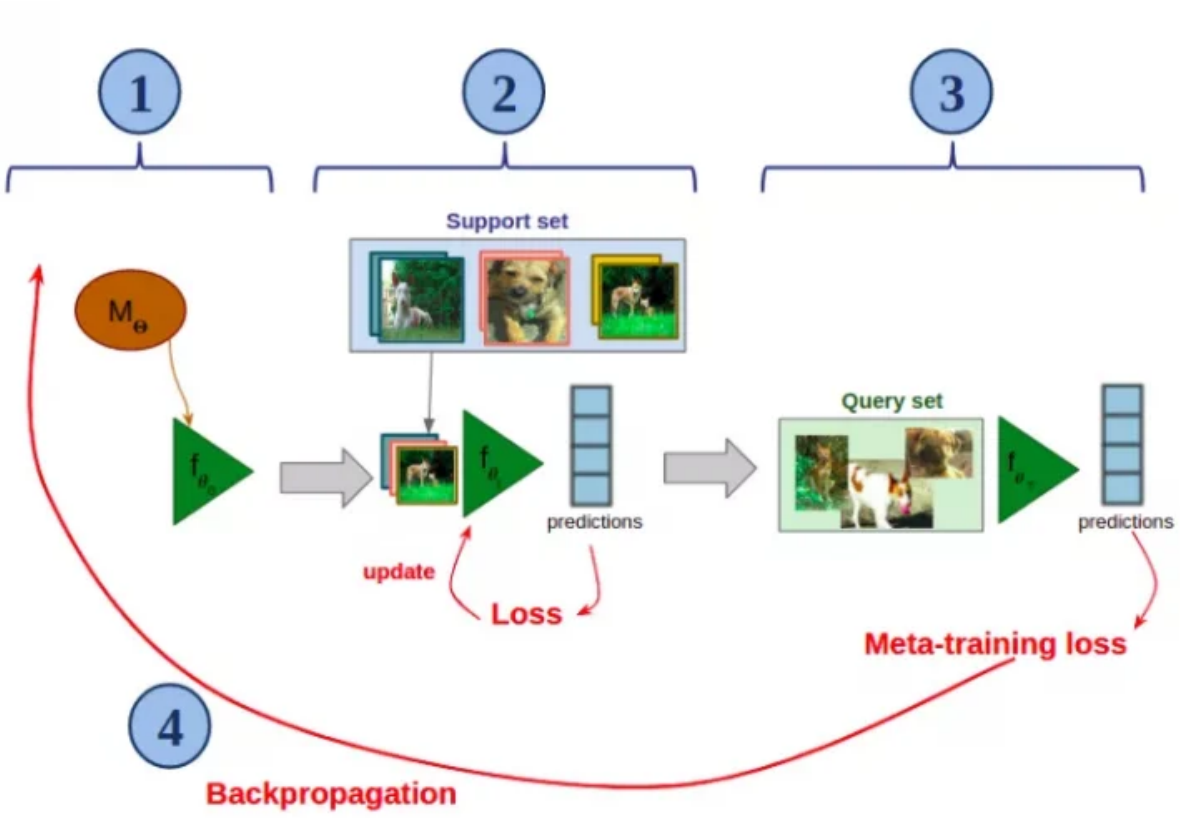

Figure 17. *Gradient-Based Meta-Learning*

4.3.3 Siamese Network

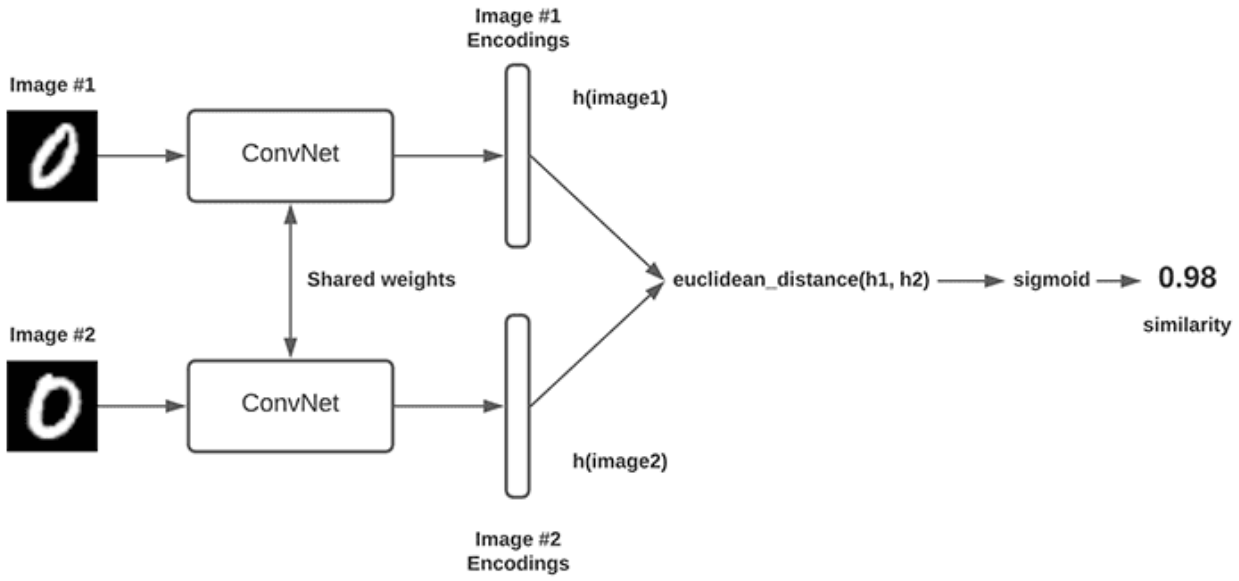

Figure 18. *Siamese Network*

The proposed Siamese Network architecture consists of two identical Convolutional Neural Networks (CNNs) that share weights. The CNNs are used to extract feature vectors from the input images. The feature vectors are then passed through a Euclidean Distance function to calculate the distance between them. The output of the Euclidean Distance function is then passed through a final output layer that predicts the similarity score between the two images.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} \left(q_i - p_i\right)^2}$$

- p, q: two points in Euclidean n-space
- q_i , p_i : Euclidean vectors, starting from the origin of the space (initial point)
- n: n-space

We built a Siamese neural network with a shared convolutional neural network (CNN) and a Euclidean distance layer. The input to the network is a pair of images. The shared CNN consists of four blocks, each containing a 2D

convolutional layer, batch normalization, and ReLU activation function. The depth of the filters in each block is increased by a factor of 2, with a starting depth of 64. The output of the final block is passed through a global