Empirical Study: Road Sign Classification

Bao Hoang*
hoangbao@msu.edu
Michigan State University
East Lansing, MI, USA

Tanawan Premsri*
premsrit@msu.edu
Michigan State University
East Lansing, MI, USA

Abstract

The rapid advancement of autonomous driving technology has highlighted the critical need for robust road sign detection systems. This project leverages Computer Vision and Deep Learning Networks to develop and evaluate models for detecting and classifying road signs. Diverse datasets, including images from Google Images, Google Shopping, and Kaggle, are collected and used to train various advanced architectures. Performance is assessed by splitting the data into training (80%) and test (20%) sets, with additional evaluations conducted on different data sources, such as Kaggle and Google. Furthermore, incorrect model predictions are analyzed to identify underlying causes and improve model accuracy. Our codes are provided in https://github.com/hoangcaobao/CSE881-Fall2024-FinalProject. Our deployed website URL is https://appdeploytest-wv9jmdfc27xqm9crhs3uad.streamlit.app/.

ACM Reference Format:
Bao Hoang and Tanawan Premsri. 2024. Empirical Study: Road Sign Classification. In *Proceedings of MSU CSE 881 - Project*. USA, 7 pages.

1 Introduction

The rapid advancements in autonomous driving technology have made the development of reliable and high-quality road sign detection systems essential for ensuring safety, efficiency, and overall system performance. Among the promising approaches to achieving reliable road sign detection are Computer Vision techniques and Deep Learning networks. Recent studies have demonstrated the potential of designing neural network architectures to achieve exceptionally high performance in this domain [1, 5, 7, 12, 13, 15, 18, 19].

In this project, we aim to evaluate the performance of various Computer Vision models in classifying road signs from a diverse set of images. We begin with collecting three primary distributions of road sign images. The first set of images is sourced from Google Images and Google Shopping, offering a broad and varied representation of road signs in diverse scenarios. While this dataset provides a comprehensive collection, it must be acknowledged that it may introduce noise and inconsistencies that could affect model performance. The second set of images is obtained from Kaggle and represents a more refined and high-quality dataset. This collection is expected to exhibit greater consistency and resolution, providing a sharper and more reliable benchmark for evaluating model performance compared to the noisier, web-scraped data. Finally, we incorporate additional synthetic images generated using a diffusion model to assess their potential as a supplementary training dataset in conjunction with real-world photographs. This is intended to demonstrate whether diffusion models can effectively augment the training image collection. After collecting all the images, we evaluate various Computer Vision models on the dataset. This includes traditional models such as VGG [17] and ResNet [4] and recent large Vision-and-Language Models (VLMs) like CLIP [14] and LlavaNext [9].

Based on our evaluation, the most effective training dataset combines Kaggle and Google images, offering a diverse range of image distributions. This combination improves the models' performance in both the Kaggle and Google evaluation portions, indicating that incorporating more diverse data could improve the model's effectiveness in this task. However, compromising synthetic images during training negatively impacts the model's performance in most cases. This finding suggests that the current diffusion model remains insufficient for integration into the data collection process. Furthermore, our results reveal a significant performance gap between fine-tuned and Vision-and-Language Models (VLMs). Surprisingly, pioneer CV fine-tuned models demonstrate superior adaptability to the dataset, achieving higher performance across all portions of the evaluation dataset. Nevertheless, VLMs exhibit promising performance even in a zero-shot setting, highlighting their potential as versatile and general-purpose models. These findings indicate that with more advanced prompting techniques, VLMs could achieve performance levels comparable to fine-tuned models, making them valuable candidates for future road sign detection and classification advancements.

Regardless of performance comparisons, all evaluated architectures demonstrate acceptable performance in the Google and Kaggle datasets. In this context, the integration of Computer Vision and Deep Learning Networks has proven transformative, offering robust solutions for the challenges posed by varying environmental conditions, diverse sign designs, and real-time processing requirements. These cutting-edge technologies represent the forefront of innovation, making them indispensable tools in the pursuit of truly autonomous driving systems.

Our contribution of this project can be summarized as follow, (1), (2),(3),(4)

2 Related works

[18] addresses the limitations of existing road sign detection methods, which typically predict only a small number of traffic sign categories, by employing Mask R-CNN and ResNet50 to scale detection capabilities to a larger variety of road signs. Similarly, [15] introduces CNN architectures integrated with a haze removal U-Net

*Both authors contributed equally to this research.

