APS360 PROJECT FINAL REPORT

Jinghang Zhu
Student# 1008965639
jinghang.zhu@mail.utoronto.ca

Yahe Zhang
Student# 1007676105
yahe.zhang@mail.utoronto.ca

Ruibo Zhang
Student# 1008774391
ruibozhang.zhang@mail.utoronto.ca

1 INTRODUCTION

With the rapid development of autonomous driving and advanced driver assistance systems (ADAS), traffic sign recognition has become crucial in intelligent transportation systems. Accurately locating and identifying traffic signs through car cameras can enhance driving safety and efficiency, helping vehicles adapt to various traffic environments and avoid driver errors or accidents. Given the variations in traffic signs due to lighting, weather conditions, and possible obstructions, building a reliable recognition system is essential for the success of autonomous driving technology.

The goal of this project is to build a deep learning model capable of accurately classifying different types of traffic signs. To make the model more user friendly, it takes an image containing traffic signs as input and outputs the corresponding category of each sign. Hence, in addition to classification, the model also needs to perform detection. Deep learning models, such as YOLO, have shown remarkable performance in image recognition and detection tasks by automatically detecting the location of the searching items and extracting high-level features from images for classification. Given the diversity in traffic sign shapes, colors, distance, and clarity, which can also be influenced by lighting and obstructions, traditional rule-based or handcrafted feature methods fall short. Machine learning models, however, can learn these complex patterns from a large dataset, improving detection/ recognition accuracy. Therefore, deep learning is suitable for developing a traffic sign detection and recognition system. This approach effectively handles the complexity of traffic sign images under different conditions, providing greater adaptability and robustness than traditional methods.

2 ILLUSTRATION / FIGURE

For our primary model, we call it as CNN classifier model with YOLO detection. It has 3 components: YOLO Backbone, VGG (for transfer learning), and a CNN classifier.

We first trained the CNN classifier with VGG independently to achieve optimal classification performance. Then we embedded it into the YOLO architecture to test its real-world application and further refine its performance.

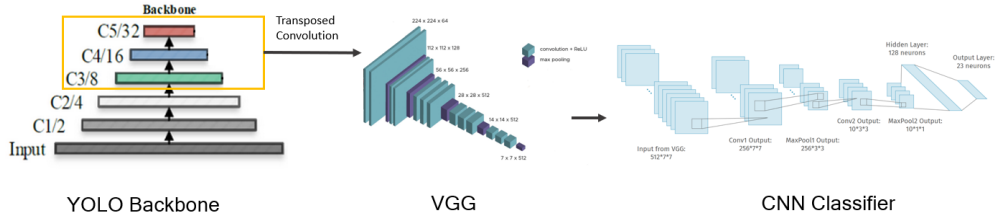

Figure 1: Overall Structure

3 BACKGROUND & RELATED WORK

Recognizing traffic signs has been widely utilized in various fields to help humans achieve their purposes. We are able to feel the real power of this model application according to the cases below.

Tesla is the typical representative in the autopilot area. For instance, Tesla Autopilot uses the vehicle's forward-facing camera to detect the road markings, and make the corresponding decisions based on received information (i.e. slowing the car for all traffic lights). Tesla's detecting system is highly correlated to our recognizing traffic sign model so that the precision of the system is also the goal of the project's model.

Google Maps is also a powerful navigation tool for humans to find the quickest route. It requires an up-to-date and huge database. To acquire more street views' data, Google has a team including street view cars to collect 360° imagery which will be used to power the world map. Then recognizing traffic signs model will play a vital role in processing data that almost all updated traffic signs will also be collected by Googles' database.

Traffic congestion is always a non-negligible issue. In 2013,Lanke & Koul published a journal about smart traffic management systems. The system includes smart traffic sensors, decision making algorithms and remote controlled light and traffic signs. The sensor is to collect real-time traffic conditions. Then the system will respond to eliminate possibilities of causing traffic jams by changing the traffic lights' gap, speed limit, etc.