average pooling layer to produce a feature vector of size 64. The two input images are fed into the shared CNN, resulting in two feature vectors. These feature vectors are then passed to a Lambda layer that computes the Euclidean distance between the feature vectors. The output of the Lambda layer is then passed to an intermediate dense layer of size 1 x 1 with a sigmoid activation function to learn the similarity between the two input images before passed to a Dense layer with a sigmoid activation function to obtain the similarity score. The reason why we chose the tanh activation function for the intermediate layer is that the tanh function is more effective at reducing the gradient vanishing problem during backpropagation than the sigmoid function. When the sigmoid activation function is used, the gradient can become very small or zero, which can cause the model to train very slowly or not at all. In contrast, the tanh function has a steeper gradient, which can help the model to converge more quickly. The model is trained to minimize the contrastive loss between the true and predicted similarity scores.

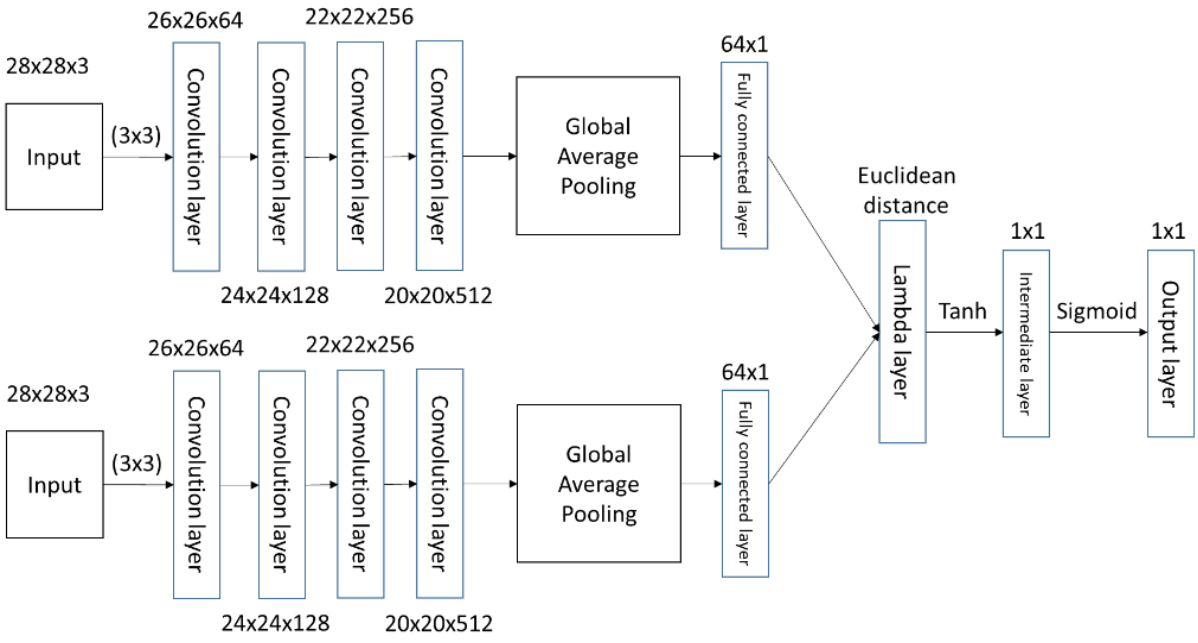

Figure 19. *Siamese Network architecture*

The input is a pair of images so we will pair up all images in our dataset in all possible pairs.