MSU CSE 881 - Project, Dec 03–05, 2024, East Lansing, MI

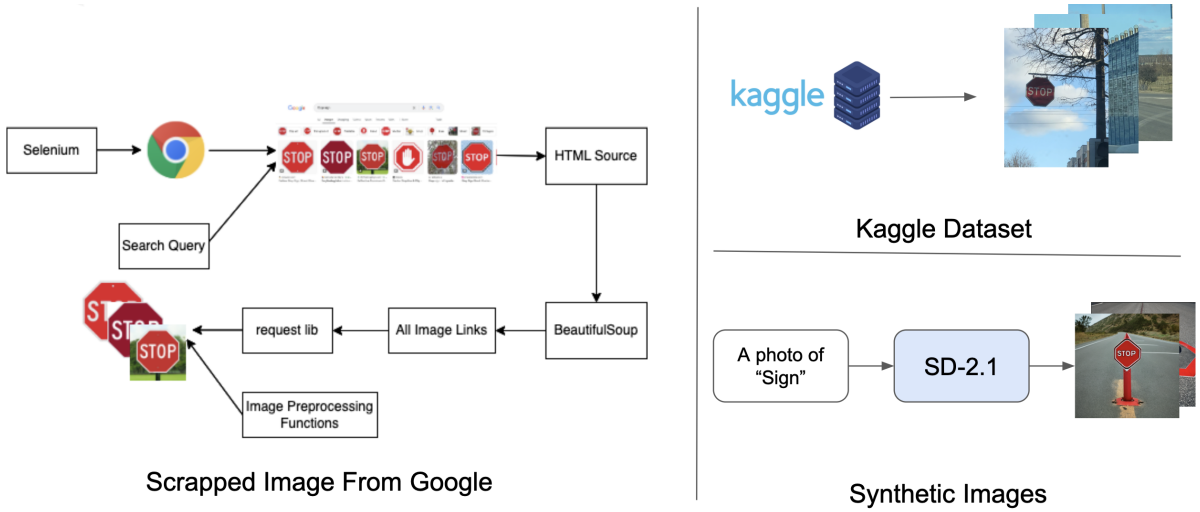

Figure 1: Three pipelines for our dataset collection, including scrapping images from Google, collecting the Kaggle dataset, and generating synthetic images.

model to enhance traffic sign detection and recognition in challenging environments with complex and noisy illumination, such as low-light or hazy conditions. Meanwhile, [12] tackles the simultaneous detection of traffic lights and signs by proposing a deep hierarchical architecture combined with a mini-batch proposal selection mechanism, ensuring efficient and accurate detection. [19] proposes a novel feature pyramid network for the YOLOv5 model, enhancing real-time multi-scale traffic sign detection and carefully evaluating it using the Tsinghua-Tencent 100K (TT100K) dataset. [20] conducts comprehensive experiments with a wide range of models, including ResNet, Inception, Inception-ResNet, MobileNet, and DarkNet, to assess traffic sign detection performance. [11] investigates adversarial attacks on traffic sign detection to improve model robustness. In conclusion, road sign detection remains an emerging research question that has attracted many researchers to propose refined methods, aiming to make it reliable enough for in-road usage in autonomous driving [1, 3, 5, 7, 13].

3 Methodology

3.1 Data Collection

All the pipelines of data collection used in our project illustrated in Figure 1

3.1.1 Google Images. We collected road sign images from two sources: Google Images and Google Shopping. Using the search query "... Sign", where "..." represents one of four label types in our project - Stop, Speed Limit, Cross Walk, and Traffic Light - we accessed images through specific website URLs for either Google Images[1] or Google Shopping[2]. To increase the complexity and variety of the dataset, we also expanded our queries with terms like "... Sign on Road" and "... Sign in City," creating a more challenging classification task.

Our scraping pipeline process is described as follows:

(1) Firstly, we utilized the Selenium library to automate web page interactions, including scrolling through the site to load additional images and capturing the complete HTML source.
(2) Next, we used Beautiful Soup library to parse the HTML and extract all image links from the <img> tags.
(3) Finally, we encountered two types of image links from Google: the first type contained direct image URLs, which we downloaded using the Requests library, and the second type was encoded in base64 format, which we decoded and saved using the Base64 library.
(4) We also remove low-quality images if their width or height is smaller than the specified thresholds.
(5) To efficiently download images, we used the multiprocessing library to download scraped images using multiple CPUs, thereby increasing the scraping speed significantly.