Robot delivery service is also a popular project that uses cameras to perceive the surrounding environment in recent years. In 2021, a video published by the Canadian Press recorded a food delivery robot company, had tested the robots' functionality in Toronto's streets. It used the camera to detect the real-time environment and would respond road markings. The recognizing traffic signs model had been integrated in the robots' algorithm.

In addition, an article published by Chanana et al. (2019) stated that the travel of pedestrians with visual impairment had been improved by tailored technological equipment such as cameras in recent years. When a human wants to cross the street, the camera will recognize the traffic lights and show the audio signals to indicate the state. Thus, our project is capable of assisting to ensure the safety of the group with visual impairment.

4 DATA PROCESSING

For our project that involves traffic sign recognition, we collected the dataset from the site Kaggle which includes two main files: images of Chinese street signs and their YOLO labels. Inside each folder, it contains a train and validation folder. Here's a detailed description of the steps we took to clean and format the data, ensuring it is suitable for model training and evaluation.

Since the dataset contains more than 20000 images, we only downloaded 5000 images to keep a reasonable amount and ensure training will be completed in a reasonable amount of time. After downloading the 5000 images and their corresponding labels, we analyzed the number of classes and then found classes with limited images. This step allowed us to find redundant classes or classes with not enough samples. We removed classes that were either redundant or had too few images. Then we manually chop the actual street signs components of the images and place them into separate folders based on their class labels; each folder was named after its respective class. Lastly, we checked the number of images in each class to confirm that they were sufficiently balanced for training. These will be used for the baseline and CNN model, which cannot detect street signs. However, the images varied in size. To standardize the input for the model, we resize all pictures to 224*224 pixels, the most common size in the dataset. Here's an example of a cleaned training sample after processing:

Figure 2: Data Samples

However, upon inspection, the number of images in each class is not a lot. Hence, we decided to use a combination of manually chopped images, pictures from another dataset, and images taken by friends in China.

To expand our dataset, we used the dataset from the Traffic Sign Recognition Database, which includes two main files: TSRD-Train and TSRD-Test. We then found classes where the YOLO dataset also contains and selected images accordingly while maintaining a balanced dataset. These images are already centered and sign-focused. To enhance the diversity of our training set, We randomly sample 50% of the data from the training set for data augmentation. This included random transformations such as scaling, slight rotation, flipping, and color changes to expand the dataset and generate additional variations of the training images, which helped improve model robustness. Lastly, to simulate realistic inputs, we asked our friends to take pictures of street signs in China and take pictures of the same sign at different times of the day, angles, and weather. This prove that our program can be used in real-life applications despite the time and weather. We then split the dataset into three subsets: training set with 1856 samples, validation set with 389 samples, and testing set with 417 samples.

After processing, we ensured that the images in each class were well-distributed. For instance, here's a table of the number of images per class in the training set (Figure 2). Lastly, we kept some fresh images for the final model evaluation.

Class	Number of Images
Be aware	156
Bike	150
Make way	110
No changing lane	128
Cannot drive on this lane	142
No honk	130
No left turn	138
No park	148
Cannot go straight	70
Cannot turn	96
Go to right lane	126
Rail road ahead	56
School zone	58
Stop	162
Stop (word version)	36
Can turn right	100
Speed limit: 5	118
Speed limit: 30	80
Speed limit: 40	176
Speed limit: 50	98
Speed limit: 60	154
Speed limit: 70	78
Speed limit: 80	152
Automation only	70

Figure 3: Class distribution of traffic sign images

5 ARCHITECTURE

Our primary model contains 3 components: YOLO Backbone, VGG, and CNN Classifier.

YOLO Backbone: YOLO is used to extract features from the input image. It localizes the traffic signs, producing feature maps that capture the visual patterns (such as shapes, edges, and colors) of the signs. Although YOLO also performs object detection, for this purpose, we focus on its ability to generate useful feature maps for classification.

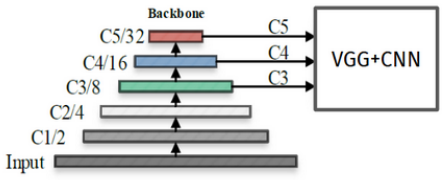

Figure 4: YOLO Backbone Architecture

VGG (Transfer Learning): The output from YOLO's backbone (512 channels) is passed to a pre-trained VGG model (e.g., VGG16). The VGG model refines the extracted features using its pre-trained knowledge from large datasets like ImageNet. This step leverages transfer learning to enhance the quality of the features for the subsequent classification.