4.3.4 Nearest Prototype Classification

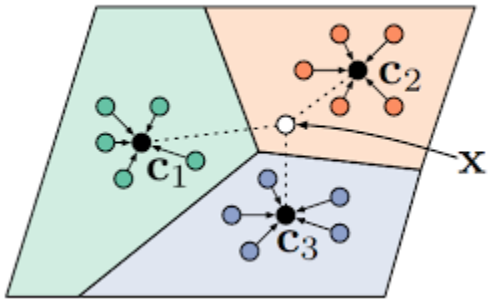

Figure 20. *Metric-Learning using Nearest Prototype Classification*

Nearest prototype classification is a type of machine learning algorithm that falls under the category of prototype-based models. The goal of this algorithm is to classify new data points based on their similarity to a set of prototypes, which are representative examples from each class.

The algorithm works by first training on a labeled dataset, where it calculates the prototype for each class by computing the mean vector of all the examples in that class. During classification, a new data point is assigned to the class whose prototype is closest to the data point, measured by some distance metric (e.g. Euclidean distance, cosine similarity).

This is a Metric-Learning approach to solve few-shot learning problems if we build a CNN to extract feature vectors from training images then we calculate mean feature vectors of each class and calculate new image's feature vector distance to mean future vectors,the new image belong to the class that have the shortest distance between mean feature vector and new image's feature vector.

We built a CNN to extract feature vectors of dataset images. The input shape of the network is (200, 200, 3). The first block of the network consists of two convolutional layers with 64 filters and a kernel size of (3, 3) followed by a max pooling layer with a pool size of (2, 2). This block is designed to capture low-level features of the input images. The second block of the network consists of two convolutional layers with 128 filters and a kernel size of (3, 3) followed by an average pooling layer with a pool size of (2, 2). This block is designed to capture higher-level features of the input images. Finally, the flatten layer is used to convert the output of the last pooling layer into a one-dimensional vector, the feature vector that we need.

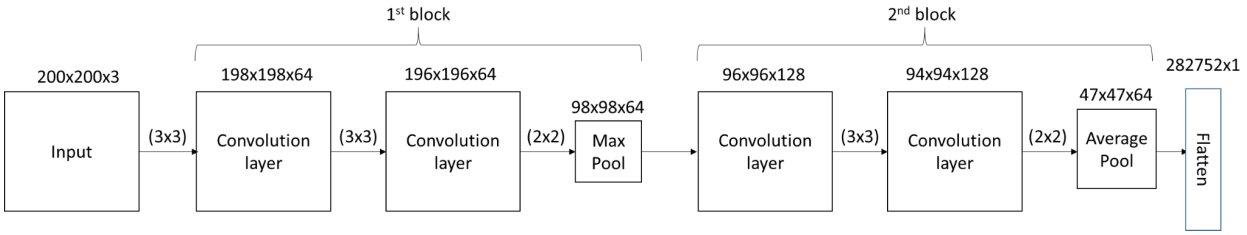

Figure 21. *CNN feature extractor*

After we extract all the image's feature vectors, we found the mean feature vector of each class. By computing the Euclidean distance between the feature vector of the new image and mean feature vectors of each class,we then find the class with the smallest distance from the new image's feature vector. This is how we perform image recognition using the Metric-Learning approach with the Nearest prototype classification method.

4.4 Training and Results

4.4.1 Traffic signs detection

The detection of traffic signs is a challenging task due to the variations in shape, color, and size of the signs, as well as the diverse lighting and weather conditions under which they can be captured. To overcome these challenges, we

used the state-of-the-art YOLOv5 object detection framework, which has shown excellent performance on various object detection tasks.

We divided our dataset of 900 images into three subsets: 900 images for training, 180 images for validation, and 90 images for testing. We used the training set to train our YOLOv5 model to detect and recognize traffic signs. The validation set was used to monitor the model's performance during training and prevent overfitting, while the test set was used to evaluate the model's performance on unseen data. We employed data augmentation techniques such as random scaling, rotation, cropping, and flipping to increase the diversity of the training data and improve the model's generalization performance.

We trained the model for 150 epochs in the first run and for 50 epochs in the second run using the stochastic gradient descent (SGD) optimizer with a learning rate of 0.01, momentum of 0.937, weight decay of 0.0005, and cosine annealing learning rate schedule. The model was trained on an AMD Radeon(TM) Vega 10 Graphics GPU, which provided fast training times and efficient memory usage. During training, we monitored the loss and accuracy of the model on the validation set after each epoch to ensure that the model was not overfitting to the training set.

The runtime of the first run was 51.06 minutes, while the runtime of the second run was 12.84 minutes. We used a batch size of 15 for both runs, which provided a good balance between memory usage and training speed.

For the first run of training, we observed that the model's performance continued to improve as the number of epochs increased up to around 140 epochs, after which the performance started to plateau.