3.1.2 Kaggle Images. To enhance the model's performance, we augmented our dataset by incorporating additional images from the Kaggle Dataset[3]. This dataset comprises 887 images belonging to four distinct classes: traffic light, stop, speed limit, and crosswalk. For this dataset, we exclusively selected images featuring a single traffic sign per image to mitigate potential confusion during training and evaluation, as multiple signs may be present within the same image. The data statistic of this dataset after selection is provided in Table 1.

3.1.3 Synthetic Images. To assess the efficacy of generative models in providing images for training datasets, we included an additional set of images generated by Statble Diffusion 2.1 (SD-2.1) [16]. SD-2.1 is a diffusion model employed to create images from the provided context. We provided the context as "The photo of {road sign}," where the road sign represents the four distinct classes in our project: traffic light, stop, speed limit, and crosswalk. Subsequently,

[1]https://www.google.com/search?q=...+Sign&tbm=isch
[2]https://www.google.com/search?q=...+Sign&udm=28
[3]https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

we generated a total of 200 images per road sign and created a separate split named "**synthetic images**" to evaluate the impact of these images on the training process. The pipeline for this dataset portion is provided in Figure 1. It is important to note that this part of the dataset is only used in the training process.

3.2 Data Pre-processing

According to our data collection, there are three sources of data: Scraped Images from Google, images from the Kaggle dataset, and synthetic images. We encountered three issues that can be solved using the preprocessing images. The three problems are listed below,

(1) Image types mismatch. The scraped Google images come in various formats, such as PNG and JPEG, while the Kaggle dataset images are exclusively in PNG format. This disparity in image formats can impact the image processing steps. To address this issue, we employed the OpenCV library to convert all the images used in our experiment to PNG format, ensuring uniformity and facilitating seamless processing.
(2) Varied image size. To ensure that images are of the same dimensions for input to deep learning models, we use the OpenCV library to resize the images to 256x256. This process prevents the issue of different sizes of the image due to diverse sources.
(3) Different label for the same category. According to different sources of images, there are different annotations for the same labels. To resolve this, we manually check the image labels from various sources and transform them into our set of labels.

After resolving the above issues, we divided the images from each data source into Training and Testing sets. The training set comprises 80% of the total images, while the remaining 20% form the testing set. Subsequently, we categorized the image pre-processing steps into two groups, depending on the input type to the underlying models. The input for the convolutional neural network (CNN) based models is the image pixel values. We normalize the image pixels based on the mean and standard deviation of the image. Since our task is classification, which does not pay attention to spatial information, we also perform a random horizontal flip of the image during training. We only converted the image into the image embedding without pre-processing for the vision transformer that required image embedding.

4 Model Architectures

Here is a brief description of the different model architectures we used in our project.

4.1 VGG16

The VGG-16 model, proposed by Visual Geometry Group at the University of Oxford [17], is a convolutional neural network (CNN) architecture with 16 layers, consisting of 13 convolutional layers and 3 fully connected layers. Its design includes a series of convolutional layers followed by max-pooling layers, gradually increasing in depth. Despite its simplicity, VGG-16 achieved a top rank in the ImageNet Challenge and remains a popular choice for many computer vision classification tasks.

4.2 Resnet50

The ResNet-50 model, developed by the Microsoft Research team [4], is a widely used convolutional neural network (CNN) architecture with 50 layers. One of its key advantages is its ability to address the vanishing gradient problem in deep networks by introducing bottleneck residual blocks. These residual blocks allow the input of certain layers to "skip" ahead and add directly to the output of layers further along in the network, helping to preserve gradient information as it passes through many layers. ResNet-50 has been shown to outperform the VGG-16 model on the ImageNet dataset, achieving higher accuracy.

4.3 CLIP

CLIP, developed by openAI [14], is a neural network designed to learn the association between images and their corresponding text descriptions. It comprises two interconnected neural network models: a vision and a language model. Each model contains an encoder that converts images or natural language into embeddings. The model's objective is to maximize the similarity between the embeddings of matching image-text pairs and minimize the similarity between non-matching pairs. Notably, CLIP was trained to align the image embedding with the textual representation, significantly enhancing its comprehension of the relationship between visual and textual information. CLIP's performance demonstrated remarkable alignment with the original ResNet50 on ImageNet in the zero-shot setting. In our experiment, we use the CLIP in two different settings. The first setting uses CLIP to match the similarity score between class labels and images. Another setting is training the classification layer from the images embedded in CLIP.