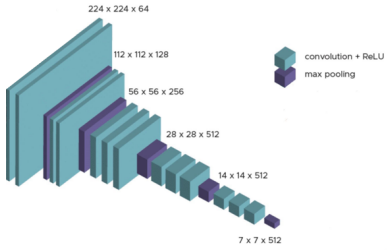

Figure 5: CNN Classifier Architecture

CNN Classifier: The refined features from VGG are processed by our custom CNN classifier:

Conv1: 256 filters (2x2) for feature extraction, reducing spatial dimensions.

Max Pooling: Helps reduce the size while preserving important features.

Conv2: 10 filters (2x2), further processing the features.

FC1: Fully connected layer with 128 neurons to abstract the features.

FC2: Final fully connected layer with 23 output neurons, each representing one of the 23 traffic sign classes.

Output: The model outputs a probability distribution over the 23 traffic sign categories using a softmax activation.

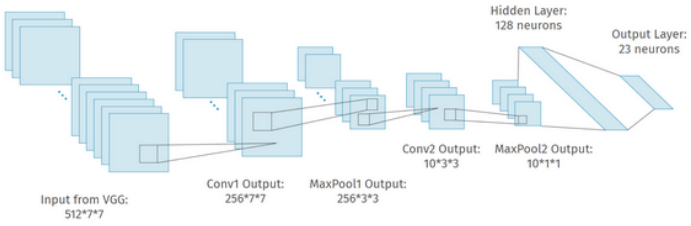

Figure 6: CNN Classifier Architecture

While the YOLO backbone handles localization, our primary focus is on feature extraction and classification. We leverage a pre-trained VGG model for feature refinement, followed by a custom CNN classifier that makes the final decision. This classifier plays a key role in ensuring accurate traffic sign recognition, which is essential for real-world applications.

6 Baseline Model

We decided to use the ANN classification as our baseline model because it is able to receive raw data like pixel values, have the ability to analyze pictures' features and reproduce the same result easily. To set our detailed model, we choose Relu function as our activation function to avoid linear computing, Cross-Entropy as loss function which is helpful in classification problems and SGD with momentum as the optimizer since it can randomly evaluate the dataset to ensure the robustness and take less time to train the model. With parameters above, the ANN also manages to output 23 types of classification results in the output layer, which meets the desired outcomes. Last but not least, the ANN is efficient to process and learn from large datasets, which is applicable to our training purpose.

The baseline ANN model contains three layers, including one input layer with 500 neurons, one hidden layer with 100 neurons and one output layer with 23 neurons. Due to the input pictures with shape 3*160*149, we need to flatten the data to the lower dimension that causes numerous inputs. Thus a hidden layer is added to the model as a function of buffer to gain more details about data, and then output the logits of 23 classes which are classified in our dataset. The structure of the baseline model is explicit to implement so that we only need to tune the batch size of data, learning rate and epochs number to find the best condition of the baseline model.

7 Quantitative Results

After training the individual CNN model with the training dataset and validation dataset, we generate a training loss of 0.001961, validation loss of 0.006442, training accuracy of 1.0, and validation accuracy of 0.9922. Those numbers represent that the CNN model with VGG has a relatively good predictive ability with very few errors. We also generate a confusion matrix from the test dataset to calculate the models' precision of 0.9885 and recall of 0.9874. The high precision and recall prove the high performance of the model.