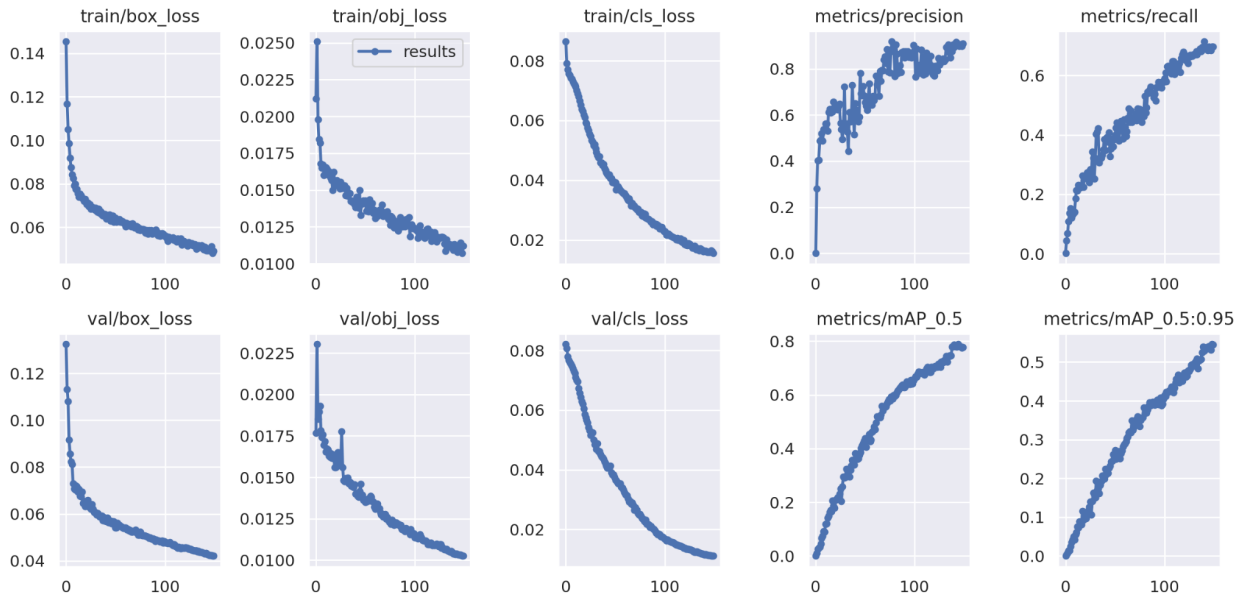

Figure 22. *Training results of the first run*

Figure 22 shows the training result of the first run. The model achieved precision score of 0.9, recall score of 0.69, mean average precision at 0.5 IOU (mAP 0.5) of 0.778 and mean average precision at 0.5:0.95 IOU (mAP 0.5:0.95) of 0.544 after 150 epochs of training. We observed significant decrease of bounding box loss, objectness loss and classification loss on both training set and validation set. On the training set, bounding box loss, objectness loss and classification loss went from 0.145, 0.021 and 0.086 at the first epoch to 0.049, 0.011 and 0.015 at the 150th epoch. And on the validation set, they went from 0.132, 0.017 and 0.082 at the beginning to 0.042, 0.010 and 0.011 after 150 epochs.