4.4 BLIP-2

BLIP-2, a vision-language model developed in [8], presents an alternative to the resource-intensive large-scale models. It trains a lightweight, 12-layer Transformer encoder, called a Querying Transformer (Q-Former), between the image encoder and the large language model to enable the interaction between the image encoder and the language model. BLIP-2 achieved state-of-the-art results on several vision and image tasks after its release. One of BLIP-2's notable tasks is generating a zero-shot caption from an image. In our experiment, we utilize BILP2 as an image caption generator to provide additional information to LlavaNext, a larger and more powerful vision and language model. We hypothesize that this extra information would help LlavaNext to perform better.

4.5 LLavaNext

LlavaNext is an advanced multimodal model capable of performing various vision-language tasks [9]. Its complexity surpasses that of other models, enabling it to follow instructions. The model can perform multiple tasks based on the instructions provided. Notably, the performance of using only prompt engineering on the instructions surpasses that of multiple previous vision-language models in both visual comprehension and reasoning. In our project, we use LlavaNext as another baseline for performing the road-sign detection to show the recent performance of the VLM on this task.

5 Experimental evaluation

5.1 Dataset

We have two primary sources for Road Sign images: **Google** and **Kaggle**, and one additional source for Road Sign images: **Synthetic Images**. After merging the images from both primary sources, we split the combined dataset into 80% for training and 20% for testing. We evaluated model performance by reporting accuracy separately on the Google test set, Kaggle test set, and the combined test set. Additionally, we trained the models on four different configurations of training data: (1) images from only the Google training set, (2) images from only the Kaggle training set, (3) the combined training set from both sources, and (4) the combined training set augmented with **Synthetic Images**. The data statistic of all image sources used in our dataset is provided in Table 1.

Sign Type	Google Images	Kaggle	SD-2.1
Stop	2302	32	200
Speed Limit	2101	481	200
Cross Walk	2343	42	200
Traffic Light	2679	65	200

Table 1: Scraped Data Statistic

5.2 Fine-tuning Models

5.2.1 VGG16 and Resnet50. This experiment used pre-trained VGG16 and ResNet models, trained initially on the IMAGENET1K [2] dataset. To adapt these models for our 4-label classification task, we replaced their final layers with a fully connected layer comprising four output nodes. The pre-trained models were then fine-tuned over 20 epochs with our scraped dataset using the Adam optimizer [6] with a learning rate of 10^{-5}, and the Cross-Entropy Loss function was used for backpropagation.

5.2.2 CLIP Classification. In this experiment, we utilize the CLIP as the image encoder and employ its image embedding to perform the classification task. We load the CLIP encoder using SentenceTransformers library. Our selection of the CLIP model is `clip-ViT-B-32`. After obtaining the image embedding from CLIP, we forward the embedded into the classification layer (linear layer) with softmax activation to classify the final prediction of the image. We train the model using the AdamW optimizer [10], and the loss is CrossEntropy Loss. Two hyperparameters were utilized in this experiment. We select the optimal hyperparameters based on the accuracy of the validation portion. The sets of hyperparameters are listed as follows.

- learning rate: $\{1e^{-6}, 4e^{-6}, 8e^{-6}\}$
- number of the epoch: 60 (for our scraped Google images), 120 (for the Kaggle road-sign dataset.)
- batch size: 32
- optimizer: AdmaW

5.3 Pretrained Models

5.3.1 CLIP. For the experiment involving pre-trained models, we utilize CLIP as the standard embedding matching technique between a given image and potential classes. We construct the textual representation of each class as "a photo of {class} sign," where the class is one of the possible classes (stop, speed limit, crosswalk, traffic light). For the prediction of this experiment, we provide a single target image along with a list of all possible classes. Subsequently, we obtain the answer based on the maximum cosine similarity returned by the CLIP model, which corresponds to the highest degree of similarity between the embedded representation of the given image and the embedded representation of the most suitable textual representation. The pre-trained weights of CLIP are derived from the `clip-vit-base-patch32` checkpoint.