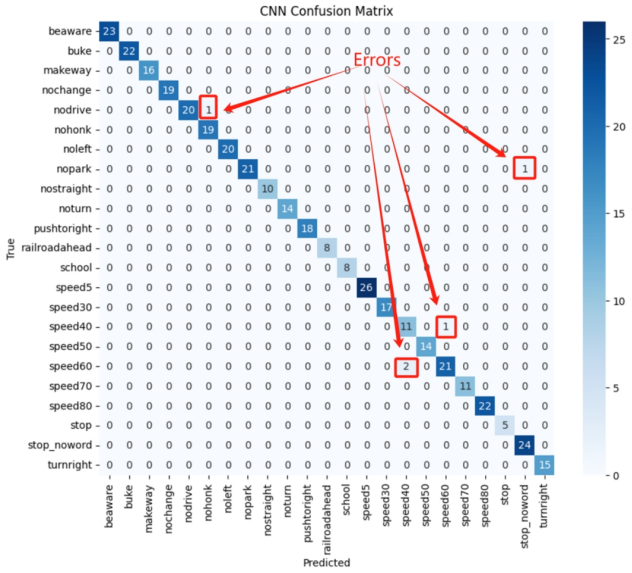

Figure 7: Confusion matrix for CNN classifier

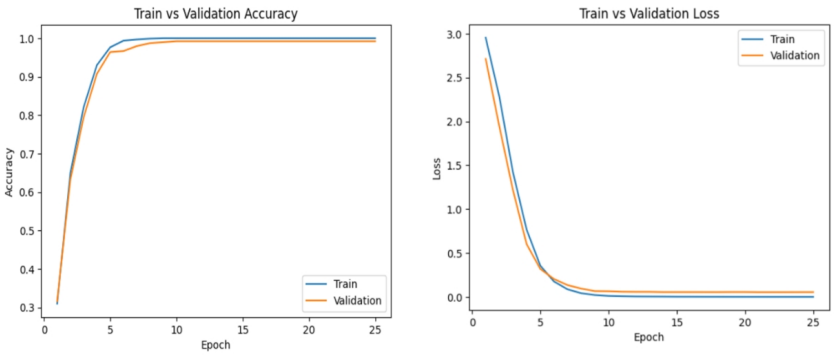

Figure 8: Accuracy curve and loss curve

We also trained the CNN classifier model with YOLO detection. The model achieved a precision of 0.702, indicating that the classifier is effective at classifying traffic signs when integrated as part of the YOLO model in real-world applications.

8 Qualitative Results

Our CNN classifier is able to recognize the classes of given pictures about the traffic signs in different conditions. As the figure shown, it outputs the corresponding class of the input picture data. From the quantitative result, the testing accuracy 0.988 is a relatively high number to show the great performance of our model, which means it can classify the traffic signs very accurately. However, there is also a class that the model does not do well on shown in the confusion matrix, which is speed60. Thus, we generate this sample output to show the worst case of our model. The error rate for this class is 8.7%, which is not considered as a really high error. In addition, the classifier gets high performance in classifying all the other classes.

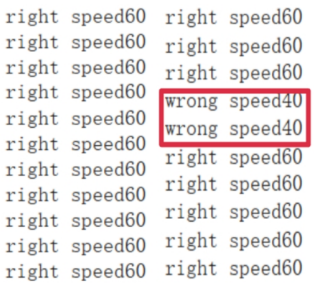

Figure 9: Worst case sample output

The classifier with YOLO detection performs exceptionally well on most of the classes, especially ”stop” and ”no access” signs. It is capable of recognizing these signs even in somewhat blurred images. However, the model somewhat struggles with specific classes, such as speed limit signs with varying values. The confidence scores for these classes tend to be lower, particularly in cases where the signs are either too small or not clearly visible.

Another significant issue observed with the model is missed detections (false negatives). However, since our primary goal is to test and further train the classifier, we are not focusing too much on this issue for now.

Figure 10: One Sample of Train Results

9 EVALUATE MODEL ON NEW DATA

Since our CNN classifier model achieves an accuracy of 0.99 on the test set, which means it has been well-trained, we decided to focus more on the CNN classifier model with YOLO detector in this part.

To ensure that the samples never used during the training process are evaluated, we leveraged previously split test data from our data processing phase. This can ensure the robustness of the general performance of the model. The confidence scores of most classes such as “stop” are still high, corresponding to the high precision of these classes.