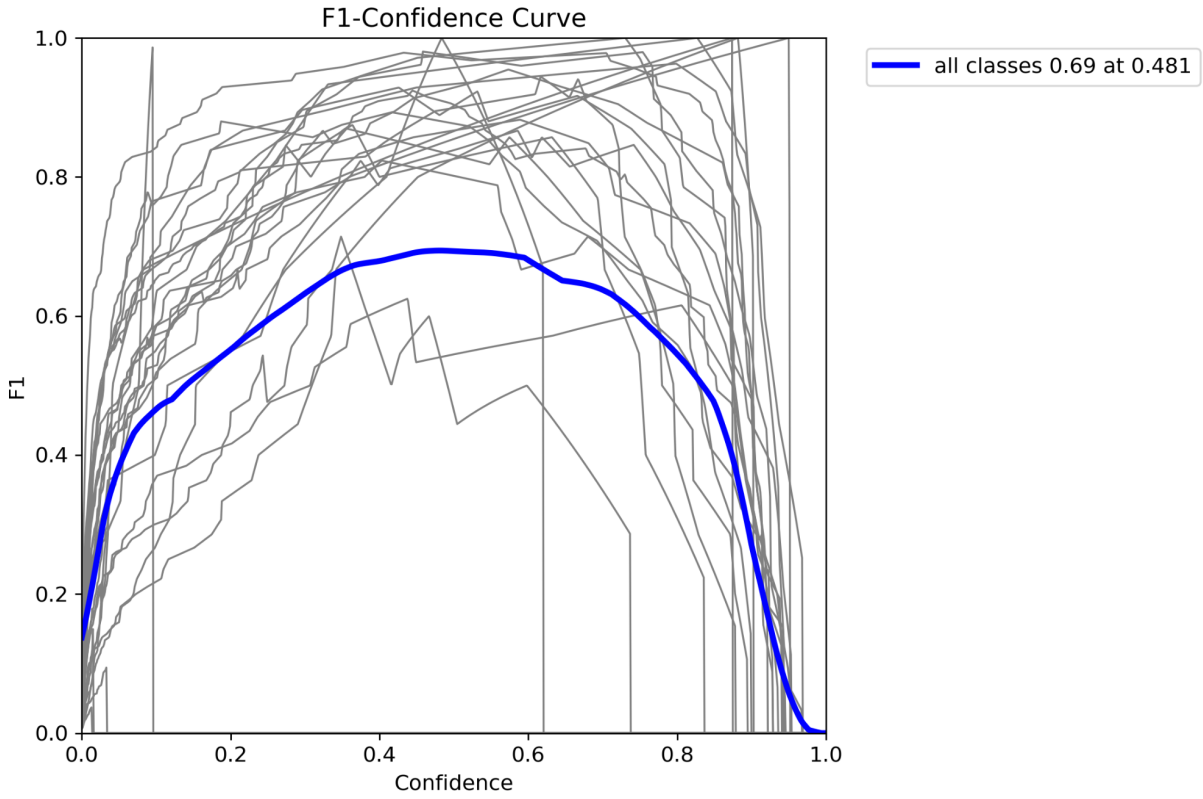

Figure 23. *F1 score of the first run. Maximum F1 score of 0.69 at confidence threshold of 0.481*

After training, we evaluated the performance of the model on the test set using precision, recall, and F1-score metrics. The precision measures the proportion of true positives (correctly detected signs) out of all detections, while the recall measures the proportion of true positives out of all actual positive instances (signs). The F1-score is the harmonic mean of precision and recall and provides a single measure of the model's overall performance.

We realized that the results did not meet our expectations and we believed that the results could be improved by continuing to train with more epochs. That is why we decided to train for the second run.

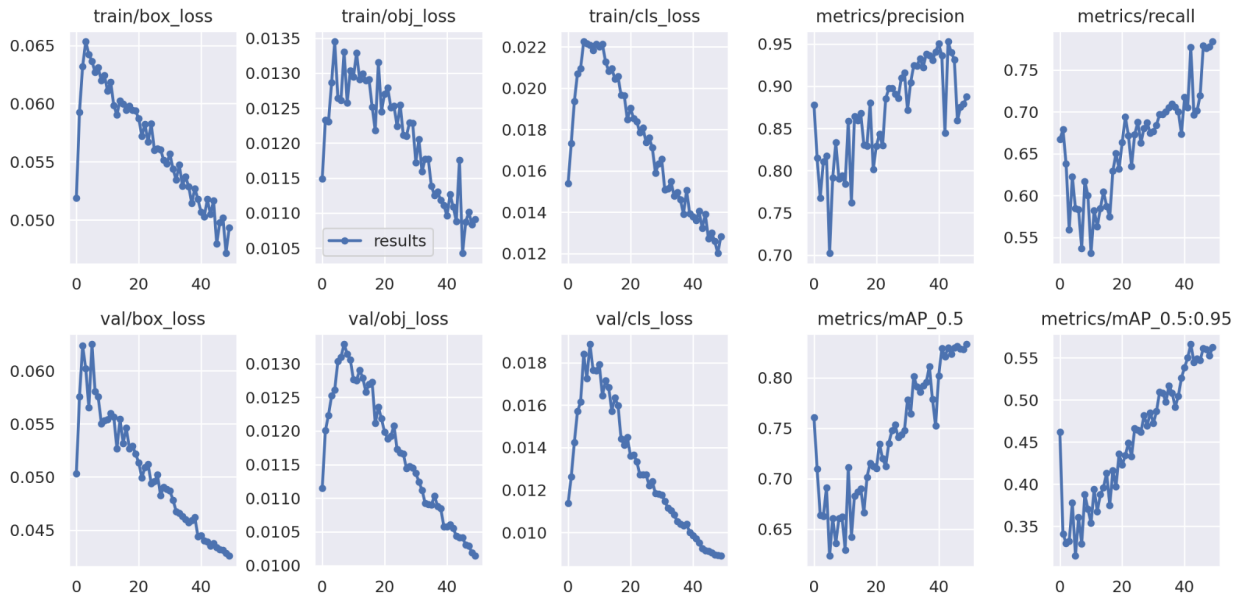

Figure 24. *Results of the second run*

Figure 24 shows the results of the second run. In the second run of training, we observed that the model's performance went down at the early stage of training, around the first 10 epochs, then improved again till the 50th epoch . The model achieved a precision score of 0.887, recall score of 0.78, mAP 0.5 of 0.83 and mAP 0.5:0.95 of 0.56 after 50 epochs of training. Again we observed a slight increase then dramatically decrease of bounding box loss, objectness loss and classification loss on both training set and validation set. On the training set, bounding box loss, objectness loss and classification loss went from 0.051, 0.011 and 0.015 at the first epoch to 0.062, 0.013 and 0.022 at the 9th epoch then down to 0.049, 0.010, 0.012 at the 50th epoch. And on the validation set, they went from 0.050, 0.011 and 0.011 at the first epoch to 0.055, 0.013 and 0.017 after 9 epochs then came down to 0.042, 0.010 and 0.008 at the last epoch.

After the second run, we have come to the conclusion that YOLOv5's accuracy had gone down a little bit compared to the first run but all other metrics and losses had been improved hence the model's performance had been better. Overall, we had made a good decision to go for the second run since the first one result did not meet our expectations. The model had been improved and our intentions with object detection had been going in the right direction.