5.3.2 LlavaNext. We provide the experiment with one of the recent vision-language models (VLMs). We utilize the weight of the model from `llava-next-72b-hf`, which is the pre-trained weight for the instructed model of LlavaNext. We load the model with the 4-bit quantization to save the GPU memory, and it can run in Google Colab GPU by reducing the precision of model weights to 4 bits. We are providing the instruction prompt to the model as follows, "*You are the most powerful image classifier for classifying road signs. You will be given the image of the road sign. Then, you will classify the image into one of the correct answers. You can make a guess if it is reasonable. Answer without any explanation and in the format of Answer: Category. The answer is only one of {stop, crosswalk, speed limit, traffic light}.*". Then, we evaluate the model in a zero-shot setting that only provides the image without any examples. The target image is the input to the model, and the question is about the sign in the given picture. This experiment intends to assess the model's native performance on road sign detection. The model's temperature is set to 0 to ensure reproducibility.

5.3.3 BLIP + LlavaNext. In this experiment, we utilize BLIP as the caption generator. We select `blip2-opt-2.7b` as the model to generate the caption from the target image. We initially provided the image to BLIP to freely create the caption as augmented information. We hypothesize that this caption potentially encourages LlavaNext to classify the image accurately. We augment the caption before asking questions in the pipeline of the LlavaNext experiment by stating, "Also consider this following information to assist in classifying the image: {caption}." We maintain all other settings unchanged for this experiment.

6 Experimental Results

6.1 Quantitative Results

We provide all experiments results in Table 2. Therefore, we utilize these results to answer our interested research questions.

Is training with different image distribution affect performance? According to Table 2, the fine-tuning model performance varies significantly based on the training examples. There are two noteworthy observations. First, training/fine-tuning with Kaggle only leads the model to perform significantly poorly with Google images. This indicates that the photos in the Kaggle dataset are very specific and cover a small distribution. The lowest performance is in the CLIP classification, which can only achieve 33.35% accuracy, which is close to random in this dataset. Conversely, training with Google images provides acceptable performance in Kaggle datasets, which illustrates more generalizability of the model when training with this portion. Another aspect to consider is that training

and fine-tuning Google and Kaggle could potentially enhance the performance of both Google and Kaggle images. The outcomes of ResNet and CLIP classification serve as evidence to support this notion. This suggests that the model can learn from both distributions, encompassing a broader range of cases and improving performance.

Are synthetic images from diffusion model help classification? We provide the results by fine-tuning the model further using synthetic images from the diffusion model in VGG16 and ResNet experiments to answer this question. We observed that synthetic images generated from diffusion models can potentially help with sign classification tasks. Still, their effectiveness depends on how closely they match the distribution of the target dataset. In the VGG16 experiment, incorporating synthetic images enhanced performance on Kaggle datasets. This suggests that the synthetic images exhibited a similar distribution to the Kaggle images, effectively augmenting the dataset. However, the performance dropped significantly when the same synthetic images were used to fine-tune models for classifying Google images. This suggests that learning these synthetic images hurts classifying road signs from Google images. One possible reason is that both Google images and synthetic images contain similar pictures with different labels, which confuses the model on classification, unlike the Kaggle images, in which the synthetic images may encourage learning of this evaluation portion. Despite this, the synthetic images demonstrate more negative effects on the model than positive outcomes, implying the insufficient quality of images generated by the diffusion model for use as augmented data in the training process of this task.

What is the performance of the current VLMs on road sign classification? Based on the results, we observed that the vision-language model's performance is still far behind the fine-tuning model. However, LlavaNext provides promising results in the Kaggle dataset, illustrating a basic understanding of the road sign. Notably, the BLIP caption description highlighted adverse effects on the model. This highlights that the generated captions contain more extraneous information than valuable data for the model's road sign classification task. Consequently, in the road sign classification task, VLM remains inferior to the actual vision fine-tuning model in terms of performance. However, it demonstrates promising results. Therefore, employing more sophisticated prompt engineering techniques could potentially facilitate the enhancement to achieve a comparable level of performance to the actual vision-based model.