Additionally, this model is capable of detecting and classifying multiple traffic signs within the same image. It shows high confidence in most data, except for a lower certainty when classifying the ”Only Motor Vehicles” sign. This may be due to the relatively lower precision of this class compared to others, and it could also indicate that the model struggles with smaller and more distant objects, where confidence scores are typically reduced

Figure 11: Images from test set

10 DISCUSSION

Our model demonstrates strong performance in classifying traffic signs, particularly in its ability to accurately predict classes from the testing dataset. The CNN classifier, trained with VGG features, achieves impressive results with a training accuracy of 1.0, a validation accuracy of 0.9922, and a test accuracy of 0.9900, indicating its robustness in recognizing traffic signs with minimal errors.

For the CNN classifier model with YOLO detection, the model achieves a precision of 0.702. The precision varies across different classes. For example, in the ”stop” and ”no access” classes, precision exceeds 0.9, likely because these signs are visually distinct from other categories. However, the model struggles with classifying speed limit signs, likely due to the difficulty in distinguishing between the numbers.

While the CNN classifier performs exceptionally well on its own, the CNN classifier model with YOLO detection does not achieve the same level of performance. We believe this may be due to interference from YOLO’s detection and localization. Since our focus was not on fine-tuning the YOLO detector, localization errors may affect the precision. Additionally, the dataset may not be perfectly balanced, and for challenging categories like speed limit signs, more training data may be needed.

Moreover, considering the precision continued to rise at epoch 50, we think more epochs can improve the model a lot. However, due to the GPU limits on Google Colab, we were unable to run additional epochs. This will be an important next step in our project, and further training will be a key goal for future improvements. We will also consider implementing additional models (e.g.

RCNN) and comparing their performance on this project in order to select the optimal model. This will help us identify the most effective approach for traffic sign classification and further enhance the accuracy and robustness of the system.

11 Ethical Considerations

While automating street sign recognition offers clear benefits for safety and efficiency, there are important ethical concerns to address. It is crucial to address these limitations to ensure the product is diverse, transparent, protective, and accountable.

One major issue is the potential biases in the training data. If the dataset used to train the model lacks representation of different environments, countries, and conditions, the system could fail to recognize street signs in less-represented areas. For example, if the training data focused on warm geographic locations when inputting pictures taken from snowy conditions could cause preventable failures. Additionally, there are concerns about the transparency of how the model is trained. Transparency is important to uphold liability and trust. If the training data is sourced from public images, proper credit should be given to the creators of these datasets. Failure to acknowledge the source and licensing agreements could lead to serious ethical and legal issues.

Another concern is the invasion of privacy and oversight of regulation; for example, the training data includes images that capture faces, license plates, and other identifiable information without authorization and consent. In such cases, the pictures should be preprocessed to remove and blur any personal information before being used in training to maintain anonymity. Failing to do so could lead to privacy violations. Any misuse could lead to damaging privacy and security.

Colab Link

https://colab.research.google.com/drive/1f1Jjddpjj6N-IepAUbu7-wHc0bcNHhqZ?usp=sharing

References

P. Chanana, R. Paul, M. Balakrishnan, and P. Rao. Assistive technology solutions for aiding travel of pedestrians with visual impairment. *Journal of Rehabilitation and Assistive Technologies Engineering*, 2019. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6453076/. Accessed: 2024-10-02.

Google. *How street view works and where we will collect images next*. URL https://www.google.com/streetview/how-it-works/. Accessed: 2024-10-02.

Kaggle. TT100K. https://www.kaggle.com/datasets/braunge/tt100k. Accessed: 29 November 2024.

N. Lanke and S. Koul. *Smart Traffic Management System*. URL https://www.researchgate.net/publication/260833477_Smart_Traffic_Management_System. Accessed: 2024-10-02.

The Canadian Press. Food delivery robots have hit canadian sidewalks - but there are roadblocks. *Global News*. URL https://globalnews.ca/news/9324385/food-delivery-robots-canada-challenges/. Accessed: 2024-10-02.

Tesla. *Model Y Owner's Manual*. URL https://www.tesla.com/ownersmanual/modely/en_eu/GUID-A701F7DC-875C-4491-BC84-605A77EA152C.html. Accessed: 2024-10-02.