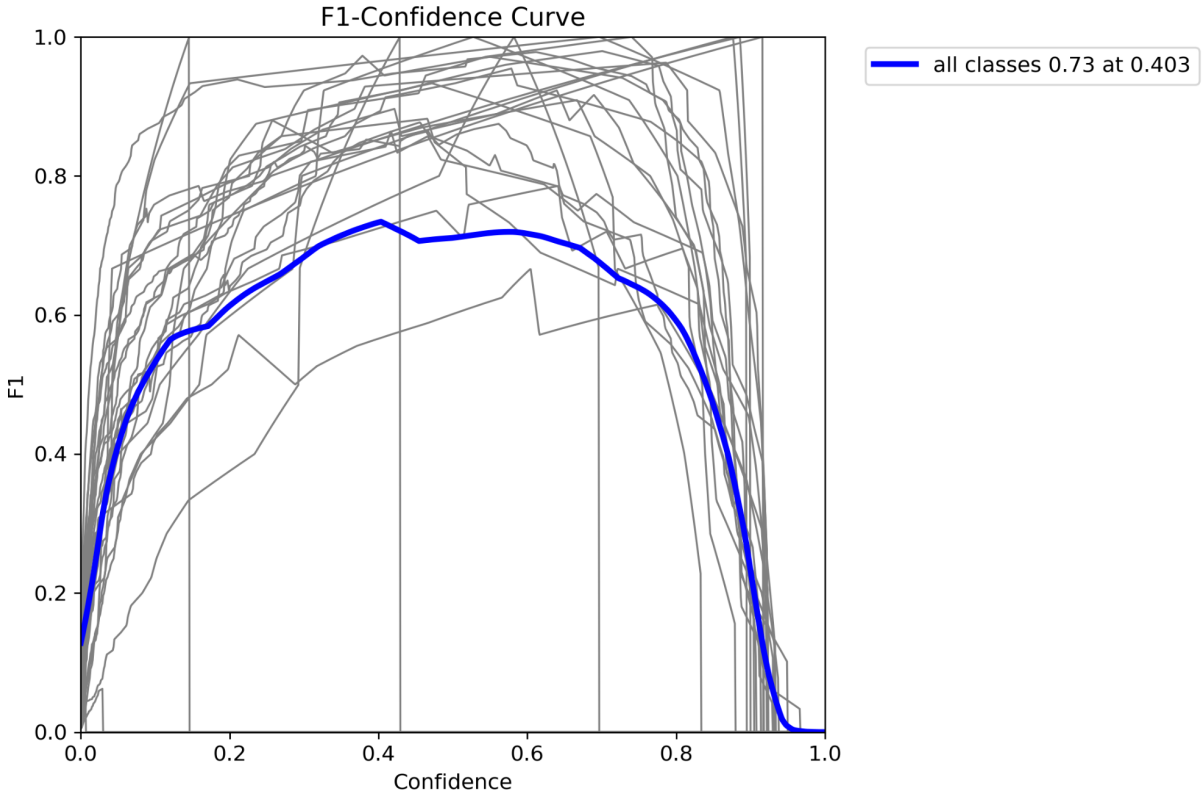

Figure 25. *F1 score of the second run. Maximum F1 score of 0.73 at confidence threshold of 0.403*

Figure 26. *Result of YOLOv5 performance*

The performance of YOLOv5 on object localization tasks is quite remarkable considering the nature of the dataset set, number of epochs and average runtime. Even though YOLOv5 did not perform too well on classification tasks, YOLOv5 solved the traffic sign detection problem accordingly. Thanks to that we move to the next part of our research, using YOLOv5 results for few-shot learning traffic sign recognition.

4.4.2 Traffic sign recognition

After we successfully detected traffic signs in images with bounding boxes, we decided to crop the result into new images containing only traffic signs. We then built a new set of data from crop images. We have to resize all crop images to 60x60x30 due to the variety of size of the bounding boxes. We decided to eliminate some classes because of the insufficient images from that class. We now have 24 classes: construction danger, direction, height limit, indication, intersection danger, no car entry/turning, no more prohibition, no motorbike entry/turning, no parking, no stopping and parking, no truck entry/turning, no turn left, no turn right, no u and left turn, no u turn, one way prohibition, pedestrian crossing, pedestrian danger, road danger, slow warning, speed limit, vehicle and speed permission lane, vehicle permission lane and weight limit.

Figure 27,28,29. *Crop images of traffic signs vary in sizes*

To perform a few-shot learning classification, we only take 10 - 12 images each class for the training set, a total of 242 images on the training set and 1 - 3

images each class for the test set, a total of 26 images on the test set. This is a very small amount of data for training with any normal Convolutional Neural Network (CNN), it will cause poor performance due to overfitting.

We trained our CNN model (**Figure 14**) for 150 epochs for about 14 minutes, achieved an accuracy of 0.85 on the training set and 0.92 on the test set. However, the model's performance on unseen images is extremely inadequate due to overfitting, with the accuracy of 0.1. This happened the same on other classic CNN models such as Lenet-5, AlexNet, MobileNetV1.