6.2 Qualitative Analysis

To further analyze the results of this task, we provide a qualitative analysis of the misclassified images for interesting models and identify four types of errors:

- Signs that are difficult to detect: Signs that are difficult to detect include those that are blurred, flipped, off-center, or out of focus, making them challenging to recognize.
- Multiple signs in a single image: Some images contain multiple different signs, which can confuse the model.
- Unusual or incorrect signs: These include images with incorrect signs, signs that are heavily distorted, or images that do not contain any signs at all (e.g., due to errors during the Google Images scraping process).
- Model incorrect generation: This is the only error unique to the LlavaNext model that model can generate an incorrect class.

We randomly selected 100 misclassified images for each vision model. We provide the percentage of each error in Table 3. Our findings indicate that the majority of errors originated from Type 3. This is primarily attributed to simple search queries that extract images from Google, often yielding irrelevant or mislabeled road sign images. This underscores the significance of human annotation in cleaning the dataset and eliminating noisy samples, thereby ensuring higher data quality for training and evaluation. Furthermore, Type 1 errors accounted for approximately one-fourth of the total error images. These errors primarily involved blurred, out-of-focus, or poorly centered images, indicating that the models could not detect such cases. This highlights the necessity of developing a comprehensive random augmentation strategy, including flipping and cropping photos, to assist the models in learning from challenging examples and enhancing their performance in such intricate scenarios.

7 Prototype Development

We developed a web-based interface using the Streamlit library to display our model predictions for road sign images. The interface allows users to easily switch between two model architectures, VGG-16 and ResNet-50, and upload their own road sign images for detection. Once an image is uploaded, the selected model generates predictions, providing a real-time road sign label.

For deployment, we used Streamlit Community Cloud as the deployment server, which deploys the Streamlit app directly from a GitHub repository. One challenge we faced was that the computer vision model weights were too large to be stored in the GitHub repository due to its size limitations. To address this, we used Google Cloud Storage to store the model weights and implemented a mechanism to download them only when needed. Figure 2 illustrates our web app deployment pipeline.

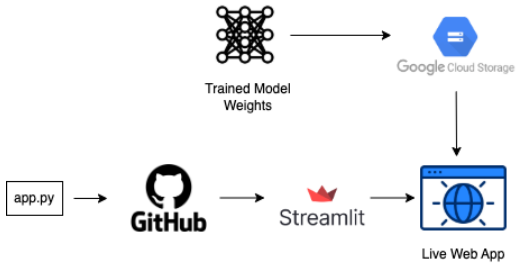

Figure 2: Deployment Pipeline

Figure 3 illustrates our web design, which includes a box for switching between the VGG-16 and ResNet-50 models, as well as buttons for uploading images. Figure 4 shows an example of the app predicting a "Stop Sign" using the VGG-16 model.

The public URL link is available at website.[4]

[4]https://appdeploytest-wv9jmdfc27xqm9crhs3uad.streamlit.app/

Model	Google images	Kaggle	Overall
VGG16 (Fine-tune on Google)	84.73%	66.67%	83.62%
VGG16 (Fine-tune on Kaggle)	51.75%	**97.56**%	54.55%
VGG16 (Fine-tune on all)	84.25%	94.31%	84.87%
VGG16 (Fine-tune on all + Image gen)	73.22%	94.60%	71.13%
ResNet (Fine-tune on Google)	85.15%	71.54%	84.32%
ResNet (Fine-tune on Kaggle)	53.08%	94.31%	55.60%
ResNet (Fine-tune on all)	**85.58**%	96.75%	**86.26**%
ResNet (Fine-tune on all + Image gen)	68.98%	89.43%	68.10%
CLIP (Train on Google)	83.08%	70.73%	82.33%
CLIP (Train on Kaggle)	33.35%	79.67%	36.18%
CLIP (Train on all)	83.19%	93.49%	83.83%
CLIP	73.12%	78.15%	77.15%
LlavaNext-72B (0-shot)	73.51%	90.00%	74.41%
BLIP + LlavaNext-72B (0-shot)	71.61%	72.72%	71.67%

Table 2: Accuracy of Different Computer Vision Model Architectures on Road Sign Dataset

Model	Type 1 Hard Examples	Type 2 Multiple Signs	Type 3 Unusual or incorrectly labeled images	Type 4 generation error
VGG16	22	20	58	0
Resnet	28	17	55	0
CLIP	37	16	47	0
CLIP Classifier	20	40	40	0
Llava	31	21	40	8
BILP + Llava	21	28	43	8