Figure 30. *Poor performance due to over fitting of normal CNN model*

To train our Siamese model, first we proceeded to pair images in the support set and the query set to fit into the input layer of the Siamese model. The images are paired up in all possible pairs, 242 images in the support set made 58564 pairs and 26 images in the query set made 676 pairs. Then we trained our Siamese model for 100 epochs using the Adam optimizer with a learning rate of 0.001, batch size of 32, binary cross entropy loss and accuracy metrics.

To prevent overfitting, we used early stopping, which stopped the training process if the validation loss did not improve for a certain number of epochs. After

19 epochs, the model stopped running and achieved an accuracy of 0.9966 on the training set and 0.997 on the test set; the training set loss was 0.01 and validation set loss was 0.0101. The runtime was 26.65 minutes. This is so much faster than training with Lenet-5, AlexNet and MobileNetV1, which usually took more than an hour.

We measured the performance of the Siamese Network model by comparing new images with train images to see the probability of those 2 images belonging to the same class, the probability threshold is 0.5. The model performed better with new images than using normal CNNs. **Figure 32** shows the similarity evaluation process between two images A and B (**Figure 31 a&b**)

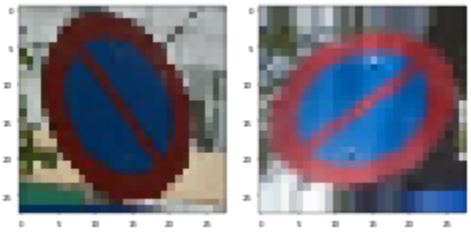

Figure 31 a,b. *Image A from test set and image B from train set*

```
1/1 [==============================] - 0s 21ms/step
Similarity score: 0.9880977
Same class
```

Figure 32. *Siamese model recognize image A from test set and image B from train set are belong in the same class*

In the metric learning method using Nearest prototype classification, we use Euclidean distance to compute distance between image's feature vectors and mean feature vectors of each class as the key to classify traffic sign class after feature extraction. The accuracy on training was 0.62 but this method performed incredibly well on unseen images. And also because Nearest prototype classification is not a deep learning method, just the feature extraction CNN is a deep learning method, the runtime of Nearest prototype classification was

incredibly fast. **Figure 32** illustrates Nearest prototype classification on a test image and compares prediction with true value.

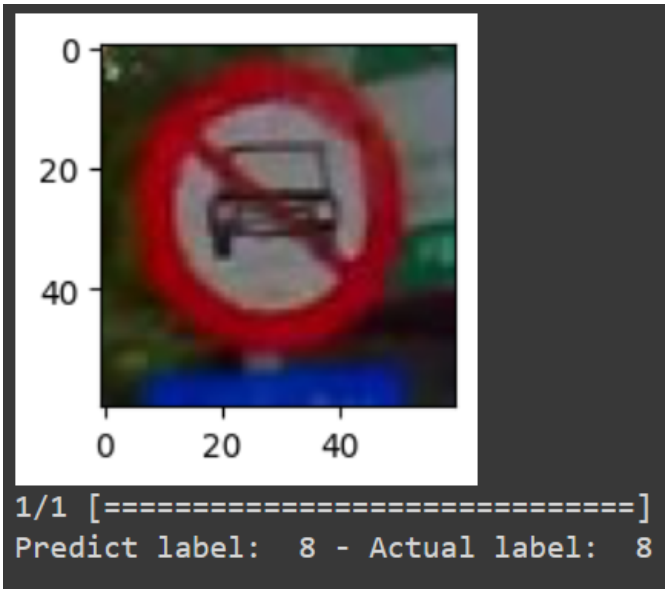

Figure 33. *Metric-Learning Nearest prototype classification method predict test image's class*

The performance of both Metric-Learning methods was good based on the result of image recognition on unseen images. However, the Nearest prototype classification based method was not a CNN classification model, it was just an algorithm to recognize new images by comparing new images feature vectors to all classes' mean feature vectors. On the other hand, the Siamese Neural Network performed really well but we can improve by training for more epochs, enhancing the architecture or we could try to increase data training samples.

5 Conclusion

The recognition of traffic signs presently plays a crucial role in enhancing the effectiveness of intelligent traffic systems. This tool enables drivers to promptly and effortlessly respond to signals or situations encountered while driving.In this research, we demonstrate how a computer vision system equipped with machine learning and hyper-learning approaches may enhance the accuracy with which drivers can recognize traffic signs using in-vehicle technologies. connected by an intelligent system.

In this research, we had built our own dataset by collecting images and labeled these images for YOLOv5 object detection. We had successfully detected traffic signs from a traffic frame with high accuracy then cropped the result to build a small dataset, or support set in terms of few-shot learning, to train a few-shot learning model. The few-shot learning model we built and trained was the Siamese network to classify traffic signs into appropriate classes. The Siamese network had worked really well on unseen images. We also built a CNN model to extract feature vectors of our few-shot support set then calculated mean feature vectors of every class. This idea was based on the Nearest prototype classification algorithm, using Euclidean distance to find the shortest distance between an unseen image and a class to perform classification. Both Siamese network and Nearest prototype classification metric-learning methods achieved high accuracy performing on unseen data, with small training data, short runtime training and saving computational power.

In addition, the driver and the vehicle's visual system are better able to interpret low-frequency traffic signals in real time with the assistance of this technology.