Table 3: Number of Images per Error in Misclassification Examples from Different Computer Vision Model Architectures

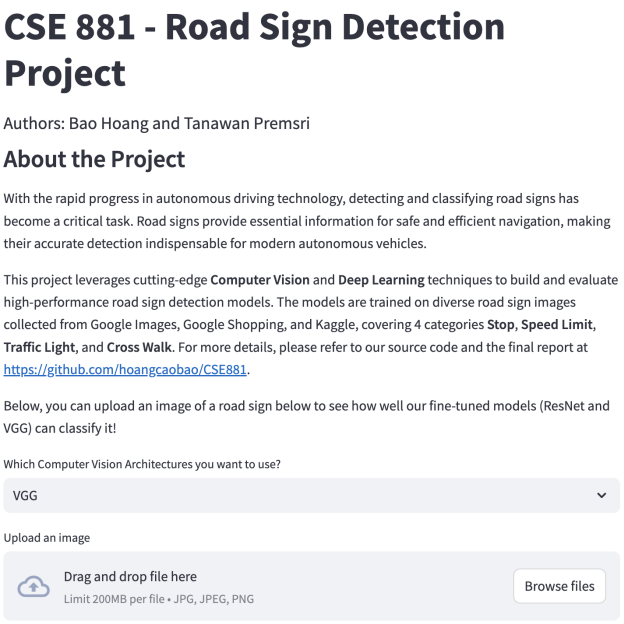

Figure 3: Web Interface

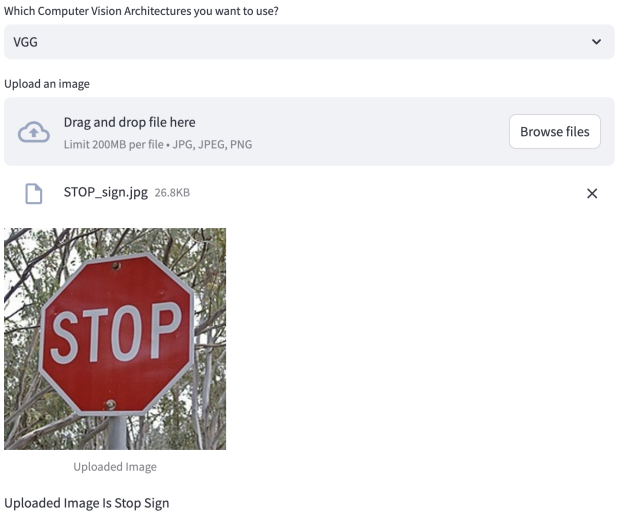

Figure 4: Prototype Prediction

8 Conclusion and Future Works

In this project, we accomplished the following tasks:

- **Automated Image Scraping:** Developed a pipeline to scrape road sign images from Google sources automatically to utilize them.
- **Model Implementation and Evaluation:** Implemented and evaluated the performance of advanced computer vision architectures in both vision-only (CLIP and VGG) and vision-language models (LlavaNext and BLIP).

- **Error Analysis:** Analyzed misclassified images to identify the reasons behind incorrect predictions made by the computer vision models.
- **Website Development:** Designed and deployed a real-time road sign prediction website using large model architectures like VGG16 and ResNet50.

Through this project, we gained several valuable insights:

- **Data Quality Matters:** A clean and well-labeled dataset is paramount to enhancing model learning. Our observations from both image scraping and synthetic image generation revealed substantial noise, which diminished model performance. Consequently, manual annotation is indispensable to eliminate noisy images from the scraped dataset.
- **Enhancing Model Robustness:** Addressing challenging cases like blurred or out-of-focus images to improve model robustness requires effective data augmentation strategies.
- **Deployment Considerations:** Deploying machine learning models in a web application requires careful handling of large model weight files. A viable strategy is to use cloud services to host model weights rather than embedding them directly in the front-end code.