Data categorization and picture recognition are the two main components of an image recognition system. We were able to precisely pinpoint the locations of road signs using YOLO5, and then send that determined dataset to Few-shot Learning methods for classification. In addition, Few-shot Learning allows us to save money during the data classification process and aids in the categorization of uncommon data.

During our research, we also faced a few challenges. First is the amount of data required for training YOLOv5. YOLOv5 needs a lot of data to be able to perform detection tasks efficiently. Second is data of Vietnam traffic are poor quality images and imbalance in traffic signs classes distribution. Some of them are too many but some are very rare and even not found. And the last one is how to find the best way to optimize our Siamese model. We had to build our own shared weight CNN and chose the most appropriate hyperparameters to optimize Siamese network performance.

We see this technique being refined and eventually used for computer vision-based traffic sign identification, making it easier for drivers to identify and react appropriately to different traffic sign types. Additionally, we will attempt to enhance the model's performance with other traffic signs by amassing additional data on the sign's structure and features and using other cutting-edge CNN models like,Faster-RCNN,... to enhance detection accuracy. The sign layers might be expanded, and the sign type could be described in more depth, among many other possible enhancements. We look forward to combining object detection methods with few-shot learning to create an end-to-end few-shot detection model. We will also work to refine and extend the model so that it may be utilized by drivers and deployed on roads.

Contribution

Đỗ Xuân Bách	- Prepare data - Studied, coded and trained YOLOv8 - Studied how to use Latex to write thesis - Written data preparation
Đỗ Đặng Gia Vương	- Written Acknowledgement, Declaration, Abstract - Written Introduction - Written Related works - Written YOLOv5 Object Detection Methodology - Written Conclusion - Written References
Nguyễn Chính Đạt	- Prepared data - Written data preparation - Written Few-shot Learning Object Recognition Methodology - Trained YOLOv5 - Coded and trained Siamese network, Nearest prototype classification method - Compared results with other methods - Written Training and Results

References

[1] C. Dewi, R.-C. Chen, Y.-T. Liu, X. Jiang, and K. D. Hartomo, "Yolo V4 for Advanced Traffic Sign Recognition With Synthetic Training Data Generated by Various GAN," *IEEE Access*, vol. 9, pp. 97228–97242, 2021, doi: 10.1109/ACCESS.2021.3094201.

[2] Y. Jin, Y. Fu, W. Wang, J. Guo, C. Ren, and X. Xiang, "Multi-Feature Fusion and Enhancement Single Shot Detector for Traffic Sign Recognition," *IEEE Access*, vol. 8, pp. 38931–38940, 2020, doi: 10.1109/ACCESS.2020.2975828.

[3] J. Zhang, Z. Xie, J. Sun, X. Zou, and J. Wang, "A Cascaded R-CNN With Multiscale Attention and Imbalanced Samples for Traffic Sign Detection," *IEEE Access*, vol. 8, pp. 29742–29754, 2020, doi: 10.1109/ACCESS.2020.2972338.

[4] X. Bangquan and W. Xiao Xiong, "Real-Time Embedded Traffic Sign Recognition Using Efficient Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 53330–53346, 2019, doi: 10.1109/ACCESS.2019.2912311.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 779–788, 2016.

[6] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 7263–7271, 2017.

[7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, doi: 10.48550/ARXIV.1804.02767.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, doi: 10.48550/ARXIV.2004.10934.

[9] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, vol. 9905, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[10] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2980–2988, 2017.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014.

[12] R. Girshick, "Fast r-cnn," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1440–1448, 2018.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2961–2969, 2017.

[15] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[16] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, “Matching networks for one shot learning,” *Adv. Neural Inf. Process. Syst.*, p. 29, 2016.

[17] Z. Zhao, P. Tang, L. Zhao, and Z. Zhang, “Few-Shot Object Detection of Remote Sensing Images via Two-Stage Fine-Tuning,” *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022, doi: 10.1109/LGRS.2021.3116858.

[18] X. Wang, Z. Cai, D. Gao, and N. Vasconcelos, “Towards universal object detection by domain attention,” *Proc. IEEECVF Conf. Comput. Vis. Pattern Recognit.*, pp. 7289–7298, 2019.

[19] G. Han, S. Huang, J. Ma, Y. He, and S.-F. Chang, “Meta Faster R-CNN: Towards Accurate Few-Shot Object Detection with Attentive Feature Alignment,” *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 1, pp. 780–789, Jun. 2022, doi: 10.1609/aaai.v36i1.19959.

[20] Flood Sung, YongXin Yang, Li Zhang, Tao Xiang, Philip H.S Torr and Timothy M.Hospedales, “Learning to Compare: Relation Network for Few-Shot Learning”.

[21] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, “A Simple Framework for Contrastive Learning of Visual Representations”.

[22] Marko Horvat, Ljudevit Jelečević, Gordan Gledec, “A comparative study of YOLOv5 models performance for image localization and classification”. *Unska 3, HR-10000 Zagreb, Croatia.*

[23] Junfan Wang, Yi Chen, Mingyu Gao, Zhenkang Dong, “Improved YOLOv5 network for real-time multi-scale traffic sign detection”.