We also point to future projects as follows:

- **Enhance the sign detection dataset by incorporating a more comprehensive range of images.** Based on our training data that utilizing both Google Images and Kaggle, we are confident that augmenting the dataset with an even greater diversity of road signs will substantially enhance the model's performance in this task. This project presents an opportunity to contribute to a more comprehensive dataset for this specific application. The development of this project necessitated substantial efforts in both human annotation and advanced data mining techniques, which we leave to explore in future endeavors.
- **Specialized modules.** One potential approach to enhancing road sign classification involves partitioning the task into smaller sub-tasks, such as sign detection and sign attribute detection. Subsequently, we employ the exported models for each sub-task to generate the outcomes, which are combined to construct a more robust model for sign detection. This approach presents an intriguing research opportunity to enhance sign detection performance further.
- **Better prompt engineering.** Based on the promising results of LlavaNext, even with the zero-shot learning approach, we hypothesized that a model with a more advanced prompting process could potentially reveal the significance improvement of the models. One potential approach is to perform the detection of a possible bounding box for the sign and crop the image accordingly. This will ensure that the model's attention is directed solely towards the sign, rather than other aspects of the image. However, this required a significant computing process for running the model with additional examples or information, which we leave for future research with the necessary equipment.

References

[1] Koorosh Aslansefat, Sohag Kabir, Amr Abdullatif, Vinod Vasudevan, and Yiannis Papadopoulos. 2021. Toward Improving Confidence in Autonomous Vehicle Software: A Study on Traffic Sign Recognition Systems. , 66-76 pages. https://doi.org/10.1109/MC.2021.3075054

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. , 248-255 pages. https://doi.org/10.1109/CVPR.2009.5206848

[3] Nicholas Gray, Megan Moraes, Jiang Bian, Alex Wang, Allen Tian, Kurt Wilson, Yan Huang, Haoyi Xiong, and Zhishan Guo. 2023. GLARE: A Dataset for Traffic Sign Detection in Sun Glare. arXiv:2209.08716 [cs.CV] https://arxiv.org/abs/2209.08716

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV] https://arxiv.org/abs/1512.03385

[5] Sami Jaghouar, Hannes Gustafsson, Bernhard Mehlig, Erik Werner, and Niklas Gustafsson. 2022. Improving Traffic Sign Recognition by Active Search. (2022), 594–606. https://doi.org/10.1007/978-3-031-16788-1_36

[6] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

[7] Amara Dinesh Kumar. 2018. Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks. arXiv:1805.04424 [cs.CV] https://arxiv.org/abs/1805.04424

[8] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. arXiv:2301.12597 [cs.CV] https://arxiv.org/abs/2301.12597

[9] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024. LLaVA-NeXT: Improved reasoning, OCR, and world knowledge. https://llava-vl.github.io/blog/2024-01-30-llava-next/

[10] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG] https://arxiv.org/abs/1711.05101

[11] Furkan Mumcu and Yasin Yilmaz. 2024. Fast and Lightweight Vision-Language Model for Adversarial Traffic Sign Detection. *Electronics* 13, 11 (2024). https://doi.org/10.3390/electronics13112172

[12] Alex Pon, Oles Adrienko, Ali Harakeh, and Steven L. Waslander. 2018. A Hierarchical Deep Architecture and Mini-batch Selection Method for Joint Traffic Sign and Light Detection. , 102-109 pages. https://doi.org/10.1109/CRV.2018.00024

[13] Andreea Postovan and Mădălina Eraşcu. 2023. Architecturing Binarized Neural Networks for Traffic Sign Recognition. arXiv:2303.15005 [cs.CV] https://arxiv.org/abs/2303.15005

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV] https://arxiv.org/abs/2103.00020

[15] A. Radha Rani, Y. Anusha, S.K. Cherishama, and S. Vijaya Laxmi. 2024. Traffic sign detection and recognition using deep learning-based approach with haze removal for autonomous vehicle navigation. *e-Prime - Advances in Electrical Engineering, Electronics and Energy* 7 (2024), 100442. https://doi.org/10.1016/j.prime.2024.100442

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695.

[17] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs.CV] https://arxiv.org/abs/1409.1556

[18] Domen Tabernik and Danijel Skočaj. 2020. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition. , 1427-1440 pages. https://doi.org/10.1109/TITS.2019.2913588

[19] Junfan Wang, Yi Chen, Mingyu Gao, and Zhekang Dong. 2021. Improved YOLOv5 network for real-time multi-scale traffic sign detection. arXiv:2112.08782 [cs.CV] https://arxiv.org/abs/2112.08782

[20] Álvaro Arcos-García, Juan A. Álvarez García, and Luis M. Soria-Morillo. 2018. Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing* 316 (2018), 332–344. https://doi.org/10.1016/j.neucom.2018.08.009